ADVANCES IN SCA AND RF-DNA FINGERPRINTING THROUGH ENHANCED LINEAR REGRESSION ATTACKS AND APPLICATION OF RANDOM FOREST CLASSIFIERS

DISSERTATION

Hiren J. Patel, Captain, USAF

AFIT-ENG-DS-14-S-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT-ENG-DS-14-S-03

ADVANCES IN SCA AND RF-DNA FINGERPRINTING THROUGH ENHANCED LINEAR REGRESSION ATTACKS AND APPLICATION OF RANDOM FOREST CLASSIFIERS

DISSERTATION

Presented to the Faculty

Graduate School of Engineering

and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Hiren J. Patel, BS, MS

Captain, USAF

September 2014

AFIT-ENG-DS-14-S-03

ADVANCES IN SCA AND RF-DNA FINGERPRINTING THROUGH ENHANCED LINEAR REGRESSION ATTACKS AND APPLICATION OF RANDOM FOREST CLASSIFIERS

DISSERTATION

Hiren J. Patel, BS, MS
Captain, USAF

Approved:

/signed/	29 Jul 2014
Michael A. Temple, PhD (Chairman)	Date
/signed/	1 Aug 2014
Rusty O. Baldwin, PhD (Member)	Date
/signed/	30 Jul 2014
Mark E. Oxley, PhD (Member)	Date
/signed/	30 Jul 2014
Christine M. Schubert Kabban, PhD (Member)	Date

Accepted:

ADEDEJI. B. BADIRU, PhD Dean, Graduate School of Engineering and Management	Date

AFIT-ENG-DS-14-S-03

Abstract

Radio Frequency (RF) emissions from electronic devices expose security vulnerabilities that can be used by an attacker to extract otherwise unobtainable information. Two realms of study were investigated, including the exploitation of 1) *unintentional* RF emissions in the field of Side Channel Analysis (SCA), and 2) *intentional* RF emissions from physical devices in the field of RF-Distinct Native Attribute (RF-DNA) fingerprinting. SCA is the study of physical characteristics of cryptographic algorithm implementation to exploit unintentional data leakages. By monitoring various side channels during an encryption or decryption, SCA attacks can deduce the secret key stored in a device that would otherwise be unobtainable to the attacker. RF-DNA fingerprinting uses physical layer features to enhance device authentication and security for distributive networks. This work advances the state-of-the-art by 1) improving SCA attack research in Linear Regression Attack (LRA), 2) increasing SCA key guess success rates with the non-parametric Random Forest (RndF) classifier, and 3) enhancing defensive methods for ZigBee distributed networks using RndF.

A type of SCA attack called LRA is improved by proper statistical analysis of the linear regression at each collected time sample. The conditions for regression are examined for the first time before proceeding to the attack phase. Attack performance showed an order of magnitude improvement when the dimensionality of the distribution estimated in the attack training phase is increased from 1 to 20, giving 98% success rate with as few as 100 test and training traces. In 8 out of 9 cases examined, this novel attack method performed better or equal to previous attacks from literature. Linear regression with an the adjusted coefficient of correlation R_a^2 indicator proved more effective in high-noise environments, requiring as few as 50 test traces at Signal to Noise Ratio (SNR)=15dB. This method was also the most successful at identifying variables with high data content in small training set conditions.

In the presence of non-Gaussian distributed data, attacks such as LRA and Template Attacks were found to be less effective due to their requirement of multivariate Gaussian noise in the collected traces. Examination of a 40-sample set of microcotrollers revealed that greater than 40% of the collected variables were non-Gaussian. Profiling attacks with the non-parametric RndF classifier for the full-dimensional data set consisting of 50,000 variables correctly extracted all 16 bytes of the AES key when the training and test devices were the same, and 15 bytes when devices differed. For a reduced set of the first 3200 variables captured during the encryption, Random Forest achieved success rates as high as 100% for attacks across 40 PIC microcontrollers from 4 different device families. With further dimensionality reduction, Random Forest still outperformed Template Attack for this data set, requiring fewer traces and achieving higher success rates with lower misclassification rates.

Finally, the use of a RndF classifier is examined for intentional RF emissions from ZigBee devices to enhance security using RF-DNA fingerprinting. Observations collected under practical conditions showed non-Gaussian variables which degraded parametric Multiple Discriminant Analysis/Maximum Likelihood (MDA/ML) classification performance. RndF was introduced into the RF-DNA fingerprinting arena and improved ZigBee device authentication was demonstrated. RndF outperformed parametric MDA/ML and non-parametric Generalized Relevance Learning Vector Quantization-Improved (GRLVQI) classifiers, providing up to G_S=18.0 dB improvement (reduction in required SNR). Network penetration, measured using rogue ZigBee devices, show that at the SNR=12.0 dB (correct classification rate %C=90%) the method correctly rejects 31 of 36 rogue access attempts based on Receiver Operating Characteristic (ROC) curve analysis and an arbitrary Rogue Accept Rate (RAR) of less than 0.1. This is better than MDA/ML and GRLVQI which only rejected 25/36 and 28/36 rogue access attempts, respectively. The key benefit of the RndF method

is improved rogue rejection in noisier environments - gains of G_S=4.0 dB and G_S=18.0 dB are realized over GRLVQI and MDA/ML, respectively.

This work is dedicated to my family. To my parents who never praised mediocrity but always honored higher education. To my wife who constantly encouraged me and occasionally gave me a reality check when research wasn't going well. And to my children who everyday made me feel like a king, whether I failed a test or aced it.

Acknowledgments

Special thanks to my advisor Dr. Michael Temple who taught me to see the usefulness of any research, and whose attention to detail and work ethic are unprecedented at AFIT.

I'd also like to thank Dr. Rusty Baldwin who mentored and guided me on how to conduct results oriented research. Also, I heartily appreciated the generous efforts of my other committee members, Dr. Christine Schubert Kabban and Dr. Mark Oxley in developing my research, giving me research ideas and improving my document drafts.

Finally, I would like to acknowledge the support from my fellow PhD students, for listening to my ideas and providing input when needed. More importantly, I thank them for keeping my moral high during tough research.

Hiren J. Patel

Table of Contents

Page

Page

List of Figures

List of Tables

Table Page

List of Algorithms

List of Acronmyms

ADC	Analog-to-Digital Converter
AES	Advance Encryption Standard
ANN	Artificial Neural Network
APS	Application Support
AUC	Area Under the Curve
CPA	Correlation Power Analysis
CDF	Cumulative Distribution Function
CEMA	Correlation Electro-Magnetic Attack
DPA	Differential Power Analysis
DRA	Dimensionality Reduction Assessment
ECDF	Empirical Cumulative Distribution Function
EM	Electro-Magnetic
ESD	Electrostatic Discharge
FFD	Full Function Devices
FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying
FVR	False Verification Rate
GRLVQI	Generalized Relevance Learning Vector Quantization-Improved
GSR	Global Success Rate
HD	Hamming Distance
HW	Hamming Weight
IO	Input-Output
I-Q	In-phase and Quadrature
IV	Intermediate Value
KNN	K-Nearest Neighbors
LFS	Learning From Signals

LRA	Linear Regression Attack
MAC	Media Access Control
MCA	Multi-Class AdaBoost
MDA/ML	Multiple Discriminant Analysis/Maximum Likelihood
MIC	Message Integrity Code
MTTF	Mean Time To Failure
NIST	National Institute of Standards and Technology
NWK	Network
OFDM	Orthogonal Frequency-Division Multiplexing
OOB	Out-Of-Bag
OOBE	Out-Of-Bag Error
O-QPSK	Offset Quadrature Phase Shift Keying
PCA	Principal Component Analysis
PHY	Physical
PUFs	Physically Uncloneable Functions
RAR	Rogue Accept Rate
RBF	Radial Basis Function
RF	Radio Frequency
RF-DNA	RF-Distinct Native Attribute
RFD	Reduced Function Devices
RFEI	Random Forest Entropy Importance
RMSE	Root Mean Square Error
RndF	Random Forest
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SCA	Side Channel Analysis

SEMA	Simple Electro-Magnetic Attack
SNR	Signal to Noise Ratio
SOST	Sum-Of-Squared pairwise T-difference
SPA	Simple Power Analysis
SSE	Sum of Squares Error
SVM	Support Vector Machine
TVR	True Verification Rate
Tx-Rx	Transceiver-to-collection Receiver
VI	Variable Importance
WGN	White Gaussian Noise
WPAN	Wireless Personal Area Networks
XEnv	Cross-Environment
XRx	Cross-Receiver

ADVANCES IN SCA AND RF-DNA FINGERPRINTING
THROUGH ENHANCED LINEAR REGRESSION ATTACKS
AND APPLICATION OF RANDOM FOREST CLASSIFIERS

I. Introduction

This chapter introduces the research topics and provides motivation behind developing attacks and countermeasures for unintentional and intentional Radio Frequency (RF) emissions.

1.1 Motivation

Motivation is presented in the following sections for the two main research areas: Side Channel Analysis (SCA) on unintentional emissions and RF-Distinct Native Attribute (RF-DNA) fingerprinting for intentional ZigBee devices.

1.1.1 Side Channel Analysis (SCA)

Modern cryptographic algorithms such as Advance Encryption Standard (AES) function under the assumption of complete key security. These algorithms are designed to prevent an attacker from decoding the secret key provided only the input *plaintext* and output *ciphertext* are available. However, when these algorithms are implemented in hardware such as a microcontroller, additional information becomes available to the attacker in the form of power consumption, Electro-Magnetic (EM) emanation, operation timing, etc. This additional *side channel* information is combined with the plaintext and ciphertext in SCA research to ascertain the secret key passively damaging the microcontroller. SCA has been very successful across a wide variety of hardware and for a variety of cryptographic algorithms [19, 34, 61, 84, 91, 92, 115]. Even masked

implementations where intermediate cryptographic values are concealed by a random value [72] have been defeated by higher order SCA attacks by combining multiple intermediate cryptographic functions [3, 4, 45, 90].

In the field of SCA, often individual bits or bytes are attacked one at a time. If these bits or bytes are stored in corresponding memory cells, then a power model can be developed where the power usage of the microcontroller is correlated to the state of the bits stored in these memory cells. For this type of attack, each bit is assumed to consume the same amount of power. However, slight differences between how adjacent memory cells are fabricated and routed lead to cases where power usage may not be equal in all bits. In [115], a SCA attack called the Linear Regression Attack (LRA) was developed to account for these differences. Since then, other research has advanced the state-of-the-art of LRA [34, 49, 50]. However, in each of these the assumptions for a valid linear regression were not verified. In addition, metrics used to identify a good model estimate fit to the collected data are often incorrect and can be unreliable. A formal study of the LRA method is needed to determine 1) if conditions for linear regression are present to allow a good linear model estimate, and 2) determine a proper metric to assess the linear model estimate fit that is more directly related to linear regression theory.

The most powerful SCA class of methods is the profiling attack, where the side channel leakage of a device is characterized or *modeled* on training hardware similar to that of the target hardware. If the side channel leakage is indeed similar, the side channel model can be used to estimate the unknown secret key in the target device. Work in [84] investigated conditions under which this assumption holds true. In most profiling attacks to date, the side channel leakage is theorized to be fixed, and the noise in the measurement assumed to be multivariate Gaussian [72]. However, in [72] it is shown that a microcontroller may emit leakage that is non-Gaussian. Work in [26] theorized that a non-parametric classifier may improve classification performance for a profiling attack and

thereby guess the correct key with fewer required target observations. Analysis of side channel data distribution is needed to determine if non-parametric methods are needed. Further, research comparing parametric and non-parametric profiling attacks are required to ensure attack success in the presence of non-Gaussian noise.

1.1.2 RF-Distinct Native Attribute (RF-DNA) Fingerprinting

A distributed network consists of many individual nodes that can be spatially separated. These networks allow tremendous flexibility by connecting users and devices and allowing either centralized or decentralized control of devices. ZigBee devices based on the IEEE 802.15.4 standard [56] offer low-power, low-cost communication alternatives and are ideal for short burst communication separated by long sleep intervals. The ZigBee protocol can be implemented in small devices allowing them to be low cost. Due to these advantages, ZigBee networks have found widespread adoption in the fields of building control [38], healthcare [58], and security systems [126]. However, their simplicity and distributed nature puts these devices at risk of network intrusion attacks. ZigBee security is commonly provided through the AES at the Media Access Control (MAC), Network (NWK) and Application Support (APS) layers. Security researchers however have developed several methods to exploit vulnerabilities in the ZigBee key-exchange process [33, 99, 127]. In addition, it has been proven that power consumption of authorized wireless sensor nodes can be passively monitored to determine the secret encryption key through Side Channel Analysis (SCA) methods [76, 108]. The ZigBee alliance has countered these threats by adopting AES in either Counter (AES-CTR) or AES-Counter with CBC-MAC (AES-CCM) mode which are more resistant to SCA attacks. However, AES-CTR mode has theoretically and practically been proven vulnerable to SCA eavesdropping attacks in [57] where the full AES-CTR key was successfully recovered. Thus key recovery in ZigBee devices is entirely possible and easily allows an attacker to insert rogue devices within an existing network.

RF-DNA fingerprinting provides a method to authenticate authorized devices on a distributed network by providing Physical (PHY) layer security. Slight hardware differences between nodes leads to corresponding variation in their emitted RF signals. The device-dependent variation can be exploited with various machine learning methods to identify nodes based on their emitted signals. This method has been successfully demonstrated in a wide variety of applications using classifiers to authenticate devices and identifying rogue devices [25, 35, 46, 107]. In [106] it was shown that non-parametric classifiers (Generalized Relevance Learning Vector Quantization-Improved (GRLVQI)) can outperform parametric classifiers (Multiple Discriminant Analysis/Maximum Likelihood (MDA/ML)) under certain circumstances. However, further research is needed to analyze ZigBee RF-DNA fingerprint features to determine potential for non-parametric classifier success in more general circumstances.

1.2 Research Contributions

Research contributions from this work support three main thrusts, as shown in Figure 1.1. First, advantages of using linear regression to estimate the power model versus using the static Hamming Weight (HW) power model used in Correlation Electro-Magnetic Attack (CEMA) SCA attacks are examined. A novel method to find time samples with high information leakage of sensitive data using the adjusted coefficient of correlation R_a^2 in a linear regression attack is introduced [92]. Three linear regression attacks from current literature [34, 50, 115] and CEMA [19] are compared with the new method for 9 different cases, where the effect of the number of dimensions, training traces and test traces on attack success are examined. A *trace* is the measured observation (emission collection) for one complete encryption or decryption operation. The advantage of using R_a^2 to identify a good fit of the linear model estimate in high noise and small training set conditions is highlighted over previous methods.

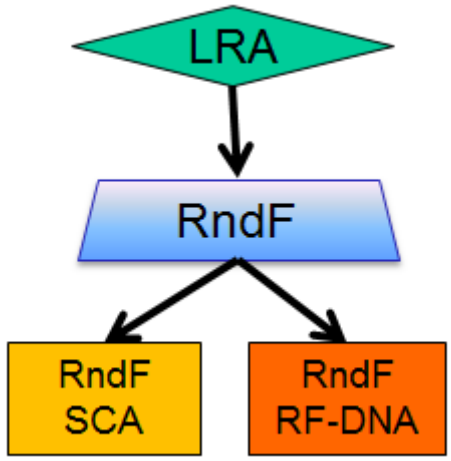


Figure 1.1: Summary of research contributions.

Second, conditions are examined where non-linear relationships exist between measured side channel data at a time sample and the sensitive information being attacked [91]. These are caused by non-Gaussian distributed data, which leads to very poor fit to a linear model estimate. *Cross-device attacks* were defined in [83, 84] as attacks performed with training and test data from different devices. An alternate non-parametric Random Forest (RndF) classifier was used instead in cross-device profiling attacks on 40 microcontrollers and benefits over parametric Template Attacks is demonstrated. Success was achieved in very high-dimensional data sets where Template Attacks could not mathematically function without collecting more training traces. Distribution analysis of the training data from 40 microcontrollers with Random Forest variable importance revealed that many with high information leakage had non-Gaussian distributions. Cross-device attacks with Random Forest and Template Attack after variable reduction were compared. Random Forest showed equal or higher success rates, lower misclassification rates and smaller number of required test traces than Template Attacks using Gaussian only, non-Gaussian only and mixed data.

Table 1.1: Summary of research contributions relative to prior work.

Technical Area	Previous Work		Current Work	
	Addressed	Ref #	Addressed	Ref #
Linear Regression Attacks				
Train/Attack: Same Device	X	[34, 50, 115]	X+	[92]
Train/Attack: Different Devices			X	[92]
Validate Regression Conditions			X	[92]
Analyze Performance Under High Noise Conditions	X	[34]	X	[92]
Analyze Performance Under Small Training Set Conditions	X	[50]	X	[92]
SCA Profiling Attacks				
Gaussian Noise Assumption	X	[24, 84, 103]	X+	[91]
Non-Parametric Classifiers	X	[51, 55, 66]	X+	[91]
Different Training/Test Devices	X	[84]	X+	[91]
Large Number of Dimensions			X	[91]
RF-DNA Fingerprinting				
MDA/ML, GRLVQI	X	[25, 46, 47, 107]	X	[93, 94]
RndF, MCA			X	[93, 94]
Dimensional Reduction Analysis (DRA)	X	[47]	X+	[93, 94]
ZigBee	X	[35, 36, 106]	X+	[93–95]

Third, the advantages of using RndF with intentional RF emissions from ZigBee devices is examined [93, 94]. RF-DNA fingerprinting has shown success over a wide range of *SNR* for Cognitive Radio [47], WiMAX [107], and 802.11a WiFi [48] applications. In all of these, the parametric MDA/ML classifier performed well with Gaussian features. However, when ZigBee device emissions were collected under a variety of conditions, non-Gaussian variables were better classified with RndF. Correct classification performance improved over previous classifiers under all SNRs tested. In addition, network intrusion detection of rogue devices impersonating authorized devices improved with RndF, which consistently identified the largest number of rogue intrusion scenarios, regardless of

the variable reduction method used, and in noisier environments than previously used classifiers. Finally, RndF was used with ZigBee emissions collected from high-cost and low-cost receivers and cost-benefit trade-off for classification and rogue identification was analyzed.

Table 1.1 highlights the research contributions in each thrust area and shows their relationship to prior related work. The symbol X+ is used to highlight areas where this research enhances prior work.

1.3 Organization

The remaining chapters are organized as follows. Chapter 2 provides background information SCA, RF-DNA fingerprinting, and machine learning. Chapter 3 gives general background in the three main research areas. Chapter 4 describes the results on LRA with the R_a^2 method as presented in [92]. Chapter 5 provides results on SCA profiling attack with the RndF classifier as presented in [91], and also includes new research into the non-Gaussian nature of the SCA collected variables. Chapter 6 details results from [93, 94] where a comparative assessment of RndF with other classifiers is examined. Also, results comparing high-cost and low-cost receiver performance with RndF is presented [95]. Finally, Chapter 7 summarizes the main contributions and provides suggestions for future research.

II. Background

2.1 Introduction

Intentional and unintentional emissions from physical devices are used in this work to enhance offensive and defensive cyber physical capabilities. Side Channel Analysis (SCA) discovers and exploits unintentional information leakages from the physical implementation of devices and specifically in this research, cryptographic algorithms. Algorithms such as Advance Encryption Standard (AES) are publicly known and the strength of their security depends on the secret key used to encrypt the plaintext data. If the safety of this secret key is assumed, cryptographic algorithms such as AES are mathematically secure and can adequately protect the sensitive plaintext information. Thus, the weak link in cryptographically secured data is its implementation. Side channel attacks the assumption of secret key security. It examines unintended physical information leakages from the hardware running the cryptographic algorithms to extract the secret key.

Intentional emissions from wireless devices can differ due to hardware differences between their analog components, such as the internal oscillator, antenna, amplifier, frequency mixer and band-pass filter [29]. These emissions can be used to generate unique profiles of each device, offering authentication that is prohibitively difficult to counterfeit.

This chapter provides background for the AES algorithm, followed by SCA and associated attacks, namely simple and differential SCA attacks and profiling attacks. Following this is background material on pattern recognition and classification methods. Finally, background information on RF-Distinct Native Attribute (RF-DNA) fingerprinting and ZigBee devices is provided.

2.2 Advance Encryption Standard (AES)

In 2000, the US National Institute of Standards and Technology (NIST) chose the Rjindael algorithm as the Advance Encryption Standard to replace the Data Encryption Standard (DES) [72]. AES can be implemented in 128, 192 or 256 bit versions with the type determining the secret key length. For the AES-128 algorithm, a 128 bit block of plaintext is represented as a 4×4 matrix of bytes known as the state matrix. The AES algorithm is composed of several *rounds* which are groups of functions. In each round the state matrix is combined with a round-key matrix of equal size for encryption. The round key is calculated via a key schedule. The pseudo code for the AES-128 algorithm is presented in Algorithm 1.

Algorithm 1 AES-128 algorithm pseudo code

```
AddRoundKey
for i = 1 → 9 do
    SubBytes
    ShiftRows
    MixColumns
    AddRoundKey
end for
SubBytes
ShiftRows
AddRoundKey
```

The four main operations of the AES algorithm are AddRoundKey, SubBytes, ShiftRows, and MixColumns. These operations are known as *intermediate functions* as they are intermediate to the input plaintext and the output ciphertext. The results generated from executing these intermediate functions are called the Intermediate Value (IV).

AddRoundKey In the AddRoundKey function, a bit-wise xor is performed with the 4×4 state matrix and the round key. This is given by,

$$AddRoundKey(x, k) = \mathsf{xor}(x, k), \tag{2.1}$$

where x and k are the plaintext and key of the same number of bits.

SubBytes In the SubBytes operation, the AddRoundKey output is used as an index into a substitution matrix known as the *S-Box*. The S-Box calculation for a given input byte x is

$$S(x) = Ax^{-1} + b, \tag{2.2}$$

where the matrix A and vector b definitions are given in [86]. Due to the computational complexity of calculating the S-Box, it is often pre-calculated and stored as a matrix for indexing into during execution.

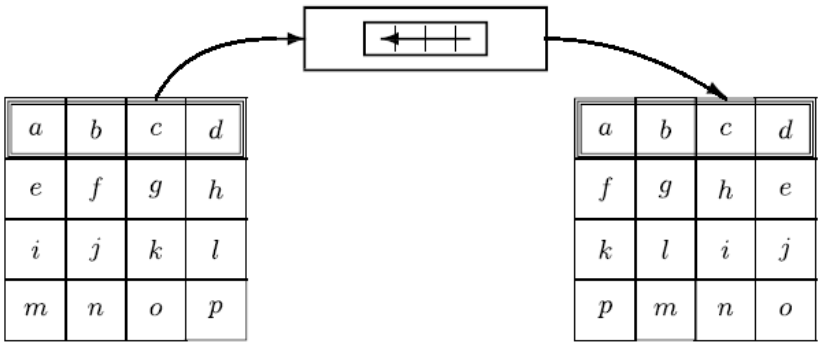


Figure 2.1: ShiftRows operation in the AES-128 Algorithm [72]

ShiftRows The Shiftrows operation is a byte-wise circular left-shift of the contents of the state-matrix. This means that for each row, contents are shifted left and the left most byte is wrapped shifted to the right-most byte. The amount of shift is 1−(row number). So for the first row, the contents are not shifted. For the second row, contents are circularly shifted 1 position to the left. The contents of the third row are shifted left twice, and those of the fourth row are shifted left three times. This is shown in Figure 2.1.

MixColumns The data in the columns of the state matrix are mixed in a manner equivalent to performing a matrix multiply as shown in Figure 2.2. The SubBytes operation provides non-linearity to AES, while ShiftRows and MixColumns *diffuse* round

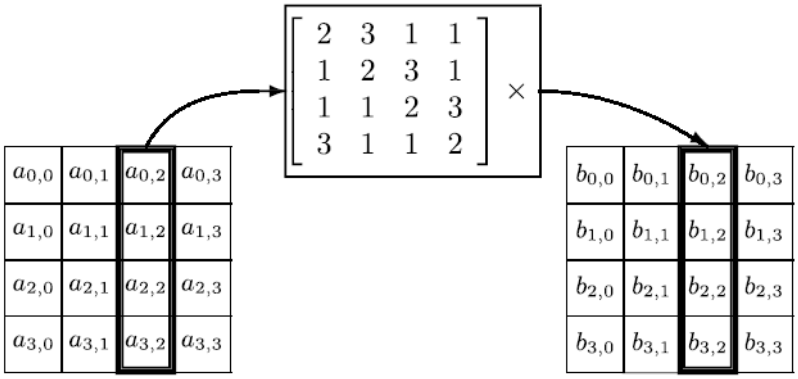


Figure 2.2: MixColumns operation in the AES-128 Algorithm [72]

key information. AddRoundKey adds *confusion* to the plaintext by obscuring it with the xor function. *Confusion* and *diffusion* are metrics described in [116] that describe how information is obscured and hidden by a cryptographic algorithm. The AES-128, 192 and 256 algorithms consist of 10, 12 and 14 rounds respectively to achieve sufficient crytographic strength through *diffusion* and *confusion* properties.

Key Schedule The Key Schedule operation, also referred to as *Key Expansion* [28], uses the original key to generate a unique key for each round. The Key Schedule represents the 128-bit key as a 4×4 key matrix and processes one column at a time. It consists of three steps:

1. RotWord: The last row of the 4×4 round key state matrix is left shifted by one position. This is similar to the ShiftRows operation for the plaintext, but is only executed on the last row the state matrix.

2. SubWord: In this step, the SubBytes operation is performed on column's four bytes.

3. RCon: Perform a bitwise xor with the round constant. A bitwise xor is again performed with the first column of the previous round key.

Thus, the key schedule follows a systematic process to generate new round keys based on the previous round key. This systematic process can be reversed for decryption.

2.3 Side Channel Analysis (SCA)

Side Channel Analysis is the study of observable physical phenomena such as timing, voltage, current and EM radiation to determine its relationship to sensitive information processed in hardware. These phenomena are called the *side channels* [72]. By observing these, the attacker can gain knowledge of sensitive information within the device, such as the secret key in a cryptographic algorithm. A brief description of power and EM side channels is provided in the following sections.

2.3.1 Power Side Channel

The data being processed by a logic circuit can be correlated to its power usage [72]. Figure 2.3 shows the Register Transfer Logic (RTL) schematic of a CMOS inverter. The inverter draws a fixed small amount of power when it is not switching, i.e., when the input goes from 1→1 or 0→0. This fixed power draw when there is no switching is termed *static switching* and represented by P_{stat} [72]. When the power switches from a logic 0→1 or 1→0, additional switching power, termed dynamic power P_{dyn}, is consumed [72]. Table 2.1

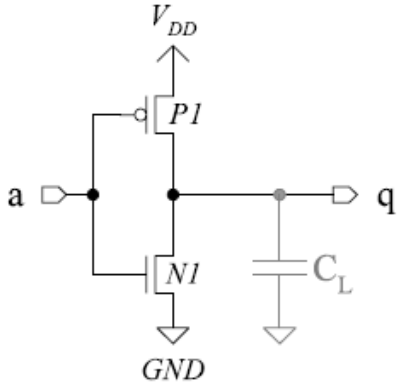


Figure 2.3: CMOS Inverter [72]

shows a breakdown of these observations. Thus, when the data is changing to a new state, there is a spike in the power usage of the inverter. If the previous state is known, then the new state can be determined by monitoring the power. This is the essense of power analysis.

Table 2.1: Power consumption of a CMOS inverter for different logic transition states.

Transition	Type of Power Consumption
0→0	P_{stat}
0→1	$P_{stat} + P_{dyn}$
1→0	$P_{stat} + P_{dyn}$
1→1	P_{stat}

This model can be scaled to work on memory cells and data buses as well. If a memory cell starts with an initial logic 0 state, writing a logic 1 will consume more power than writing a logic 0, according to Table 2.1. For a buffer of one byte, memory usage will be directly proportional to the number of logic high bits. The power model generated from this direct relation is called the Hamming Weight (HW) power model, where the HW is simply the total number of logic high bits. For a data bus, the power consumption due to data only leaks when the previous state and the current state do not match. Similar to the case of the transistor, the power consumption by the bus is proportional only to the number of 0→1 and 1→0 transitions. The total number of bits that are different between two states is called the Hamming Distance (HD) and the power model based on HD is called the Hamming Distance power model. If the previous state of the bus can be guessed, then combined with the HD, the new state can be uncovered. Additionally, the HW of the two states can be used to calculate the HD by

$$HD(state1, state2) = HW(state1 \oplus state2). \tag{2.3}$$

Although these models work well for many implementations, they can sometimes fail if there is not a direct relationship between the model and the power traces. The model assumes that all memory cells draw the same amount of power, which may not be true due to manufacturing differences. The model also assumed equal and negligible power draw for 1→1 or 0→0 transitions, which may not be the case for real devices. These deviations can lead to divergence of actual power measurements from associated HD and HW power models.

2.3.2 Electromagnetic Side Channel

When current passes through parts of a device such as Input-Output (I/O) wires, logic circuits and memory units, unintended EM emanations are generated. For synchronous circuits, change of logic states occur at the change of the clock. If the circuit of interest is synchronous, its strongest EM side channel occurs at the clock frequency.

There are two main types of unintentional EM emanations: *direct emanation* and *indirect emanation* [2]. Direct emanation results from current draw as the circuit changes its logic state. This type of emanation is observable over a wide band of frequencies, but often requires miniature probes placed very close to the logic source and depackaging the device [2, 98].

Indirect emanations occur due to EM coupling between components in the circuit located close together. These emanations are often in the form of modulation of one circuit's direct emanations onto a carrier signal from another stronger source such as the clock signal, which is the strongest signal on a synchronous circuit [21]. Then through Amplitude Modulation (AM), the carrier signal can be removed and the leakage from the circuit of interest can be attained. This side channel lends itself very well to longer distance signal exploitation and usually does not require precise probe placement near the actual logic circuit of interest or depackaging the device [2].

2.4 Side Channel Attacks

Side channel attacks exploit information captured from SCA to reveal protected information such as the secret key from cryptographic algorithms. Three main attacks are presented here, namely Simple SCA attacks, Differential SCA attacks, and profiling attacks. First, an explanation of terms used in following sections in regards to SCA is presented.

2.4.1 Random Variables in SCA

Side channel attacks require collecting information in the time domain. For this research the collected EM measurements were continuous time signals that are sampled, resulting in n random variables for observation trace x_i. Thus, x_i is an *ensemble* of n random variables. A collection of K such observations is represented in a *trace matrix* format as shown in Figure 2.4. Here, the random variable t for all traces x are stored in the same column of this matrix. The rows represent a trace or ensemble x_i. Now as each random variable t is actually a time sample, a column of the trace matrix represents the value collected for that instant in time for all the observations. The value of random variable t_i for trace x_i is functionally represented as $x_i(t_i)$. For the remainder of this document, the term *SCA data* will stand for this type of sampled data represented in such a trace matrix.

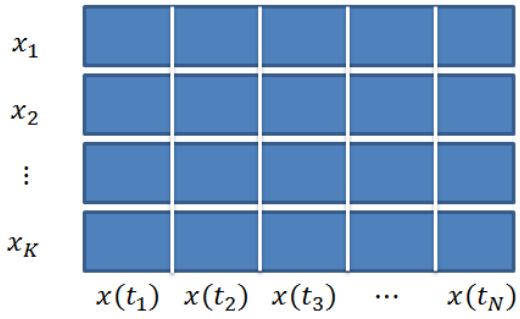


Figure 2.4: Trace matrix of SCA collected data. A row represents an observation trace x_i consisting of n random variables which are sampled in time.

2.4.2 Simple Side Channel Analysis

When the side channel measurement directly depends on the sensitive information, this sensitive information can be recovered by visual inspection of the trace. If the side channel of interest is power, this type of attack is referred to as Simple Power Analysis (SPA). For example, RSA encryption performs a squaring function and a multiplication when the key bit is a 1 and just the squaring function when the key bit is 0 [112]. The part of the RSA algorithm highlighting this vulnerability is shown in Algorithm 2 [82]. In

Algorithm 2 RSA Algorithm Pseudo-code

Input: X, N, E
Output: $Z = X^E mod N$

$Z = 1$
for i = *key bit* **do**
  $Z = Z \times Z \bmod N \leftarrow$squaring function (S)
  if $bit_i = 1$ **then**
    $Z = Z \times X \bmod N \leftarrow$multiplication (M)
  end if
end for

Algorithm 2an extra multiplication step (M) is performed only if the key bit value is a 1, while the squaring step (S) occurs regardless of the bit value. This equates to a different power consumption when a key bit value 1 is processed. Figure 2.5 shows the power consumption trace for an RSA encryption on an Field Programmable Gate Array (FPGA). The multiplication step is clearly visible and hence the key can be determined directly via observation. Usually with SPA, many traces can be averaged to remove the effect of noise on the power consumption and make the key dependent power consumption more easily visible. Simple SCA with the EM side channel is called Simple Electro-Magnetic Attack (SEMA).

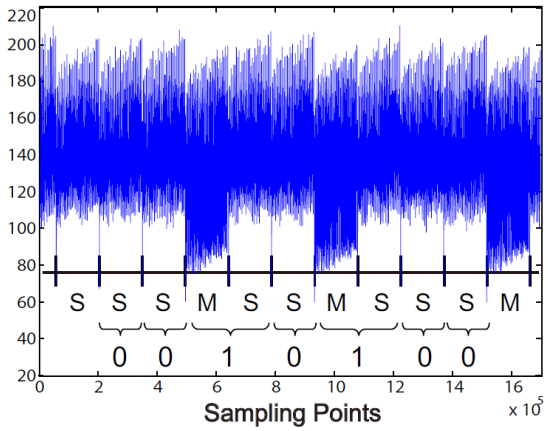


Figure 2.5: SPA on RSA algorithm for an FPGA [82]

2.4.3 Differential Side Channel Analysis

For algorithms that do not directly influence the device side channels, more sophisticated attacks are required. An example is the AES algorithm. As described in Section 2.2, the first step of the algorithm is a bit-wise xor of the plaintext with the secret key. If the result is stored in memory, the information leakage through the power consumption will be dependent on the resulting value, *not* on secret key. Thus, the secret key cannot be directly determined by examining the power consumption of a trace as in SPA. In addition, as different plaintext is being combined with the key for each trace, the first AddRoundKey step will produce different results for each trace. Thus, averaging traces as in simple SCA attacks will minimize the information leakage. In such circumstances, a stronger attacks based on differntial SCA are used.

The first differential SCA attack was developed in [61] using the power side channel and named Differential Power Analysis (DPA). A sub-category of differential SCA uses correlation to determine information about an algorithm running on a device. In the case of the AES algorithm, Correlation Power Analysis (CPA) can be used to recover the secret key from a microcontroller [19, 72] by collecting a large number of power traces

while the device runs the encryption algorithm. For electromagnetic traces, a similar attack Correlation Electro-Magnetic Attack (CEMA) is used [2]. It assumes that the power consumption of the microcontroller follows the HW power model [72]. Correlation is calculated between the theoretical power consumption represented as the HW of an intermediate value of the encryption and the actual power measurement. As there is a one-to-one mapping between the key and the intermediate value, the key corresponding to the highest correlation is guessed as the correct key. The methodology for CEMA on a microcontroller running AES-128 consists of 5 steps [72]:

1. **Choosing an intermediate result of the algorithm:** CEMA uses a divide-and-conquer approach, attacking a large key by segmenting it down into smaller sections. For example, a 128 bit AES key can be attacked by breaking it up into 16 one-byte sub-keys and attacking each sub-key individually. So the first step in CEMA against AES is to determine which intermediate function and sub-key to attack. This intermediate function can be represented by $\phi(x, k)$ where x is known, non-constant data such as plaintext or ciphertext, and k is the sub-key.

2. **Measuring the Side Channel:** Traces can be captured via a current or Electro-Magnetic (EM) probe using an oscilloscope with storage for post-processing.

3. **Calculate hypothetical intermediate values:** For an 8 bit sub-key, there are 256 possible values. An attacker would use each of the possible $k \in \{0, \ldots, 255\}$ values with D different plaintexts of known values of $x \in \{0, \ldots, 255\}$ to determine what the hypothetical values $\phi(x, k)$ for each key guess would be.

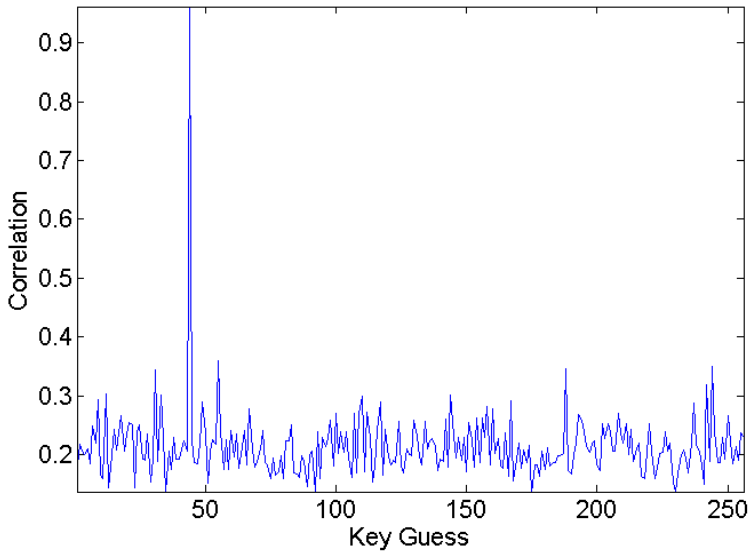
4. **Mapping intermediate values to leakage model values:** These hypothetical values are mapped to a leakage model such as the Hamming Weight model to estimate the side channel leakage for the component for each key value.

5. **Compare hypothetical values to side channel values:** The estimated leakage values are now compared to the real side channel measurement values in the traces. If the side channel follows the leakage model, the hypothetical leakage for the correct 8-bit sub-key will have the highest correlation with the collected traces. When the value of the correct key byte is unknown comparing correlation with the leakage model for each possible key byte reveals both the correct key value and samples that are correlated with the intermediate value being targeted.

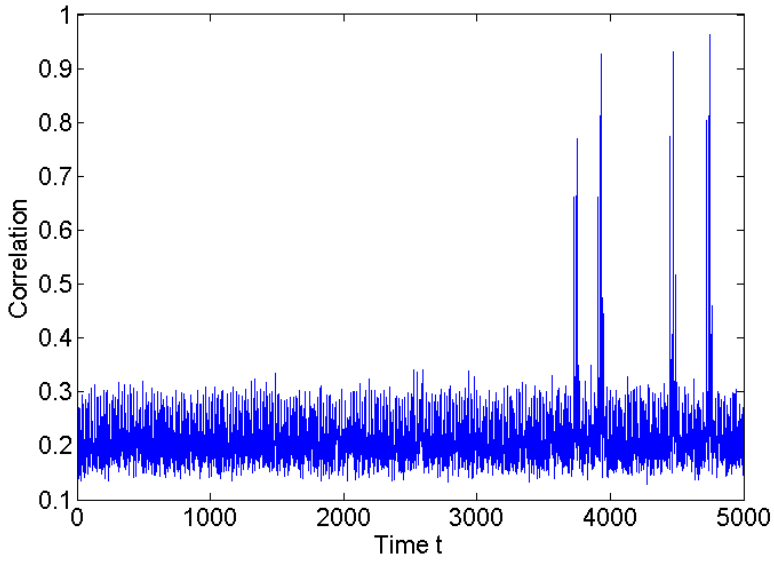
Figure 2.6a shows the maximum correlation of the actual power consumption to the hypothetical power consumption of AddRoundKey1 with the 256 possible sub-key values. The experiment was conducted using a 16 bit microcontroller. Sub-key guess 43 has the maximum correlation and therefore most likely to be the correct key guess for the first key byte. Plotting the maximum correlation over time as shown in Figure 2.6b shows the points in time where the maximum leakage occurs. The advantage of CEMA is that it does not require profiling and is not computationally expensive. A major disadvantage is that CEMA performance is dependent on the power model being accurate, which is often not the case for different hardware. In addition, CEMA often requires many more traces on the target device than profiling attacks.

2.4.4 Profiling Attacks

The most powerful class of side channel attacks used today is the *Profiling Attack*. If an assumption can be made that the side channel leakage between devices are similar, then knowledge gained by studying one device can be used to attack another. When the target hardware and encryption implementation is known, the side channel leakage during encryption can be modeled on separate hardware that is of the same or similar type using known secret keys. This similar hardware is termed the *training* hardware. With the known plaintext and key used in the training device, the Intermediate Value (IV) values $\phi(x, k)$ can be calculated for each trace. Next the side channel leakage for each IV value is



(a)



(b)

Figure 2.6: Maximum correlation values versus key guess (a), and trace time (b).

modeled with the training traces. Once the model is generated, it can be applied to the target hardware, also known as the *test* hardware to determine the unknown IV value for

each trace. This process is shown in the block diagram in Figure 2.7. As there is a one-to-one relationship between the guessed IV and the key for a known plaintext, the IV guess can be used to determine the key using the inverse intermediate function $\phi^{-1}(x,k)$ given by the decryption algorithm. thus, the profiling attack can be thought of as a classification problem with side channel leakage variables and IV values as the classes. In a *bit-wise* profiling attack, the side channel leakage of one bit of the IV is modeled at a time, yielding a 2-class classification problem. In a *byte-wise* profiling attack with 256 possible IV values for a byte, the profiling attack is a 256-class classification problem. Although profiling attacks require many training traces, they usually require fewer test traces than CPA to determine the correct key.

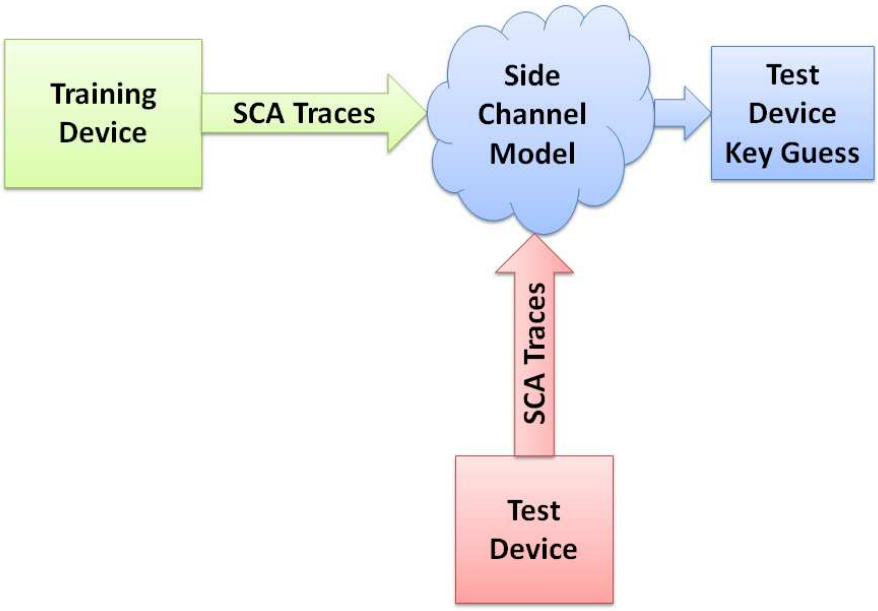


Figure 2.7: Profiling attack methodology

2.5 Classifier Description

Two main categories of classifiers are used in this work: parametric classifiers and non-parametric classifiers. Bishop defines parametric distributions as those "governed

by a small number of adaptive parameters, such as the mean and variance in the case of a Gaussian (distribution)" [14]. Parametric classifiers, by extension, are those that classify observations by assuming class parametric distributions and estimating their distribution parameters. Non-parametric classifiers, in contrast, do not assume an underlying distribution to the features of the data and therefore classify observations solely from the observed values in the training set. Parametric classifiers are able to estimate decision boundaries for regions in the variable space not covered by the training observations, provided the data follows a parametric distribution and specifically the distribution assumed. Non-parametric classifiers are able to estimate decision boundaries for data that does not follow a known distribution, but like mis-specified parmetric classifiers, boundary decisions can be inaccurate for regions in the variable space not included in the training set.

2.5.1 Parametric Classifiers

Several parametric classifiers were used or referenced in this work. Their descriptions are included here.

2.5.1.1 Multiple Discriminant Analysis/Maximum Likelihood (MDA/ML)

MDA/ML is a parametric classification process consisting of a transformation (MDA) followed by a parametric classification decision (ML). MDA is the multi-class form of Fishers Linear Discriminant Analysis [40] that seeks the direction $\boldsymbol{w}\in\mathbb{R}^M\times\mathbb{R}^{c-1}$ for a c-classes that is a linear transformation of $\boldsymbol{x}\in\mathbb{R}^M$ shown in [125]

$$\boldsymbol{y} = \boldsymbol{w}^T\boldsymbol{x} \quad . \tag{2.4}$$

The *within-class scatter* is defined by

$$S_w = \sum_{i=1}^{c} p(i)\Sigma_i \, , \tag{2.5}$$

where $p(i)$ is the prior probability for class i, and Σ_i is the class covariance matrix estimated from the training data. The *between-class scatter* is defined as

$$S_b = \sum_{i=1}^{c} = p(i)(\mu_\mathbf{i} - \mu_\mathbf{0})(\mu_\mathbf{i} - \mu_\mathbf{0})^T \ , \tag{2.6}$$

where μ_0 is the global mean vector from the training data given by

$$\mu_\mathbf{0} = \sum_{i=1}^{c} p(i)\mu_i \ . \tag{2.7}$$

The optimal direction is obtained by maximizing the Fisher's Discriminant Ratio (FDR) given by [67]

$$FDR(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_b \boldsymbol{w}}{\boldsymbol{w}^T S_w \boldsymbol{w}} \ . \tag{2.8}$$

$FDR(\boldsymbol{w})$ is also called the *Rayleigh coefficient* [67] and is maximized by solving the generalized eigenvalue problem

$$S_b \boldsymbol{w} = \lambda S_w \boldsymbol{w} \ . \tag{2.9}$$

The projection matrix $\boldsymbol{w}$ is the resulting matrix of eigenvectors corresponding to the $(c-1)$ largest eigenvalues of $S_w^{-1} S_b$. The projection matrix tries to find the subspace that maximizes inter-class mean distance while reducing intra-class variation and improving overall class separability. This transformation also effectively reduces the N_F-dimensional input data to an $(c-1)$ dimensional space. Given that MDA uses linear eigenvector decomposition to generate transformed features, and assuming Gaussian input features, the transformed features are Gaussian as well. However, if non-Gaussian input is provided the transformed features are not guaranteed to be Gaussian.

The MDA transformed features are classified via a ML decision assuming a multivariate Gaussian distribution of projected data. A mean vector μ_i is estimated for each class, along with a pooled covariance matrix Σ that is used for all classes. Likelihood estimation is calculated for a multivariate Gaussian distribution given by [59]

$$p(\mathbf{x}|i) = \frac{1}{(2\pi)^{N_F/2}|\Sigma_i|^{1/2}} exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_i)\Sigma^{-1}{}_i(\mathbf{x} - \mu_i)^T\right\} \ , \tag{2.10}$$

where the superscripted T denotes transpose. A sample from the input *testing* set is projected via MDA and class likelihood values are assigned using Bayesian decision theory, where the conditional posterior probability is given by

$$p(i|\mathbf{x}) = \frac{p(\mathbf{x}|i)p(i)}{p(\mathbf{x})} \, . \tag{2.11}$$

The final class estimate of a test observation is assigned to the class i with highest probability, or

$$p(i|\mathbf{x}) > p(j|\mathbf{x}), \ \forall i \neq j \, . \tag{2.12}$$

The class yielding highest probability is assigned as the class estimate for the unknown testing input. As large dimensional distributions can have very small probabilities [59], log-likelihood is used here. Given a *known* distribution, ML provides optimal classification performance [125].

2.5.1.2 Naive Bayes

If N training observations are required to accurately estimate the probability $p(\mathbf{x}|i)$ Bayes classification for one variable, and if $\mathbf{x}$ is a vector made up of l variables such that $\mathbf{x} = [x_1, x_2, \ldots, x_l]^T$, then at least N^l data points would be required to fully estimate the multivariate distribution of the training data. To reduce computational complexity, if the l dimensions are assumed to be statistically independent, the following holds [125]

$$p(\mathbf{x}|i) = \prod_{j=1}^{l} p(x_j|i), \ i = 1, 2, \ldots, c \, . \tag{2.13}$$

This equation states that the multidimensional distribution $p(\mathbf{x}|i)$ can be represented as l one-dimensional distributions. Assuming mutual independence of the variables, their joint probability can be determined simply by multiplying the individual l probabilities. This can be used in Bayes classification per (2.11) and is known as the Naive Bayes (NB) classifier.

2.5.2 Non-Parametric Classifiers

Random Forest (RndF) is the main non-parametric classifier used in this research. However, others are also referenced and compared to RndF and their descriptions are included in this section.

2.5.2.1 Support Vector Machines

Support Vector Machine (SVM) is a classifier that attempts to find the optimal split between two classes, as shown in Figure 2.8 where a two variable (i.e. in $\mathbb{R}^2$, two class problem is presented. The SVM provides a linear discriminant to separate the two classes with the maximum possible margin. Hence, SVMs are often referred to as *Max Margin Classifiers*. *Generalization* is the property of a classifier to correctly classify a given

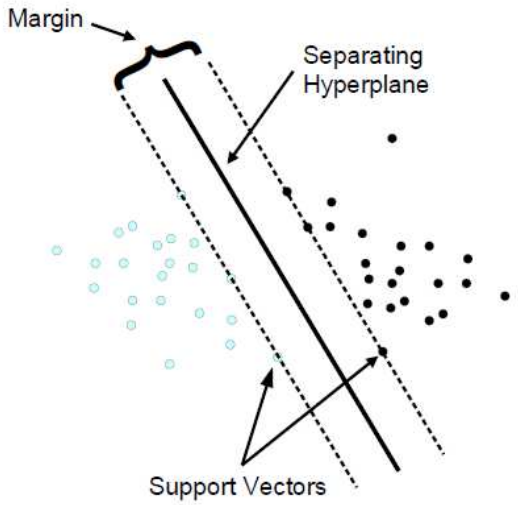
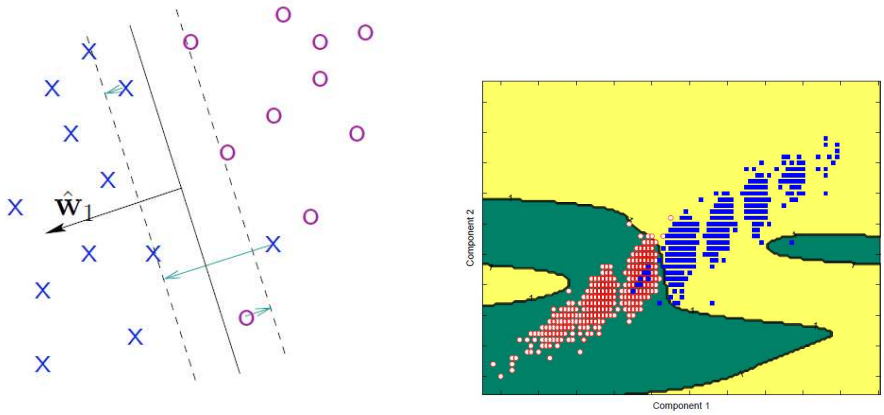


Figure 2.8: Support Vector Machine in 2-variable space for a 2-class separable problem [77]. Bold line represents the optimal separating hyperplane, and corresponding parallel dotted lines represent support vectors.

unknown measurement that is not part of the training set. Since SVMs attempt to maximize the separable space between classes, they have very good generalization properties. For an

l-dimensional problem, the SVM generates an $l-1$ dimensional hyperplane to separate the two classes.

For the case when the two classes are not separable, the SVM still generates a hyperplane that separates the two classes as much as possible while minimizing the number of observations that incorrectly lie on the wrong side of the hyperplane as shown in Figure 2.9a.



(a) (b)

Figure 2.9: (a) Support Vector Machine in 2 variable space for a 2 class non-separable problem , and (b)a LS-SVM non-linear boundary decision from 2-component SCA data to differentiate 1 bit (2-classes) non-separable power leakage [55].

An alternate method if classes cannot be separated by a linear hyperplane is to use a kernel method with the aid of one of many kernel functions. This method called the *kernel trick* implicitly maps the original dataset to a higher dimensional space in a computationally efficient manner. If the classes can be separated by SVM in this higher dimensional space, then the decision boundary hyperplane in that higher dimensional space can be mapped back to the original feature space to give a non-linear decision boundary to separate the

classes. In [55], the Least Squares SVM (LS-SVM) was used with a kernel function called the Radial Basis Function (RBF) for non-linear classification by converting the original SCA power leakage data set to higher dimensional space. Figure 2.9b shows the two-variable space and the decision regions defined by the LS-SVM with RBF. Red dots represent a 0 value for the bit under attack in the training set, and blue dots represent a 1 value. The yellow and green regions represent the class regions identified by the classifier based on the training data. Although the kernel method provides a means to develop non-linear decision boundaries, it requires derivation of parameters and does not guarantee separability. SVMs are inherently two-class classifiers but can be extended to multi-class classification by *one-vs-all* or *pair-wise* classification [51].

2.5.2.2 Perceptron and Neural Networks

A linear classifier is defined in [125] as an algorithm that separates classes c_1 and c_2 in l-dimensional space by a hyperplane defined as $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$. Figure 2.10 shows one of the simplest non-linear classification problems representing 2-bit xor data. With this data, it is not possible to introduce a linear classifier that can achieve less than a 50% miss rate. In such circumstances, a non-linear classifier is needed. A logical approach to this problem is to divide the space using *two* linear separators as shown in Figure 2.11a. The associated

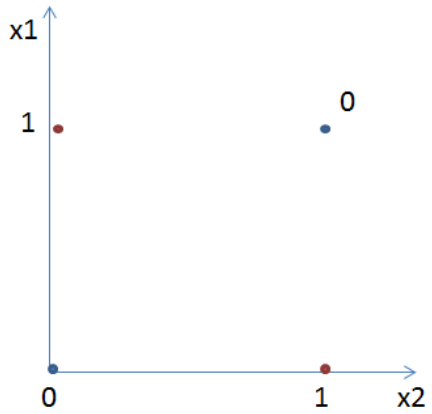


Figure 2.10: Non-linear xor classification problem.

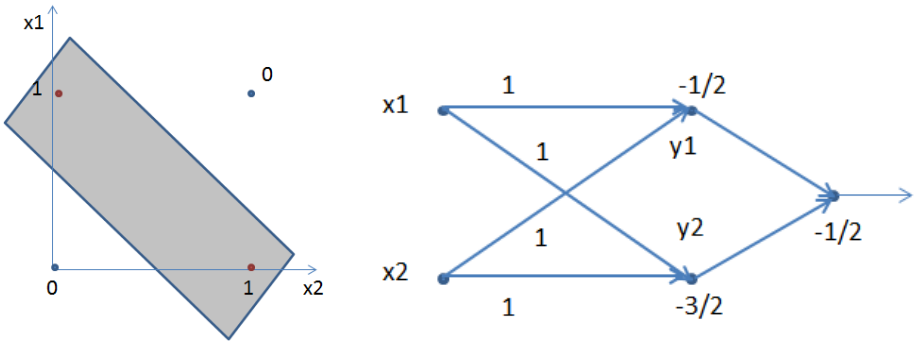
system of equations to these discriminants is

$$y_1 = x_1 + x_2 - 1/2 = 0 \tag{2.14}$$

$$y_2 = x_1 + x_2 - 3/2 = 0$$

$$g(\mathbf{y}) = y_1 + y_2 - 1/2 = 0\ .$$

These equations can be graphically represented as in Figure 2.11b as a two-layer



(a) (b)

Figure 2.11: (a) xor problem solved with two discriminants, and (b) the associated perceptron to this solution

perceptron. This type of structure is also referred to as a *feedfoward neural network* as the data process goes in the forward direction only from the input to the output [125]. The first column of nodes represented by x_1 and x_2 are called the input layer. Nodes that are not in the input layer are referred to as *neurons*. Thus, this layer can be as large as the dimensions of the problem. The middle layer is also called the *hidden layer* where the data is mapped to an intermediate decision space, and then finally an output decision is reached at the output node. For a multi-class problem, the perceptron can have more than one output nodes, allowing them to easily account for multi-class datasets. In addition, for more complex decision spaces, more than one hidden layer may be required. Hidden layers attempt to account for interactions between variables or groups of variables. One disadvantage

of neural networks is their high computational time which grows exponentially with the number of variables.

2.5.2.3 GRLVQI

Generalized Relevance Learning Vector Quantization-Improved (GRLVQI) is a neural learning algorithm that iteratively adjusts prototype vectors to define class boundaries [63]. Initially, the classifier LVQ was developed as a system of K prototype vectors arbitrarily chosen for the data set. Weight vectors are assigned to a first layer of neurons with each neuron representing one class. The weight vector of a class neuron is iteratively updated (learned) to reduce the training data distance of that class in a manner similar to supervised K-Nearest Neighbors (KNN) processing. LVQ was generalized to GRLVQ in [114] using gradient descent to minimize drifting of the prototype vectors from their optimal locations. Hammer modified the algorithm to incorporate variable relevance ranking [44] which was key to successful implementation of feature Dimensional Reduction Analysis (DRA) work in [106]. GRLVQI was successfully used for RF-DNA fingerprinting work in [47] involving identification and verification of WiFi and WiMAX devices. In addition to vector class labels, GRLVQI produces a distance metric that provides a measure of confidence in the class estimate. Device ID verification performance was improved in [106] by modifying the confidence measure to account for prototype vector angle as well as distance.

2.5.2.4 K-Nearest Neighbors

KNN is a very simple yet powerful non-parametric classifier that is frequently used and scales very well to multi-class problems. The basic KNN algorithm is defined as follows:

1. For an unknown vector $\mathbf{x}$, find the k nearest training data observations in the M-dimensional space, regardless of their class.

2. The class with the maximum number of observations out of k is given as the class of $\mathbf{x}$.

This algorithm works well for data that is separable within the variable space. It has been shown theoretically that as the number of training observations grows without bound, the error probability becomes bounded [37]. For k-nearest neighbors, as k$\rightarrow \infty$, the classification error approaches the optimal error and would do well for large training sets. However, the computational complexity and memory burden becomes great for very large number of observations as a large search space needs to be navigated to find the KNN. In addition, this model requires the full training set to asses the test set, which can be cumbersome for algorithm deployment [125].

2.5.2.5 Decision Trees

Binary decision trees are low-bias classifiers capable of accepting high-dimensional data. The tree is *grown* by asking several two-response questions, each question being called a *node*. The node is split into left and right *child* nodes each splitting the training data into smaller groups. The particular response to the question brings the algorithm down the next *branch* and the process is repeated until a class is determined. The recursive algorithm for growing a tree is shown in Algorithm 3. For each variable, each possible

Algorithm 3 Decision Tree Node Splitting Recursive Algorithm

```
SplitNode (Data, Impurity)
if Impurity == minimum Impurity then
   return
else
   for each variable x_i do
      for each possible value t do
         x_i.leftChild = Data>t
         x_i.rightChild = Data≤t
         Calculate x_i.leftChild impurity i_L
         Calculate x_i.rightChild impurity i_R
         Calculate impurity reduction for (x_i, t)
      end for
   end for
   BestNode = max( Impurity reduction for all variables and all thresholds)
   SplitNode( BestNode.leftChild, BestNode.i_L )
   SplitNode( BestNode.rightChild, BestNode.i_R )
end if
```

observable value is examined as a threshold. Data is split into two groups called the $leftChild$ and $rightChild$ based on each threshold examined. After the split, a metric called *impurity* is calculated for the split in which the impurity of the classification for each of the resulting children (leftChild and rightChild) is computed. Impurity can be calculated based on entropy reduction or information gain from the split [15], Gini impurity reduction [100] and acAUC [39]. The entropy reduction method is presented below.

Shannon's Entropy Impurity For a c-class problem, the probability of class i at node a is represented as $p_a(i)$ and is efficiently estimated by counting the number of observations (frequency count) for class i in a group of data. Then the impurity at node a can be calculated by finding the Shannon Entropy from information theory in

$$I(a) = -\sum_{i=1}^{c} p_a(i) log_2 p_a(i) \ . \tag{2.15}$$

$I(a)$ takes on the maximum value when all probabilities are equal to $1/c$ and is 0 when all the observations belong to only one class.

Impurity Reduction After choosing a method of calculating impurity of node a, the decrease in impurity after the split is calculated. Suppose after the split, N_l is the number of data points in the left child node, N_r the number in the right child node, and N_a the number in the parent node a. Then the impurity reduction at node a for threshold t is

$$\Delta I_t(a) = I_t(a) - \frac{N_l}{N_a} I_t(leftChild) - \frac{N_r}{N_a} I_t(rightChild) \ , \tag{2.16}$$

where $I_t(leftChild)$ and $I_t(rightChild)$ are the impurities of the left and right child nodes respectively for a given threshold t. To grow a node, this process is repeated for m variables. The variable and threshold combination (v^*, t^*) that reduces the impurity the most after the split from parent to child nodes is chosen to be included in the model. This process is described by

$$(t^*, v^*) = \arg\max_{t,v} \Delta I_{t,v}(a) \ . \tag{2.17}$$

Stop Splitting Rule The tree will stop growing when a pre-defined minimum impurity is met by a node. This terminal node is called a *leaf*. The majority class in that node then becomes the leaf class label. Choosing a smaller minimum impurity leads to larger trees as the nodes keep splitting until the threshold is met. There is a temptation to grow the tree until the leaf node is pure, i.e. until there is only one class in the final leaf. However, this often leads to *overtraining* where the error in the training set is reduced to zero, but the error in an unknown separate test set grows. Allowing a certain level of impurity in the leaf nodes gives greater generalization power to the classifier in order to classify unknown data, separate from the training set, more accurately [15, 125].

Decision trees benefit from being able to divide the decision space into irregular shape subsets. Figure 2.12 shows the Fisher Iris dataset, which Fisher used to develop the Linear Discriminant Model [40]. Class labels are the names of iris flowers used in the dataset. Figure 2.12a shows the two-variable space partitioned with the ML classifier for an estimated multivariate Gaussian distribution. Similarly, Figure 2.12b shows how the decision tree would partition the same space. Figure 2.12c shows the graphical view of the decision tree. By using a number of nodes, the tree algorithm is able to partition the space to a greater level of refinement than ML. However, outliers can negatively affect the tree performance, especially if the tree is overtrained, resulting in overly partitioned data.

Decision trees are capable of handling multi-class problems in high-dimensional datasets and have low bias in their estimations. A disadvantage of decision trees is that they are also high-variance classifiers [125]. Small changes in the data can lead to different ways of growing trees, which can then lead to a greater variation in their predictions. To overcome this, a number of decision trees can be combined through classifier ensemble methods such as *Boosting* and *Random Forests*.

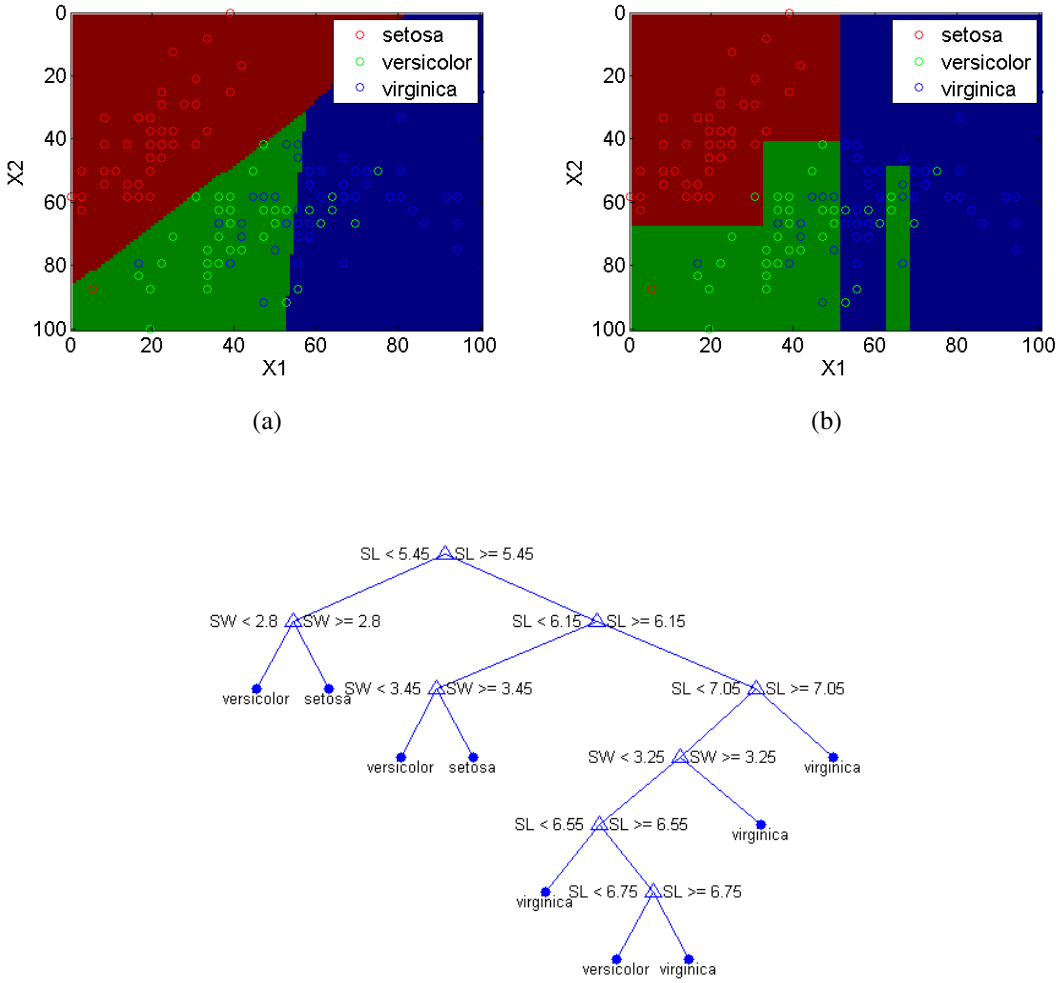


Figure 2.12: Classification regions for Fisher Iris Data with linear discriminants (a) and for a binary decision tree (b). Colored regions represent the class decision for the variable space made by the respective classifier. Colored dots represent the training data used to train the classifier. The binary decision tree structure is shown in (c).

2.5.2.6 Boosting Approach to Combining Classifiers

Boosting attempts to evolve a weak learning algorithm, termed a weak *learner*, into a strong one with good error performance [125]. For a boosting algorithm, a sequence

of classifiers is iteratively designed using a *base* weak learner. In each iteration of classification, a subset of the training data is supplied with a distribution that assigns the most misclassified data a higher weight. Thus, observations that were misclassified in the previous iteration are more likely to be used to develop the classifier in the next iteration. For an example, the AdaBoost algorithm is described here [133]. Suppose there is a two-class problem such that for N observations, the corresponding class labels are $y_i \in \{-1, 1\}$, for $i \in \{1, 2, \ldots, N\}$. Let $T^{(j)} : X \rightarrow \{-1, 1\}$ be the classification assigned by the weak learner $T^{(j)}$ to the input variable x_i and initial distribution of the training samples is uniform. The AdaBoost algorithm is given by Algorithm 4 [133]. $\mathbb{I}$ is the identity function, which is 1

Algorithm 4 Two-Class AdaBoost algorithm for growing a strong learner

Initialize N observation weights to $w_i = 1/N$.

for j=1,...,J **do**

(a) Fit a classifier $T^{(j)}$ to the training data using weights w_i.

(b) Compute $err^{(j)} = \sum_{i=1}^{n} w_i \cdot \mathbb{I}\left(c_i \neq T^{(j)}(x_i)\right) / \sum_{i=1}^{n} w_i$.

(c) Compute $\alpha^{(j)} = log\frac{1-err^{(j)}}{err^{(j)}}$.

(d) Set $w_i \leftarrow w_i \cdot exp\left(\alpha^{(j)} \cdot \mathbb{I}\left(c_i \neq T^{(j)}(x_i)\right)\right)$, $i = 1, 2, \ldots, N$.

(e) Re-normalize w_i.

end for

Output: $C(x) = \underset{k}{\text{argmax}} \sum_{j=1}^{J} \alpha^{(j)} \cdot \mathbb{I}\left(T^{(j)}(x) = k\right)$.

when its conditions are true and 0 otherwise. $\alpha^{(j)}$ is chosen through a cost function that penalizes misclassifications much more than correct ones. Thus, harder training traces are given more weight than easier ones, leading to higher probability of them being chosen in the next iteration. After J boosting iterations, $C(x)$ is the final strong learner.

A main advantage of boosting has been its immunity to overfitting [17, 125]. Even when the classification error on the training set goes to 0, the error on the test set

continues to decrease until some asymptotic value is reached. Breiman [17] believed that classification ensembles perform better when individual classifiers are uncorrelated. He stated that Adaboost might work well because at each step the algorithm attempts to decouple the next classifier from the current one. Amit et al [6] show that Adaboost attempts to keep covariance between classifiers low.

Adaboost is designed specifically for a two-class problem. The multi-class Adaboost algorithm named *SAMME* was successfully introduced in [133] by adding a $log(K-1)$ term to step (b) for a K-class problem.

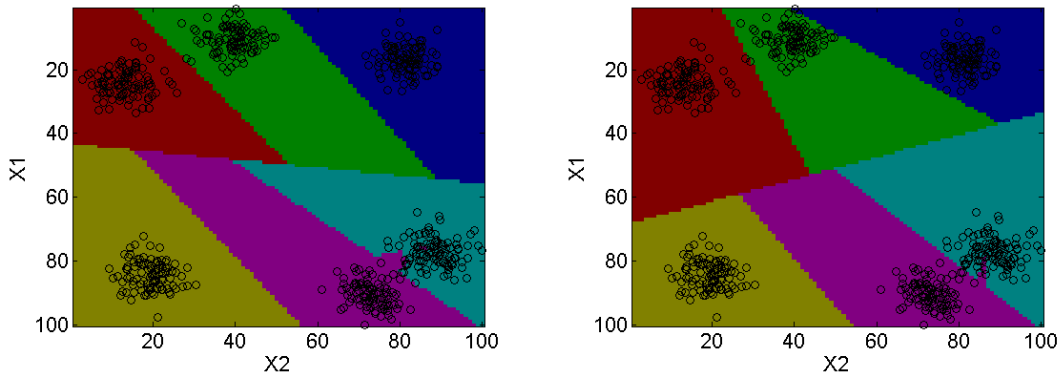
Boosting algorithms like Adaboost are sequential algorithms and require significant computational time as ensemble growth cannot be parallelized. In [128, 131], an alternate method to Adaboost is proposed where multiple base classifiers are trained in parallel on different features. At each iteration of the boosting algorithm, the classifiers are combined. This has a close relationship to another ensemble algorithm called *Random Forest*.

2.5.2.7 Random Forest

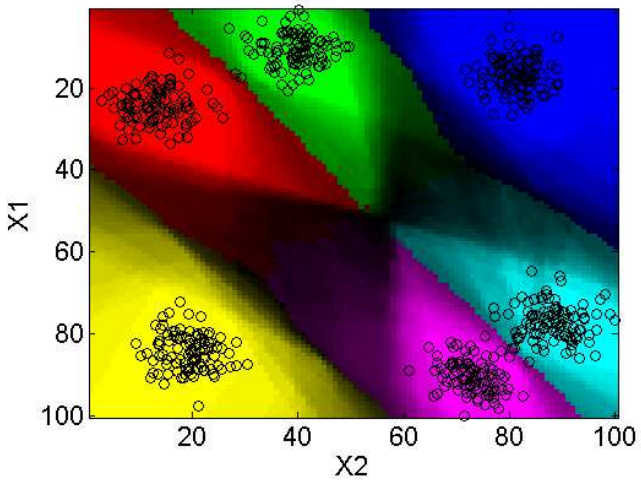
Decision trees provide an unbiased classifier at the cost of high variance, i.e. slight changes in the input training data can cause the tree to be grown differently. In addition, trees are susceptible to noise in the test data, leading to misclassification. A widely used method to minimize these effects is the Random Forest classifier. Ensemble classifier performance primarily depends on two factors: strength of the individual weak classifiers s, and the correlation between the classifiers ρ [17].

To reduce correlation in the Random Forest, trees are grown using a random sampling of variables $m<M$ at each node, where M is the total number of variables. The sampling is performed with replacement, so it is possible that a variable is reused for various nodes. As in decision trees, entropy reduction or information gain from the split [15] to decide how to split the node. Additionally, each tree is grown on a unique random sampling of observations chosen with replacement and termed *in-bag* observations. Thus, each tree

gets a unique "view" of the decision space based on the in-bag variables used to grow it, as shown in Figures 2.13a and 2.13b.



(a) (b)



(c)

Figure 2.13: Boundary decisions by three different random decision trees on the same training data in (a) and (b), and (c) the boundary decisions of a forest of 50 trees. Shown is the entropy from the classification, with higher entropy decisons in darker shaded regions.

To reduce bias and between-tree correlation, trees are grown to full length, i.e., they are grown until each terminal leaf node is pure. In the test phase, an observation is classified

by each trained tree to reach a classification decision. Final class decision of the forest is reached by majority vote among the ensemble. As Random Forest determines a test class by voting among the trees of the forest, the probability of each class can be calculated by $p(c_i|\mathbf{x}) = v_{c_i}/N_t$ where v_{c_i} are the number of votes for class c_i and N_t are the total number of trees in the forest (i.e. total number of votes). Figure 2.13c shows the entropy of the decisions of a 50 tree forest on a two-variable space, with darker shaded regions indicating higher entropy (regions where multiple trees disagree in their class estimates). In [88], Random Forest was found to provide good posterior probability results when compared to SVM, Logistic Regression, KNN, Artificial Neural Networks and Naive Bayes classifiers. In [17], Adaboost was compared with Random Forest and was found to have comparable performance. Also, the Random Forest algorithm produced a 3% test error rate which was very close to the optimal Bayes error rate of 1% [17].

Random Forest enjoys the inherent multi-class capability of decision trees. It can effectively handle high dimensional data sets such as those of SCA data without needing to transform the data space. The classifier also includes a built-in variable importance metric. For each node of each tree in the Random Forest, the reduction in entropy from the splitting of parent to children over a variable is recorded. The entropy reduction for each variable x_i is averaged over all nodes and all trees to give an entropy based variable importance metric for each variable. The Random Forest Entropy Importance method will be abbreviated as *RFEI* from here on. As with AdaBoost, Random Forest also shows immunity to overtraining. Figure 2.14 shows that training correct classification rate rises to 100% with as few as N_t =12 trees, because trees are grown to full length. However, the classifier is not overfitted as more trees are added and test correct classification rate continues to rise until a steady state is reached after N_t =600 trees.

Random Forest has gained popularity due to its non-parametric and non-linear properties, high performance in high dimensional data sets and ease of implementation. It's

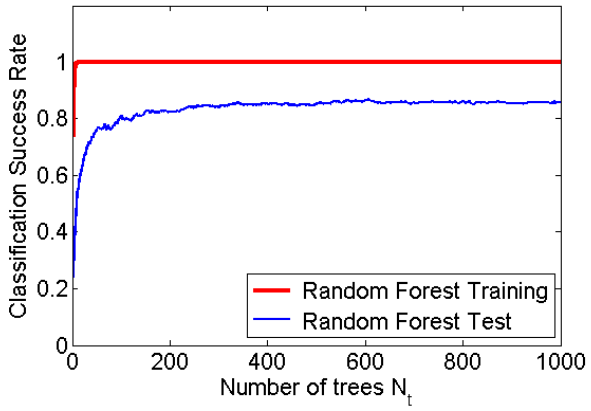


Figure 2.14: Training and test errors for Random Forest as the number of trees N_t increases.

application can be found in classifications of genes [31], spectral data [74], geographical landmarks [85] and agriculture [89] to name a few. In empirical studies, Random Forest has outperformed SVM, KNN and Boosted Decision Trees in large dimensional data sets [85, 89, 109]. In [88], Random Forest was found to provide good posterior probability results when compared to SVM, Logistic Regression, KNN, Artificial Neural Networks and Naive Bayes classifiers.

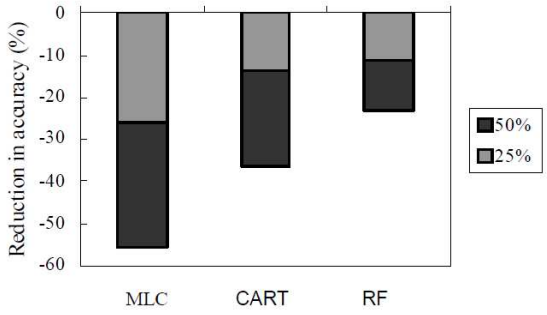
Although the true convergence theory behind Random Forest remains an area of active research [13, 70], a few conclusions may be derived directly from its implementation. First, each tree is grown on a different data set, thereby reducing the influence of data outliers. Next, a random sampling of variables is analyzed at each node of each tree in the forest. From each sampling, the variable most useful in classification is used for partitioning the data further. This variable filtering reduces the influence of non-data dependent variables and ensures that more useful variables are used in the classification, as mathematically proven in [12]. The Random Forest determines decision regions strictly from the data alone without any assumption of an underlying distribution. In addition, as the decision space is split at each node, non-linear decision regions can be determined.

2.5.3 Comparison of Classifiers

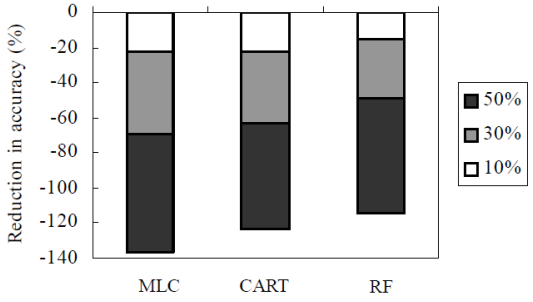
An empirical comparison of the classifiers listed here was conducted in [22] which included boosted trees, SVMs, neural nets, Random Forest and KNN. Results showed that for low-dimensional data up to 4000 dimensions, boosted decision trees performed the best with Random Forest performing next best with respect to correct classification. Above 4000 dimensions, the Random Forest performed the best. In [89], SVMs narrowly outperformed Random Forest (less than 1% better) and neural networks for satellite imagery of crop data when 3 or more images were used for classification. All three easily outperform the Naive Bayes Maximum Likelihood method. In [85], the Random Forest is compared to single decision tree and maximum likelihood geographical data. It is not known how the probability distribution of the training data is estimated for the maximum likelihood algorithm or what assumptions on data probability distribution type were made. However the Random Forest performed the best here, with overall accuracy of 90.96%, while single decision tree accuracy was 89.16% and maximum likelihood was 83.58%. Interestingly, when the training set size was reduced by 25%, the Random Forest performance only decreased by 11%, compared to 13.8% with single decision tree and 26% with maximum likelihood as, shown in Figure 2.15a. In addition, when noise was added to the training set, it was found that all three classifiers were highly affected, but that the Random Forest degraded the slowest as shown in Figure 2.15b.

2.5.4 Machine Learning and SCA

Few efforts to date have focused on using classifiers other than Maximum Likelihood with Bayes rule in template attacks. In [55], the Least Squares Support Vector Machine (LSSVM) is compared to standard Template Attacks using maximum likelihood with multivariate Gaussian distribution for the estimates of $p(\mathbf{x}|i)$, where i is the class defined by the intermediate value in the AES algorithm. Only two experiments were conducted for two-class problems that revealed very little information. The first attempted to classify



(a)



(b)

Figure 2.15: Comparison of ML (MLC), single decision tree (CART) and RndF (RF) from [85] showing (a) decreasing in performance when the training set size was reduced, and (b) performance degradation when noise was added to the training set.

those traces where the 4^{th} least significant bit was either a 0 or 1. The second experiment classified traces based on even or odd Hamming Weights of the intermediate value *i*. Thus, both of these experiments provided limited data and did not reveal the actual key value. The LSSVM was found to perform either in-line or below Template Attacks.

The second, more substantial research effort was conducted in [66]. Here, Random Forests, Template Attacks, SVM and Self Organizing Maps were compared using SCA data on a FPGA device. The encryption was 3DES which uses 3 64-bit keys. The data was classified *bit-wise*, where a different classifier was trained for each bit of the key. Thus,

for a 128-bit AES key, 128 separate classifiers were grown for classification. Results show that although some bits had as high as 96% correct classification rate, others were as low as 50% and many were somewhere in between. In addition, in some scenarios authors rounded lower errors up to 50% as they claim that 50% is as good as random guessing. This inflates their performance results as the error rate of lower performing classifiers are not truly depicted. Overall, Random Forest with PCA performed the best, and authors claim an improvement in key recovery over Template Attacks from 5.80% to 15.33%. However, template attack misclassification rates were not provided for comparison. In addition, the test data success was not measured by the commonly used classification rate which is the ratio of the correctly classified observations over the total number of observations. Instead, authors use an *enhanced brute force* attack, where if the key is incorrect, the most difficult to predict bit value is flipped. This is repeated until the key is recovered. If it cannot be recovered, it is considered a miss. Thus, multiple attempts are allowed until a miss is classified. On average, Template Attacks with *enhanced brute force* required 11 attempts, while Random Forest with PCA required 21 attempts.

Finally, SVM was used most recently in [51] for HW-based pair-wise classification. Models were generated in a one-versus-one strategy with each HW pair-wise combination being modeled separately. For L HWs, $(L-1)L/2$ models were trained and combined to jointly provide a multi-class classifier. Success was achieved with lower guessing entropy [119] using smaller training sets than Template Attack under moderate to high noise conditions.

2.6 RF-DNA Fingerprinting

Distributed networks such as WiFi, ZigBee and WiMax allow for network creation and sustainment with little human interaction required. These networks are designed to allow their hardware to be remotely located, leaving vulnerabilities for eavesdropping and cloning attacks. With access to a network key, an attacker can enter these networks with a rogue

device and gain access to transmitted data. RF-fingerprinting provides a means to exploit physical layer characteristics of devices to enhance network security. As the physical characteristics between devices are randomly generated through the manufacturing process, they are very difficult to reverse engineer, thus, providing a useful hardware authentication tool. RF-DNA fingerprinting is the subset of RF-fingerprinting that is used in this research.

Figure 2.16 shows the signal collection and RF-DNA fingerprint generation methodology [129]. A statistical RF-DNA fingerprint (**F**) is derived from the signals instantaneous amplitude (a), phase (ϕ) and/or frequency (f) responses as described in [102]. The corresponding response sequences $a[n]$, $\phi[n]$ and $f[n]$ are generated from the N_S complex I-Q signal samples $s[n] = s_I[n] + js_Q[n]$ within the region of interest by,

$$a[n] = \sqrt{s_I^2[n] + s_Q^2[n]}\,, \tag{2.18}$$

$$\phi[n] = tan^{-1}\left[\frac{s_Q[n]}{s_I[n]}\right] \text{ for } s_I[n] \neq 0\,, \tag{2.19}$$

$$f[n] = \frac{1}{2\pi}\left[\frac{d\phi[n]}{dt}\right]\,, \tag{2.20}$$

where $n = 1, 2, \ldots, N_S$. These sequences are centered (zero mean) and normalized (divide by maximum value) prior to calculating statistical RF-DNA features of standard deviation (σ), variance (σ^2), skewness (γ) and/or kurtosis (κ) within selected signal region(s) of interest. The regional fingerprint markers for the signal are generated by 1) dividing each response sequence into N_R contiguous equal length sub-sequences, 2) calculating N_S statistical metrics for each sub-sequence, plus the entire region as a whole for $N_R + 1$ total regions, and 3) arranging the metrics in a vector of the form

$$\mathbb{F}_{R_i} = [\sigma^2_{R_i}\ \gamma_{R_i}\ \kappa_{R_i}]_{1\times 3}, \tag{2.21}$$

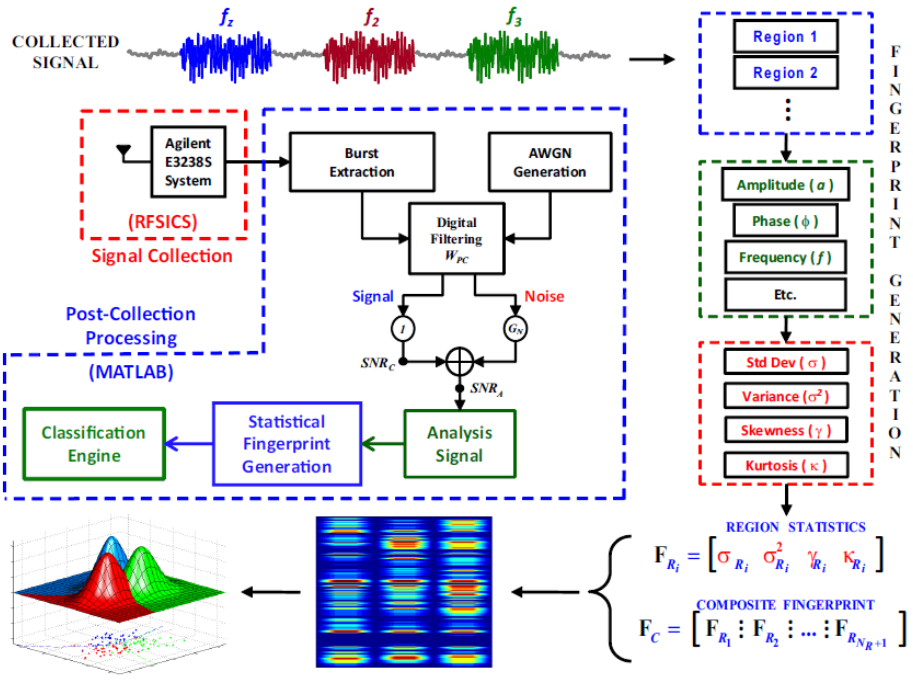


Figure 2.16: RF-DNA Fingerprint Generation Process [129]

where $i = 1, 2, \ldots, N_R + 1$.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x[n]-\mu)^2}\,, \tag{2.22}$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x[n]-\mu)^2\,, \tag{2.23}$$

$$\gamma = \frac{1}{N\sigma^3}\sum_{n=1}^{N}(x[n]-\mu)^3\,, \tag{2.24}$$

$$\kappa = \frac{1}{N\sigma^4}\sum_{n=1}^{N}(x[n]-\mu)^4\,, \tag{2.25}$$

where $x[n]$ is the n^{th} feature vector element and N is the total number of samples in each subsequence used to calculate the statistic.

2.6.1 Machine Learning and RF-Fingerprinting

RF-DNA fingerprinting has shown success over a number of RF devices with multiple classifiers, including using MDA/ML with Cognitive Radio networks [47], WiMAX [107],

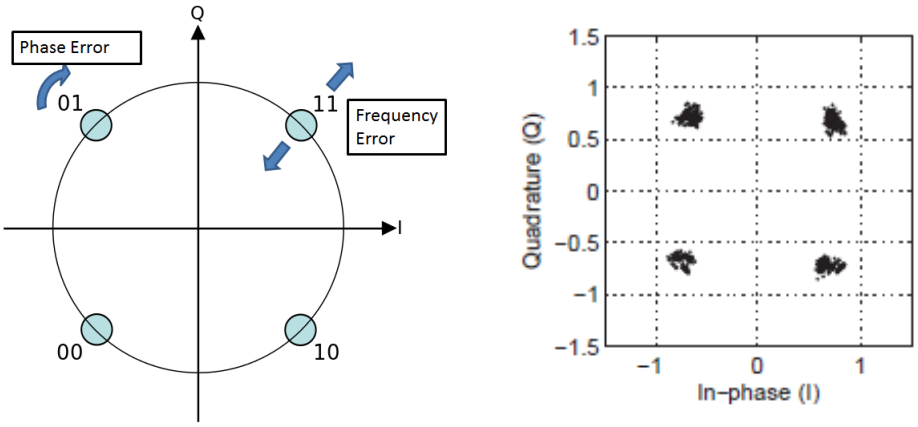
and 802.11a WiFi [48]. GRLVQI was successfully used in [106] to improve classification performance on ZigBee devices using RF emissions collected in multiple environments. In [94], RF-DNA fingerprints for this dataset were shown to be highly non-Gaussian, and RndF was shown to improve performance over the parametric MDA/ML classifier.

RF-DNA fingerprinting is a subset of a larger group of research called RF-fingerprinting which includes various methods to exploit signal differences to classify RF devices. In [29], the transient part of the 802.15.4 signal from CC2420 radio transceivers were used with Mahalanobis matching and MDA to achieve an Equal Error Rate of 0.24%. In [30], 100 IEEE 802.11 Network Interface Cards were tested with the KNN classifier and achieved a success rate of greater than 99%. In [132], SVM with the Radial Basis Function (RBF) was used with IEEE 802.16-2009 devices to detect network intrusion attacks, yielding a False Positive Rate (FPR)=0.67% and a False Negative Rate (FNR)=2.15%. Thus, a variety of classifiers have been successfully applied in the field of RF-fingerprinting to enhance security protection for a number of RF devices.

2.7 802.15.4 ZigBee

ZigBee devices based on the IEEE 802.15.4 standard [10] have gained popularity in a variety of applications as devices of choice for low-cost, low-power and low-complexity communication applications [60]. Per the IEEE standard, ZigBee devices operate as either Full Function Devices (FFD) capable of functioning as a network coordinator, or as Reduced Function Devices (RFD) capable of communicating only with an FFD. ZigBee networks can operate as decentralized mesh networks [68] such as Star, Peer-to-Peer or Cluster Tree Topology, allowing new devices to be discovered and incorporated easily into an existing network. ZigBee devices represent a 100× lower-power alternative to Bluetooth communications for short message burst communications [9, 60]. The low-cost, low-power attributes of ZigBee have greatly increased their popularity and they have been widely

adopted for monitoring and control in industrial and building [38], healthcare [58], and security system applications [126].



(a) (b)

Figure 2.17: (a) Theoretical QPSK constellation and the effect of frequency and phase error, and (b) constellation plot in [30] showing variation in QPSK symbol location.

Per the IEEE 802.14.5 MAC Physical Layer Specifications manual [56], ZigBee devices can support several modulation methods for RF communications, including Offset Quadrature Phase Shift Keying (O-QPSK), Orthogonal Frequency-Division Multiplexing (OFDM) and Frequency Shift Keying (FSK). For this work we will examine O-QPSK as it is used in the Atmel RZUSBstick [8] and the TI CC2420 [124], which are the ZigBee devices under test. QPSK, which is the basis for O-QPSK, is a 4 level (4-ary) phase shift keying modulation where $N_S = 4$ symbols can be expressed as

$$s_i(t) = \sqrt{\frac{2E}{T_s}} cos\left(\omega_0 t - \frac{(2i-1)\pi}{N_S}\right) \qquad 0 \le t \le T_s, \quad i = 1, \ldots, N_S \ , \tag{2.26}$$

where E is the received energy over symbol duration T_s, and ω_0 is the carrier frequency [117]. Ideally, these four symbols would be placed at the 45, 135, 225 and 315 degree locations on the circle shown in Figure 2.17a. However, slight manufacturing

defects in hardware can cause errors in the amplitude, phase and frequency of the emitted signal. These errors are small enough to assure that the specified bit-error rate can still be achieved, and are therefore deemed by the manufacturer to be tolerable. Figure 2.17b shows the actual constellation points for received QPSK symbols from 802.11 WiFi signals. The variation in amplitude, phase, and frequency that cause this distribution of constellation points can be exploited by methods such as RF-DNA fingerprinting for device classification [93].

2.8 Summary

This research addresses two main areas: SCA with unintentional emissions, and RF-DNA fingerprinting with intentional emissions. SCA on cryptographic devices involves collecting unintended information to determine otherwise unobtainable sensitive information. For the AES algorithm, this information is in the form of the secret key stored in the device memory. By examining the EM, power, timing and other side channels, various methods such as CPA and profiling attacks can be used to determine the key value. Profiling attacks offer the best known performance at finding the correct key with the fewest number of test traces at the cost of requiring a much larger number of training traces. However, once the test device is known, a similar or same model device can be obtained afterward and trained at the attacker's leisure. Once sufficient training has been accomplished, the attacker can collect a smaller set of test traces from the target device to determine the correct key.

In this chapter, two main groups of classifiers were discussed: parametric classifiers and non-parametric classifiers. Parametric classifiers such as MDA/ML assume class parametric distributions and estimating their parameters through the training data. Non-parametric classifiers by contrast do not assume an underlying distribution, but instead estimate class boundaries strictly from the training set observations for a given variable space. Side channel collection of the entire encryption operation, sampled at multiples of

the Nyquist sampling rate, consist of observations with large numbers of samples. This results in very high dimenisonal data. A comparison of various classifiers shows that the Random Forest provides the best combination of computational efficiency and high performance datasets [22], which characterizes SCA data.

For intentional RF emissions, slight hardware differences in RF devices lead to corresponding variation in the emitted signal's frequency, phase and amplitude. This variation is used by RF-DNA fingerprinting to enhance the security of distributed networks by providing a physical layer authentication method that is nearly impossible to duplicate. It has been shown to be effective over a range of SNRs and for a variety of RF devices using parametric and non-parametric classifiers. ZigBee devices are gaining popularity in a large variety of sensitive applications. RF-DNA fingerprinting can be used to reliably authenticate authorized ZigBee devices to prevent network intrusion from rogue devices [93].

III. Methodology

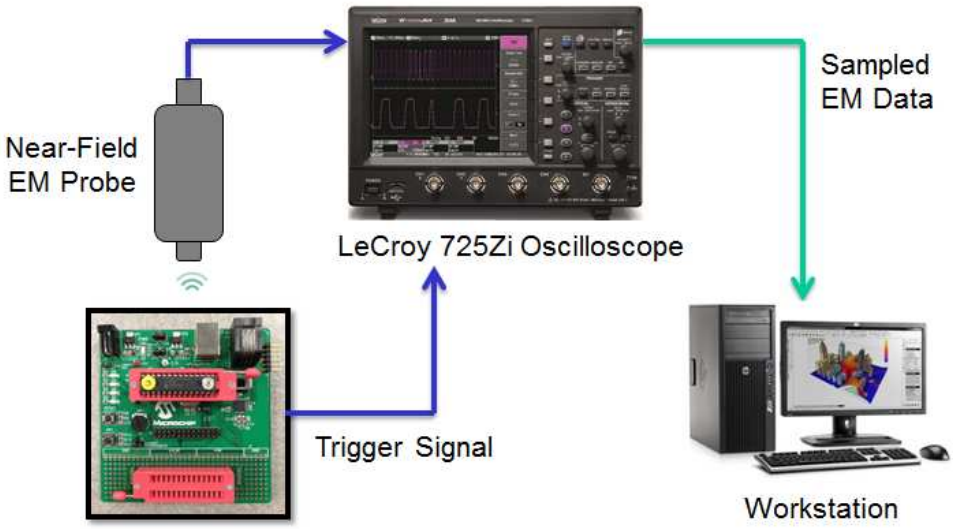
This chapter describes the research methodology used for Random Forest (RndF), Side Channel Analysis (SCA) and RF-Distinct Native Attribute (RF-DNA) fingerprinting. First, data collection methods are specified for SCA *unintentional* EM emissions and ZigBee *intentional* emissions. This is followed by the general methodology used for Linear Regression Attack (LRA), RndF and Multi-Class AdaBoost (MCA) profiling attacks and RF-DNA fingerprinting with RndF. Methodology specific to each research contribution is further explained in Chapters 4, 5, and 6.

3.1 Data Collection Methodology

Data collection methods differed for unintentional and intentional emissions. SCA unintentional emissions were collected from Microchip Peripheral Interface Controller (PIC) microcontrollers during AES encryption in [91, 92]. ZigBee intentional emissions were collected and converted to RF-DNA fingerprints using statistical methods in [93–95].

3.1.1 Unintentional PIC Microcontroller EM Data Collections

Unintended EM emissions were collected from four families of 16-bit PIC microcontrollers from Microchip Technology Inc. Figure 3.1a shows the functional diagram of the SCA collection method. Figure 3.1b shows the hardware setup using the Riscure EM probe station [111]. The EM measurements during the AES-128 encryption operation were the side channel of interest for all experiments. The encryption operation was controlled by first sending the plaintext and key to the PIC from a workstation over a serial connection. The microcontroller then executed the AES encryption and the EM emissions were captured by a Riscure broadband EM probe. Probe responses were sampled by an oscilloscope and the sampled observation sent back to the workstation for storage. The collected traces were temporally aligned using a trigger signal that was sent by the microcontroller before



(a)

(b)

Figure 3.1: (a) Functional diagram of the SCA collection setup with the PIC microcontroller, and (b) hardware setup using a Riscure EM probe station [111].

the start of the encryption, which triggered the oscilloscope to start collection. The different microcontrollers were spatially aligned using a custom designed test jig that can be fixed to the microcontroller board relative to the probe. Post-processing cross-correlation was used to ensure that the collected traces were well temporally aligned.

To minimize noise from external sources during collection, only the AES algorithm is running on the microcontroller. Serial communication with the collection computer is halted during the encryption process. The best location on the chip for EM collection is determined by performing a spatial X-Y scan, where the surface of the chip is divided into a grid of multiple sub-regions and a trace is collected at each region. The region yielding the highest spectral intensity within a DC-30 MHz band is chosen as the best location for data collection. This region is fixed for all devices used in this work, unless otherwise specified.

3.1.2 Intentional ZigBee EM Data Collections

Two datasets were adopted from prior research and used in this research including 1) a the ZigBee *Cross-Environment (XEnv)* dataset [93, 94] and 2) a the ZigBee *Cross-Receiver (XRx)* dataset [95]. Analyzing multiple datasets enables generalization of research results and reveals the usefulness of RndF to improve ZigBee classification and verification for different devices collected by different receivers. Test setup differences between the two datasets provide insight unique to each perspective such as RndF performance on collections from a single environment (XRx dataset) versus multiple environments (XRx dataset). Table 3.1 summarizes the test setup for the two ZigBee datasets. Data collection methods for each dataset are described in the following sections.

3.1.2.1 ZigBee Cross-Environment Data Collection

The ZigBee XEnv dataset was collected in support of research conducted by Ramsey et al. [101] using an Agilent E3238S-based receiver [1]. Data was collected in three environments for all authorized devices, including 1) Line-Of-Sight (*LOS*) where there is a direct unobstructed LOS between the receiver and ZigBee device; 2) an obstructed *Wall*

Table 3.1: Test setup conditions for Cross-Environment (XEnv) and Cross-Receiver (XRx) datasets.

	XEnv	XRx
Number of Authorized Devices	4	3
Number of Rogue Devices	9	3
Number of Training Observations per Authorized Device	1500	300
Number of Test Observations per Authorized Device	1500	300
Receiver	Agilent E3238S	NI PXIe, NI USRP
Collection Environment	LOS Wall Cage	LOS

environment where there is an office wall separating the receiver and the device; and 3) a *Cage* environment where the device and receiver are in an anechoic chamber. A total of 1000 observations were collected for each authorized device per environment, giving 3000 total observations per device. These are divided evenly into test and training sets of 1500 observations each per device.

3.1.2.2 ZigBee Cross-Receiver Data Collection

The ZigBee XRx dataset was originally collected in support of research conducted by Stubbs et al. in [121]. The high-cost receiver used for data collection was the National Instruments (NI) PXIe-1085 system and the low cost receiver was the NI USRP-2921. Six Atmel RZUSBsticks were used as the transmitters. To provide a meaningful comparison, as many parameters as possible were fixed during signal collection. Both receivers collected observations simultaneously and were at an equal distance from the transmitter. Transmit power was set to 1 mW and a common receiver antenna was used for both receivers. A total of 600 transmission preambles were collected from each RZUSBstick along a direct

LOS to the receivers and were divided evenly into training and test sets of 300 observations per device.

3.2 Linear Regression Attack Methodology

This section describes the general methodology used for experiments in [92] involving linear regression attacks using unintentional EM SCA emissions from PIC microcontrollers during AES encryption. The attack used here was first described in [115] and subsequently adopted in related research [34, 50, 92].

Suppose the side channel of interest is the power measurement of the device when performing an encryption operation. An s-bit subkey is represented by $k \in \{0,1\}^s$ and corresponding known p-bit plaintext by $x \in \{0,1\}^p$. The power measurement at time t is then represented by

$$I_t(x,k) = h_t(x,k) + \epsilon_t \,, \tag{3.1}$$

where $I_t(x,k)$ and $h_t(x,k)$, respectively, are the total power measurement and deterministic power measurement for the encryption operation segment that depends on x and k at time t. The non-deterministic part of the power measurement ϵ_t does not depend on x or k. For an estimate $\hat{h}_t(x,k)$ of the deterministic power measurement, the minimum

$$\min_{\hat{h}_t:\{0,1\}^s\times\{0,1\}^p\to R} \sum_{i=1}^{N_1} \left(\left(I_t(x,k) - \hat{h}_t(x,k) \right)^2 \right) \tag{3.2}$$

is attained when $\hat{h}_t{=}h_t$. In other words, the best deterministic power estimate is the one yielding the smallest Sum of Squares Error (SSE). The result of subtracting the measured value $I_t(x,k)$ from the estimate $\hat{h}_t(x,k)$ is the residual $\mathcal{R}_t$ at time t given by

$$\mathcal{R}_t = I_t(x,k) - \hat{h}_t(x,k) \,. \tag{3.3}$$

Now let $\mathcal{F}_{u,t} \subset \mathcal{F}_t := \{\hat{h}_t : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R}\}$ be a subspace of the full function space $\mathcal{F}_t$, and is spanned by u known linear basis functions $g_{i,t}(x,k)$ and $i = 1,\ldots,u$, then the

deterministic power consumption for this subspace can be estimated by

$$\hat{h}_t(x,k) = b_{0t} \cdot 1 + \sum_{i=1}^{u} b_{it} \cdot g_i(x,k) \; . \tag{3.4}$$

The basis functions $g_i(x,k)$ can be constructed with a chosen intermediate value function $\phi(x,k)$ by the function composition $g_i(x,k) = \bar{g}_i(\phi(x,k))$. For example, $g_i(x,k)$ can give the i^{th} bit of the result of the intermediate value function $\phi(x,k)$. How $\phi(x,k)$ and correspondingly $g_i(x,k)$ are chosen is dependent on the attacker's knowledge of the cryptographic algorithm and device architecture. For example, if the attacker knows that the encryption algorithm is AES, he can choose $\phi(x,k)$ to be the SubBytes function for the first byte of the first encryption round, and $\bar{g}_i(x,k)$ extracts the i^{th} bit of that byte value.

A desired property is that the expectation $E(\mathcal{R}_t) = 0$ over x and k, i.e., on average the estimate will be correct. The least squares estimator is used to estimate the model parameters $\mathbf{b}_t = [b_{0,t}, \ldots, b_{u,t}]$ as it meets this condition and provides a solution that minimizes the SSE. As a result, it produces the best estimator $\hat{h}_t^*$ for a given choice of basis functions. In the profiling phase, the least squares estimator is generated by

$$\mathbf{b}_t^* = (A^T A)^{-1} A^T I_t(\mathbf{x}, k) \; , \tag{3.5}$$

where

$$A = \begin{pmatrix} 1 & g_{1,t}(x_1,k) & \ldots & g_{u-1,t}(x_1,k) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & g_{1,t}(x_{N_1},k) & \ldots & g_{u-1,t}(x_{N_1},k) \end{pmatrix} , \tag{3.6}$$

for N_1 observations, and $I_t(\mathbf{x},k)$ is the set of measurements at time t for N_1 observations, each with a different plaintext x_i. Substituting 3.5 in 3.4,

$$\hat{h}_t(x,k) = (A^T A)^{-1} A^T I_t(\mathbf{x},k) \times \mathbf{g}(\mathbf{x},k)^T \; , \tag{3.7}$$

where $g_0(x,k) = 1$ for all realizations of x. Then, $\hat{h}_t^*$ is used to estimate the deterministic power consumption for a second set of N_2 observations and the residuals $\mathcal{R}_t$ are generated

for each time sample t. The expanded derivation of the Least Squares Estimator is provided in Appendix A.

The probability density function $f(\mathcal{R})$ for the residuals $\{\mathcal{R}_1, \ldots, \mathcal{R}_m\}$ is estimated with an m-variable multivariate Normal distribution. For a good estimate, only m time samples with maximum information leakage should be used. Methods in [34, 50, 92, 115] follow the same basic strategy to this point. They differ in their approach to determine the time samples where there is high information leakage, which are subsequently used for determining the correct m dimensions for $\mathcal{R}_t$. Once these time samples are chosen, the regression residuals $\mathcal{R}_t$ from the training data at these m points in time are then used to estimate a covariance matrix C, effectively building a template for the multivariate estimation residual $\mathcal{R}$ for a correct key guess. As there is a zero mean for the residuals given by the $E[\mathcal{R}_t] = 0$ requirement of the least squares method, C alone is needed to build an m-dimensional normal density function,

$$f(\mathcal{R}) = \frac{1}{\sqrt{(2\pi)^m}} \exp\left(-\frac{1}{2}\mathcal{R}^T C^{-1}\mathcal{R}\right) . \tag{3.8}$$

By assuming that the subspace $\mathcal{F}_{u,t}$ contained in $\mathcal{F}_t$ is spanned by the u known linear basis functions, only this single template for $\mathcal{R}_t$ is required.

For the *key extraction phase*, N_3 test measurements are used that have an unknown fixed key and random plaintext. The ML method is applied on the N_3 test traces as follows

$$\alpha(x_1, \ldots, x_{N_3}; k') = \prod_{j=1}^{N_3} \tilde{f}_{k'}(\mathcal{R}) , \tag{3.9}$$

for all sub-keys $k' \in \{0,1\}^s$, $\mathcal{R} = (i(x_j, k) - \tilde{h}^*(x_j, k'))$ for time instants $t_1, \ldots, t_m$. The value of k' that gives the maximum probability is chosen as the sub-key hypothesis. As multivariate densities often produce very low probabilities, especially as m grows, an alternate method is to use the log-likelihood method and sum the exponents in (3.8) by

$$\alpha(x_1, \ldots, x_{N_3}; k') = -\sum_{j=1}^{N_3} (\mathcal{R}^T C^{-1}\mathcal{R}) . \tag{3.10}$$

3.3 RndF Profiling Attack Methodology

The general methodology of experiments presented in [91] for RndF profiling attacks on PIC microcontrollers is described in this section.

3.3.1 Attacking PIC Microcontrollers

Unintentional EM emissions from PIC microcontrollers were sampled by the oscilloscope and transferred to the workstation for post-processing. Each observation was sampled over time with common time samples across observations representing input variables. The collected observations are stored in a matrix format with each observation stored in a row and the columns representing the classification variables. RndF uses these variables for classification in a profiling attack. Byte-wise profiling attacks are considered, where one byte of an AES Intermediate Value (IV) is extracted. The IV of interest is the result of the AES SubBytes function for the first round. A single IV byte yields $2^8 = 256$ possible values, making the byte-wise profiling attack of the first SubBytes output a 256-class classification problem. With a training set, the EM side channel measurements for each of the byte values are *profiled* to create a RndF model for that byte. In the testing phase, the trained model is used to determine the IV byte value of a device with an unknown key. It is assumed that the attacker has access to the plaintext. Once the IV is guessed by the RndF model, the inverse SubBytes function (provided in the AES decryption functions) is used with the known ciphertext to determine the key.

Prior to applying RndF processing on PIC data, the traces are analyzed to gauge normality in the variables. In the following section, the variable analysis method using the Kolmogorov-Smirnoff test is described.

3.3.2 Input Variable Analysis

Classical SCA theory states that side channel leakage for a particular IV is constant and the measurement noise is multivariate Gaussian spanning multiple time samples [24, 72]. Histograms of several variables from the PIC data collections however were found

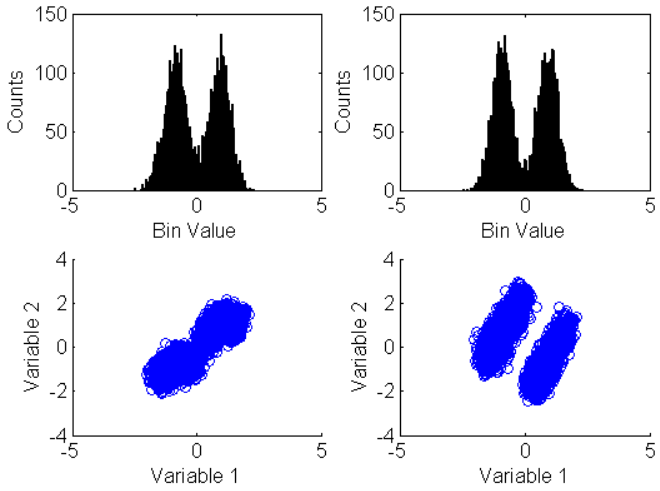


Figure 3.2: 1-D (top) and 2-D (bottom) distributions for an arbitrarily chosen sample set of two PIC data variables. KS-test for normality revealed 32,102 out of 50,000 total variables exhibited similar distribution shape.

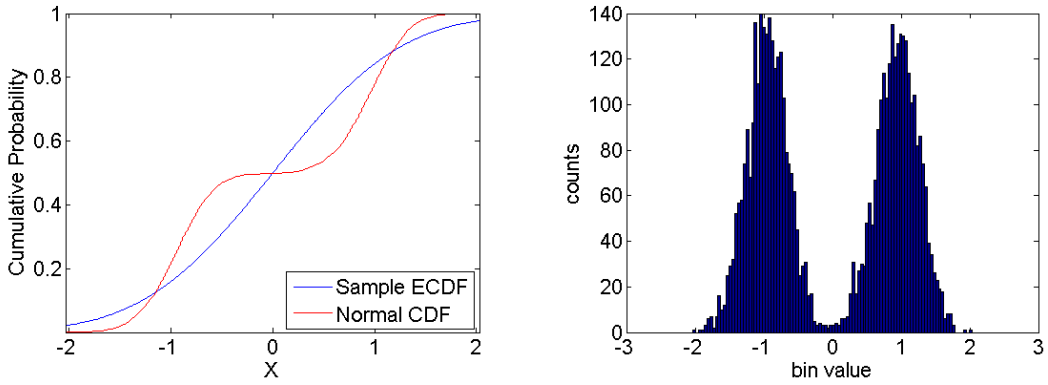
to contain distinctly non-Gaussian shapes as shown in Figure 3.2. The figure also shows 2-variable distribution plots of $p(\mathbf{x})$ exhibiting distinct *multivariate* non-Gaussian shapes. Many more variables were found to show this binomial distribution; only these arbitrarily chosen variables are shown here for clarity.

A one-sample Kolmogorov-Smirnoff test (KS-test) for normality was used to determine how closely variables resembled a standard normal distribution. This test's null hypothesis is that a variable is normally distributed for a given significance level α. The test is performed by calculating the Empirical Cumulative Distribution Function (ECDF) $S_N(X)$ of N samples by first sorting the sample values from lowest to highest in the order $\{X_1, X_2, \ldots, X_N\}$ and then using

$$S_N(X) = n(i)/N, \tag{3.11}$$

where $n(i)$ is the number of points greater than sample value X_i [27]. This is compared to the Cumulative Distribution Function (CDF) $F(X)$ of a normal distribution with mean $\mu = \bar{X}$,

the sample mean, and variance $\sigma = s$, the sample variance [69]. If $D = max(|S_N(X) - F(X)|)$ is larger than the critical value at a given significance level, then the null hypothesis of normality is rejected. Figure 3.3a shows a comparison of a sample ECDF from a PIC microcontroller variable compared to the CDF of a normal distribution. This variable rejected the null hypothesis at α=1e-20. Figure 3.3b shows the associated histogram of the variable. This type of test is referred to as a *one-sample* KS-test. In other research, the two-sample test can be used which compares the ECDF of one variable with the ECDF of another variable.

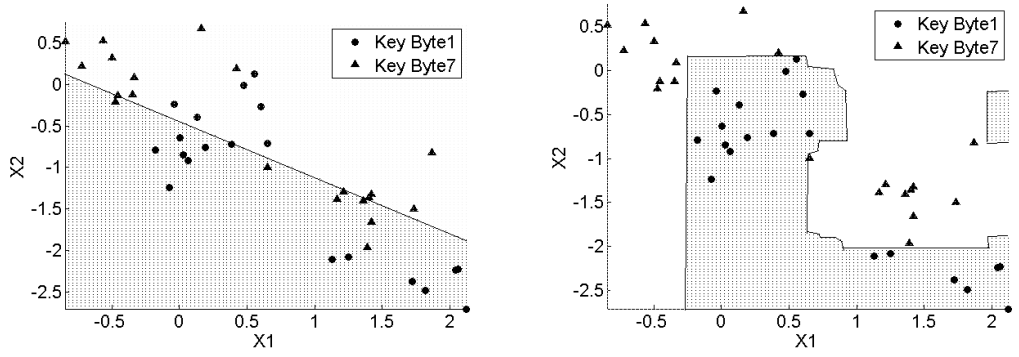


(a) (b)

Figure 3.3: (a) Sample ECDF of a variable from a PIC microcontroller data collection compared to the standard normal CDF. This sample variable rejected the null hypothesis of normality at α =1e-20, and (b) the associated histogram of the variable.

A very low α=1e-20 was chosen to identify those variables that were clearly not normally distributed. It was found that this low α value when used with the KS-test, reliably distinguishes variables housing non-Gaussian distribution such as those seen in Figure 3.2. This method was used in subsequent tests here to distinguish Gaussian and non-Gaussian variables. Analysis revealed that 32,102 of 50,000 total variables rejected the null hypothesis of normality at this α level, and clearly have a non-Gaussian distribution.

Figure 3.4 shows an arbitrarily chosen 2-variable space for key byte values of 1 and 7. Only two variables and two classes are shown for clarity. The figure shows the bimodal nature of variables in Figure 3.2 is also clearly present for $p(\mathbf{x}|Byte_1)$ and $p(\mathbf{x}|Byte_7)$. This behavior has been verified to exist with each of the 256 classes in the byte-wise attack considered here, as well as with other variables. Circle and triangle shaped dots in bold are the locations of the training data used to generate the models. Light and dark shaded regions in the 2-variable space show the classification regions assigned by each respective classifier, i.e., they show how the classifier would partition the 2-variable space given the training set. These non-Gaussian properties were observed for the collections of 40 PIC microcontrollers, demonstrating that non-Gaussian noise can exist in side channel collected data.



(a) (b)

Figure 3.4: Arbitrarily chosen two variable decision space for two IV byte values as determined by (a) Template Attack, and (b) RndF. Light and dark shaded regions represent the decision regions determined by each classifier, and bold shapes represent the training observations used to determine the decision regions.

3.4 RndF RF-DNA Fingerprinting Methodology

RF-DNA fingerprint experiments were conducted with two data sets for ZigBee devices: the ZigBee *XEnv* dataset, and the ZigBee *XRx* dataset. In each case, device *classification* and *verification* experiments were conducted using the methodology for each described below.

3.4.1 RF-DNA Fingerprinting Device Classification

EM emissions from ZigBee devices were collected and RF-DNA instantaneous and statistical features generated for each authorized device. These features were input to different classifiers, including MDA/ML, RndF, MCA, and GRVLQI to develop classification models for authorized devices. These models were then tested with a separate reserved test set of observations. The correct classification rate $\%C$ was used as the metric to determine classification performance. White Gaussian Noise (WGN) was generated, filtered and added to collected observations to achieve various Signal to Noise Ratio (SNR) levels. Data at various SNR was used in the model building and classification process to gauge classifier performance under various noisy environments. An arbitrarily chosen $\%C = 90\%$ benchmark and corresponding SNR assess classifier performance. This is useful in comparing classifiers, as it reveals how much or little noise a classifier can tolerate to maintain this level of correct classification for authorized devices.

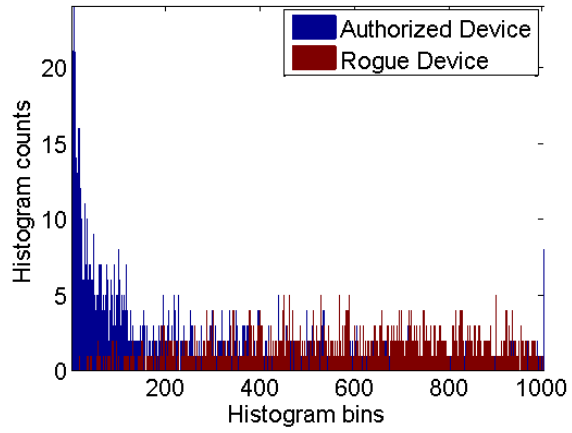
3.4.2 RF-DNA Fingerprinting Device ID Verification

Once a given classifier has been trained on *authorized* network devices, it can be used to identify *unauthorized* rogue devices attempting to gain network access. In this case, rogue devices impersonate authorized devices by presenting false bit level credentials matching those of an authorized device (a common approach for spoofing attacks). Rogue detection and rejection presents considerable challenge given their emissions were not available during classifier training. This was successfully addressed in [25, 35] using a biometric-based verification process with a similarity measure (test metric) reflecting how

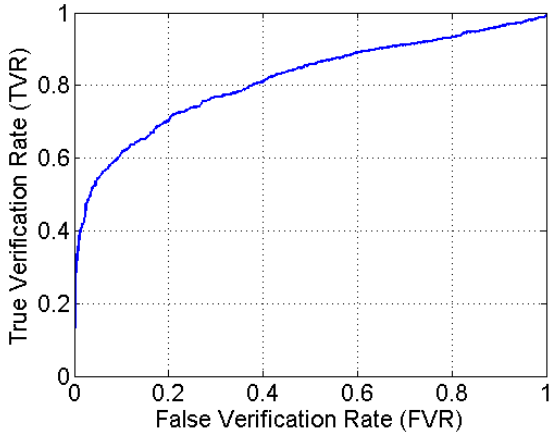
much a given rogue device "looks like" each of the authorized devices. Test metrics provide a level of confidence in the classifiers decision and are generated for each rogue device. For MDA/ML and RndF assessment, the test metrics include posterior probability estimates for each rogue device. For the GRLVQI assessment, the combined angle-distance metric in [106] is used.

Each of the N_r rogue devices are presented as having a claimed bit-level identity matching each of the N_a authorized devices, for a total of $N_a \times N_r$ rogue assessment scenarios. For each rogue-authorized device pair, 1000 bin histograms were generated using test metrics reflecting 1) how much the rogue device "looks like" the authorized device (Rog:Auth), and 2) how much the authorized device "looks like" itself (Auth:Auth). A representative rogue-authorized histogram pair is shown in Figure 3.5a where the span of histogram bin values is determined by the authorized device test metric values.

Histograms such as that shown in Figure 3.5a are used to generate Receiver Operating Characteristic (ROC) curves by varying a threshold (left-to-right) across the distributions and calculating True Verification Rate (TVR) and False Verification Rate (FVR). Figure 3.5b shows the associated ROC curve generated from the histograms in Figure 3.5b. TVR reflects a correct decision whereby an authorized device is present and correctly granted network access. FVR corresponds to Rogue Accept Rate (RAR) and reflects an incorrect decision whereby a rogue device is present and errantly granted network access. The resultant ROC curve for a given rogue-authorized device pair is simply generated by plotting FVR vs. TVR, with higher TVR and lower FVR points reflecting better performance. ROC curve assessments here are based on an arbitrary benchmark performance range defined by TVR>0.9 and FVR<0.1. For rogue-authorized scenarios that do not meet these criteria, the rogue is deemed to have successfully "spoofed" the network.



(a)



(b)

Figure 3.5: (a) Representative histograms of MDA/ML posterior probability test metrics for a rogue-authorized device pair at $SNR = 12.0$ dB, and (b) the associated ROC curve.

3.4.3 RF-DNA Fingerprinting: ZigBee XEnv Dataset

The experimental methodology from [93, 94] is presented here for completeness. The ZigBee XEnv dataset included observations from $N_D = 4$ devices, with 1000 observations from three different collection environments (*LOS, Wall, Cage*), divided evenly between test and training sets. Experiments were conducted initially with Variable Importance (VI)

analysis and RndF classification on the 5760 instantaneous amplitude ($a[n]$), phase ($\phi[n]$) and frequency ($f[n]$) variables (1920 from each domain). This dataset is denoted herein as the *XEnv Inst* dataset. Subsequent analysis focused on 729 RF-DNA statistical features extracted from these variables (time samples); this dataset is denoted herein as the *XEnv Stat* dataset. With the models built during classification of authorized devices, the 9 rogue devices are tested for network intrusion detection scenarios where they attempt to impersonate each of the 4 authorized devices. The number of rogue devices correctly rejected by the model is the metric used to compare different classifiers' verification performance.

3.4.4 RF-DNA Fingerprinting: ZigBee XRx Dataset

The XRx dataset consists of observations captured simultaneously from two NI receivers including 1) a high-cost PXIe receiver and 2) a low-cost USRP receiver [95]. A total of 300 observations each was used for training and test sets for each authorized device in the LOS collection environment. Receiver classification performance with RndF was assessed for RF-DNA statistical amplitude, phase and frequency features individually. Three devices were randomly chosen from a pool of 6 for classification purposes. Each such combination of ${}_6C_3$ authorized devices is tested for their classification performance to gain insight into the average classification performance using emissions each receiver. A classification study of all 6 devices was also conducted for both receivers. Following this, network intrusion detection via verification experiments was conducted for each choice of 3 authorized devices. Verification performance was compared between receivers to gauge performance trade-off from using a low-cost USRP receiver over the more costly PXIe receiver. Results from these experiments are in Chapter 7.

3.5 Summary

In this chapter, data collection methodology for SCA and RF-DNA applications was described. This was followed by general methodology for Linear Regression Attack,

RndF profiling attack and RF-DNA fingerprinting. RF-DNA classification and verification methods were described in addition to a summary of the two ZigBee datasets. In the following chapters, specific contributions to the fields of SCA and RF-DNA fingerprinting will be presented.

IV. Results: Linear Regression Attack

This chapter contains research results from the article 'Statistical Analysis and Comparison of Linear Regression Attacks on the Advanced Encryption Standard' accepted to appear in the *International Journal of Information Communication Technologies* [92].

4.1 Introduction

Side Channel Attacks (SCA) attempt to exploit unintended information leakage to derive sensitive information otherwise unobtainable by an adversary. Of particular interest is the secret key used during an encryption operation. Various SCA methods have successfully extracted the secret key from a variety of cryptographic hardware devices [61, 71, 72]. The stochastic attack, which we group under a larger class of Linear Regression Attacks, is shown to improve attack efficiency [115] by estimating the data dependent side channel leakage function for cryptographic hardware instead of assuming a fixed leakage function such as that used in the Hamming Weight model. By assuming a set of linear basis functions that span a function subspace of the overall leakage function, the requirement for the theoretical minimum number of training traces for adequate profiling is much reduced. By estimating the leakage function using actual training data, Linear Regression Attacks offer much more flexibility to attack hardware whose side channel leakage does not follow fixed models such the Hamming Weight model, either due to differences in manufacturing tolerance, chip layout or other causes.

Several methods have been developed that advance the state of the art in Linear Regression Attacks. In this research, four methods from literature and one new method are examined and compared. These methods use R^2, R_a^2, CPA, $||b||$ and symmetry to estimate the distribution of the non-data dependent leakage by choosing the best time samples to use

in profiling the training data. Several practical experiments are then conducted to gauge the envelope of performance of Linear Regression Attacks.

The rest of this work is structured as follows: Section 4.2 provides a background on SCA and CPA, Linear Regression, Stochastic Attack, and recent work in Linear Regression Attacks. Section 4.3 describes the hardware setup. Analyses of the linear regression models is conducted in Section 4.4 , followed by a comparison of methods in Section 4.5. Finally, results of practical experiments with Linear Regression Attacks are described in Section 4.6.

4.2 Background

In this section, a brief review of SCA and CPA is provided, followed by background on linear regression and the stochastic approach from [115], and finally a description of similar work in Linear Regression Attacks.

4.2.1 SCA and Correlation Attack

SCA obtains information about the internal operation of a device by observing unintentional information leakage from the device. Depending on the technology, a transistor may consume more power to output a high state than to output a low state. By monitoring the power consumption, or the 'power side channel' of the transistor, one can deduce the output state without directly monitoring the transistor output. This same effect occurs in devices composed of many transistors such as a microcontroller [72]. Analyzing the SCA power consumption of a microcontroller in operation can reveal the attributes of signals internal to the device otherwise inaccessible to the user. A significant advantage of SCA is that invasive procedures such as depackaging the device is often not required.

CPA is a subclass of SCA which uses statistical methods to determine information about an algorithm running on a device [19]. It assumes that the power consumption of the microcontroller follows the Hamming Weight (HW) power model [72]. Correlation is calculated between the theoretical power consumption represented as the HW of an

intermediate value of the encryption and the actual power measurement. As there is a one-to-one mapping between the key and the intermediate value, the key corresponding to the highest correlation is guessed as the correct key.

4.2.2 Linear Regression and Error Analysis

For an independent variables $X = \{x_1, \ldots, x_n\}$ and a variable Y dependent on X we seek a function relationship of the form by *Y=f(X)*. Linear regression can be used to build a linear model to represent the linear function f of the form

$$f(X) = \beta_0 + \sum_{i=1}^{n} \beta_i \cdot g_i(x_i, k). \tag{4.1}$$

For a multivariate linear regression, estimates $\{b_0, \ldots, b_n\}$ for the true model parameters $\{\beta_0, \ldots, \beta_n\}$ are determined. The residual error $\mathcal{R}$, or simply the 'residual', is the difference between the regression estimate and the measured value. Certain conditions are required for linear regression to provide accurate estimation:

- There is a *linear* relation between independent variables x_i and the dependent variable Y.
- $\mathcal{R} \sim \mathcal{N}(0, \sigma^2)$. This is referred to as the *normality assumption*.
- The variance σ^2 of the residuals $\mathcal{R}$ is constant over all observations. This is referred to as the *homoscedasticity assumption*.
- $\mathcal{R}$ is *uncorrelated* for all observations, i.e. $\sigma^2\{\mathcal{R}_i, \mathcal{R}_j\} = 0$ for all $i, j; i \neq j$

The coefficient of determination R^2 and adjusted coefficient of determination R_a^2 measure the proportionate reduction in total variation in Y associated with the use of $x_1, x_2, \ldots, x_u$ [65, 80] and are given by

$$R^2 = 1 - \frac{SSE}{SSTO} \tag{4.2}$$

$$R_a^2 = 1 - \left(\frac{n-1}{n-p}\right)\frac{SSE}{SSTO} \tag{4.3}$$

where SSE and SSTO are the sum of squares error and sum of squares total, respectively. These are given by

$$SSE = \sum \left(Y_i - \hat{Y}_i\right)^2 \tag{4.4}$$

$$SSTO = \sum \left(Y_i - \bar{Y}\right)^2 \tag{4.5}$$

where $\hat{Y}_i$ is the estimate of the i^{th} observation from the regression model and $\bar{Y}$ is the observed mean.

A weakness of the R^2 metric is that it will monotonically increase as independent terms are added. Thus adding independent terms that only marginally decrease the variance in Y will still raise the R^2 value. R_a^2, however, may decrease by adding an independent term that does not offset the loss of a degree of freedom caused by adding it to the regression, thereby demonstrating an increase in unexplained variation.

4.2.3 Stochastic Model for SCA

In this section, the stochastic model of [115] with the AES algorithm is described. Suppose the side channel of interest is the power measurement of the device when performing an encryption operation. An s-bit subkey is represented by $k \in \{0,1\}^s$ and corresponding known p-bit plaintext by $x \in \{0,1\}^p$. The power measurement at time t is then represented by

$$I_t(x,k) = h_t(x,k) + \epsilon_t \ , \tag{4.6}$$

where $I_t(x,k)$ and $h_t(x,k)$, respectively, are the total power measurement and deterministic power measurement for the encryption operation segment that depends on x and k at time t. The non-deterministic part of the power measurment ϵ_t does not depend on x or k. For an estimate $\hat{h}_t(x,k)$ of the deterministic power measurement, the minimum

$$\min_{\hat{h}:\{0,1\}^s\times\{0,1\}^p\to R} \sum_{i=1}^{N_1} \left(\left(I_t(x,k) - \hat{h}_t(x,k)\right)^2\right) \tag{4.7}$$

is attained when $\hat{h}$=h_t. In other words, the best deterministic power estimate is the one yielding the smallest residual $\mathcal{R}_t$ at time t given by

$$\mathcal{R}_t = I_t(x,k) - \hat{h}_t(x,k) \,. \tag{4.8}$$

Now let $\mathcal{F}_{u,t} \subset \mathcal{F}_t := \{\hat{h}_t : \{0,1\}^p \times \{0,1\}^s \rightarrow \mathbb{R}\}$ be a subspace of the full function space $\mathcal{F}_t$, and is spanned by u known linear basis functions $g_{i,t}(x,k)$ and $i = 1,\ldots,u$, then the deterministic power consumption for this subspace can be estimated by

$$\hat{h}_t(x,k) = b_{0t} \cdot 1 + \sum_{i=1}^{u} b_{it} \cdot g_i(x,k) \,. \tag{4.9}$$

The basis functions $g_i(x,k)$ can be constructed with a chosen intermediate value function $\phi(x,k)$ by the function composition $g_i(x,k) = \bar{g}_i(\phi(x,k))$. For example, $g_i(x,k)$ can give the i^{th} bit of the result of the intermediate value function $\phi(x,k)$. How $\phi(x,k)$ and correspondingly $g_i(x,k)$ are chosen is dependent on the attacker's knowledge of the cryptographic algorithm and device architecture. For example, if the attacker knows that the encryption algorithm is AES, he can choose $\phi(x,k)$ to be the SubBytes function for the first byte of the first encryption round, and $\bar{g}_i(x,k)$ extracts the i^{th} bit of that byte value.

A desired property is that the expectation $E(\mathcal{R}_t) = 0$ over x and k, i.e., on average the estimate will be correct. The least squares estimator is used to estimate the model parameters $\mathbf{b}_t = [b_{0,t},\ldots,b_{u,t}]$ as it meets this condition and provides a solution that minimizes the Sum of Squares Error (SSE). As a result, it produces the best estimator $\hat{h}_t^*$ for a given choice of basis functions. In the profiling phase, the least squares estimator is generated by

$$\mathbf{b}_t^* = (A^T A)^{-1} A^T I_t(\mathbf{x},k) \,, \tag{4.10}$$

where

$$A = \begin{pmatrix} 1 & g_{1,t}(x_1,k) & \ldots & g_{u-1,t}(x_1,k) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & g_{1,t}(x_{N_1},k) & \ldots & g_{u-1,t}(x_{N_1},k) \end{pmatrix}, \tag{4.11}$$

for N_1 observations, and $I_t(\mathbf{x}, k)$ is the set of measurements at time t for N_1 observations, each with a different plaintext x_i. Substituting 4.10 in 4.9,

$$\hat{h}_t(x, k) = (A^T A)^{-1} A^T I_t(\mathbf{x}, k) \times \mathbf{g}(\mathbf{x}, k)^T \ , \tag{4.12}$$

where $g_0(x, k) = 1$ for all realizations of x. Then, $\hat{h}_t^*$ is used to estimate the deterministic power consumption for a second set of N_2 observations and the residuals $\mathcal{R}_t$ are generated for each time sample t. The expanded derivation of the Least Squares Estimator is provided in Appendix A.

The probability density function $f(\mathcal{R})$ for the residuals $\{\mathcal{R}_1, \ldots, \mathcal{R}_m\}$ is estimated with an m-variable multivariate Normal distribution. For a good estimate, only m time samples with maximum information leakage should be used. Methods in [34, 50, 92, 115] follow the same basic strategy to this point. They differ in their approach to determine the time samples where there is high information leakage, which are subsequently used for determining the correct m dimensions for $\mathcal{R}_t$. Once these time samples are chosen, the regression residuals $\mathcal{R}_t$ from the training data at these m points in time are then used to estimate a covariance matrix C, effectively building a template for the multivariate estimation residual $\mathcal{R}$ for a correct key guess. As there is a zero mean for the residuals given by the $E[\mathcal{R}_t] = 0$ requirement of the least squares method, C alone is needed to build an m-dimensional normal density function,

$$f(\mathcal{R}) = \frac{1}{\sqrt{(2\pi)^m}} \exp\left(-\frac{1}{2}\mathcal{R}^T C^{-1} \mathcal{R}\right) . \tag{4.13}$$

By assuming that the subspace $\mathcal{F}_{u,t}$ contained in $\mathcal{F}_t$ is spanned by the u known linear basis functions, only this single template for $\mathcal{R}_t$ is required.

In the second phase, known as the *key extraction phase*, N_3 test measurements are used that have an unknown fixed key and random plaintext. The Maximum Likelihood method is applied on the N_3 test traces as follows

$$\alpha(x_1, \ldots, x_{N_3}; k') = \prod_{j=1}^{N_3} \tilde{f}_{k'}(\mathcal{R}) \tag{4.14}$$

for all sub-keys $k' \in \{0,1\}^s$, $\mathcal{R}=(i(x_j,k)-\tilde{h}^*(x_j,k'))$ for time instants $t_1,\ldots,t_m$. The choice of k' that gives the maximum probability is chosen as the sub-key hypothesis. As multivariate densities often produce very small probabilities, especially as m grows, an alternate method is to use the log-likelihood method and sum the exponents in (4.13) by

$$\alpha(x_1,\ldots,x_{N_3};k') = -\sum_{j=1}^{N_3}(\mathcal{R}^T C^{-1}\mathcal{R}). \tag{4.15}$$

4.2.4 Related Work

After the initial work in [115] more recent research developments have advanced the state of the art of Linear Regression Attacks. In [50], a new symmetry metric is introduced to compare leakage models for stochastic attacks. A leakage model with good symmetry will have the property

$$\beta_{j,t,k'_{(y)}} = \beta_{j,t}, \text{ for all } k'_{(y)} \in \{0,1\}^u, \tag{4.16}$$

meaning that the model parameters β for the y^{th} byte of the key are equal for all subkey values for that byte. The target cryptographic functions are modified to allow for 0 mean and orthonormal bases. For example, the function

$$S^{-1}(x_{(y)} \oplus k_{(y)})_j \tag{4.17}$$

for the value of the xor of the 9^{th} AES round output with the 10^{th} round key is modified to

$$2((S^{-1}(x_{(y)} \oplus k_{(y)}))_j - 0.5) \text{ for } j = 1,\ldots,8 \tag{4.18}$$

A degree of symmetry metric is defined for a 8-bit subkey as

$$\frac{2\sqrt{\sum_{j=1}^{8}(b_{j,t,k'} - b_{j,t,k''})^2}}{\sqrt{\sum_{j=1}^{8} b_{j,t,k'}} + \sqrt{\sum_{j=1}^{8} b_{j,t,k''}}}, \tag{4.19}$$

where k' and k'' are different subkey values. Values from (4.19) are lower for leakage models with better symmetry (inverse relation). Leakage models with good symmetry were found to perform better in stochastic attacks than those with poor symmetry. The

definition of symmetry has a close tie to the Equal Images under different Subkeys (EIS) property defined in [115] for stochastic attacks. Choosing functions with good symmetry and EIS will allow any subkey value to be used for profiling, and the regression model estimated therefrom can be used to attack any subkey value. An important consequence of the symmetry calculation shown in [50] is that points in time with good symmetry coincide with times of high information leakage. Also shown is that these points in time had the highest maximum estimated model parameter values $|\mathbf{b}_t|$ if only $\{b_{1,t}, \ldots, b_{u,t}\}$ are considered.

The coefficient of determination is introduced in [34] as a metric for goodness of fit for Linear Regression Attacks such as the stochastic attack. The formula for R^2 for the linear regression modelled with key guess k' is provided as

$$R^2(k') = \frac{E((\tilde{h}_t^*(x,k) - h_t(x,k)^2)}{var(h_t(x,k)^2)}. \tag{4.20}$$

This equation can be further broken down to

$$R^2(k') = \frac{SSE}{SSTO}, \tag{4.21}$$

which is inversely related to the classical definition in (4.2) [65, 80]. Although [34] compares non-profiled attacks, the definition of R^2 is still applicable for profiled Linear Regression Attacks. Points with the highest R^2 were chosen for Linear Regression Attack and found to be superior to CPA in terms of minimal number of traces required for a correct key guess [34].

In all of these approaches, the statistical significance of the linear regression is not considered. That is, the linear regression is not analysed to determine if the regression model is valid nor whether the model satisfies the assumptions of linear regression. This work fills this void by examining the byte-wise linear regression model using the methods described in this section as well as R_a^2 and CPA to determine time samples of interest.

4.3 Hardware Setup

The data collection methodology is described in this section. Majority of the experiments were conducted on a PIC-24FJ64GA002 microcontroller running an AES-128 algorithm. This microcontroller is called *Yellow1*. For later hardware comparison tests, an additional microcontroller from the same Part Number (PN) as *Yellow1* was used and called *Yellow2*. In addition *Red1* from PN PIC-24FJ48GA002 and *Blue1* from PN PIC-24FJ64GA102 were used. The *Yellow* and *Red* microcontroller families differed only in the amount of on-board memory. The *Blue* microcontroller family differed the most from the other microcontrollers as its hardware peripherals and on-board memory were different.

The EM measurements during the AES-128 encryption operation are the side channel of interest for all experiments. Traces were initially collected at a 2.5GHz sampling rate and then low-pass filtered and downsampled to 104 MHz. This is roughly 4 times the PIC clock frequency of 27 MHz, thus meeting Nyquist sampling criteria. Matlab, JMP and SPSS statistical software were used to perform all regression analysis.

The intermediate value function $S(x \oplus k)$ in the first round of AES is the calculation of interest in all cases. In [50], it is shown that leakage models with good symmetry perform better for Linear Regression Attacks such as stochastic attack. Good symmetry can be determined by points in time t with large $|b_{1,t}, \ldots, b_{8,t}|$, excluding the intercept $b_{0,t}$. Figure 4.1 plots the maximum $|b_{j,t}|$ for the estimator using the $S(x \oplus k)$ function. There are localized large peaks surrounded by low values at the remaining points which are an indicator of a good leakage model [50]. So this function is used in our leakage model for testing Linear Regression Attacks in this research.

4.4 Linear Regression Analysis

In this section, the linear regression is analysed for a time instant with maximum information leakage as proposed by the following five methods:

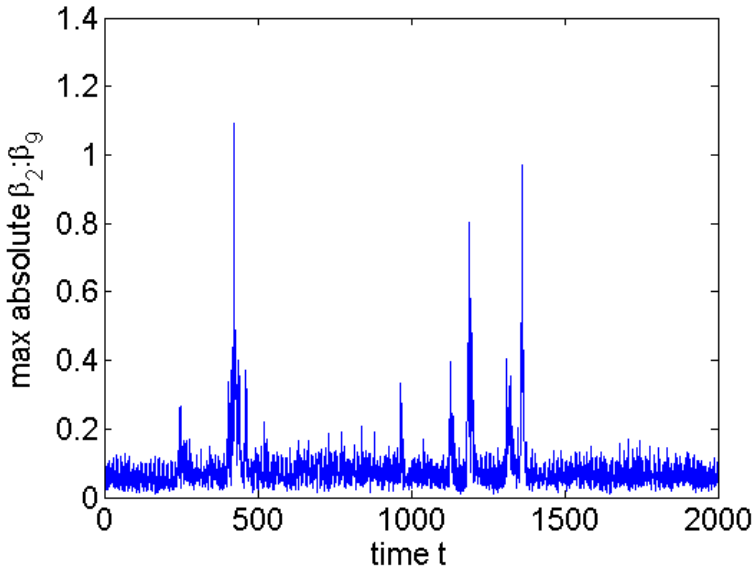


Figure 4.1: Maximum $|b_{1,t}, \ldots, b_{8,t}|$ values from the estimator $h_t^*(x,k)$ using intermediate value function $S(x \oplus k)$ for $t = \{1, 2, \ldots, 2000\}$

1. **Method$_{\|\mathbf{b}\|}$:** Time instants with high L^2-norm value for the model parameters defined in [115].

2. **Method$_{\mathbf{sym}}$:** Time instants with high $|b|$ corresponding to the best symmetry defined in [50].

3. **Method$_{\mathbf{CPA}}$:** Time instants with high correlation from a CPA attack [19].

4. **Method$_{\mathbf{R^2}}$:** Time instants with high coefficient of determination R^2 defined in [34].

5. **Method$_{\mathbf{R_a^2}}$:** Time instant with high adjusted coefficient of determination R_a^2 defined in Section 4.2.

These methods will be referred to by these titles for the remainder of this research. These methods differ in their choosing the correct time samples (variables) to build the template for C. The estimation of $h_t^*(x,k)$ and key-extraction phases are the same for all methods.

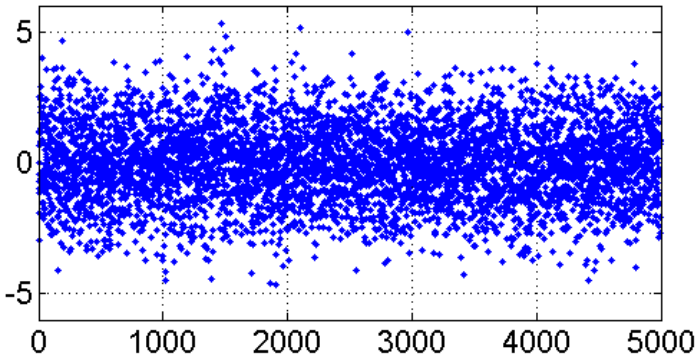
4.4.1 Analysis of Regression Assumptions for Method$_{\mathbf{R_a^2}}$

In this section, the linear regression assumptions described in Section 4.2.2 are analysed for Method$_{\mathbf{R_a^2}}$. Deviations from these assumptions can lead to poor prediction performance of the linear regression, and hence it is important they are checked and not assumed. To our knowledge, this is the first research where this analysis has been performed. The byte-wise linear regression in (4.9) is performed for each time instant of the training traces collected for the first sub-byte. A strong linear regression results in a high value for its coefficient of determination R^2. For multiple regression such as in this case, the *adjusted* coefficient of determination R_a^2 can be used to give higher weight to linear models where all independent variables are contributing significantly. Linear regression assumptions are now analysed for the time instant t with maximum information leakage as determined using Method$_{\mathbf{R_a^2}}$with N_1=5000 training traces.

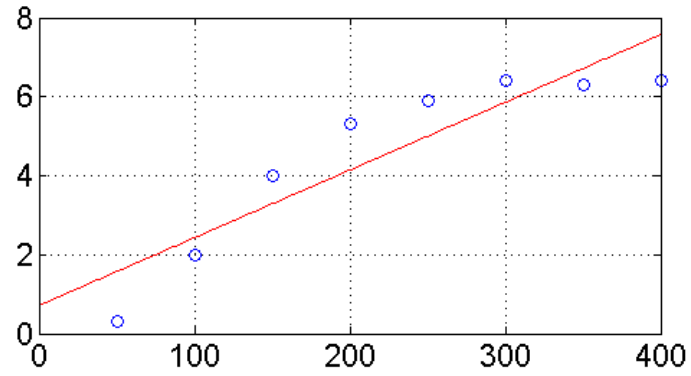
Linearity Assumption: The linearity assumption can be tested by examining the plot of residuals $\mathcal{R}_t$ [65], as shown in Figure 4.2a. The plot shows a uniform shape for the residuals, thus indicating a linear relation between independent and dependent variables. If the relation were non-linear, the shape would take on non-linear shapes such as curves, sinusoids, etc., an simulated example of which is shown in Figure 4.2b.

Table 4.1: SSE after Box-Cox transformation $Y' = Y^{\lambda}$.

λ	SSE
-2	11931.68
-1	11337.56
0	10975.98
1	10821.03
2	10858.37



(a)



(b)

Figure 4.2: (a) Plot of residuals $\mathcal{R}_t$ training traces showing a linear shape, and (b) simulated example of residuals for a non-linear regression.

As a further test for linearity, a Box-Cox transformation on the data was performed [65]. This method chooses different values of λ to transform the data to

$$Y' = Y^{\lambda}, \tag{4.22}$$

except for $\lambda = 0$, where the transformation is $Y' = ln(Y)$. If the SSE is significantly reduced by choosing λ much different than $\lambda = 1$ (no transformation), then it can be concluded that there is a problem with the linearity assumption in the linear regression. Table 4.1 shows

the SSE after several non-linear transformations and indicates that $\lambda = 1$ gives the lowest SSE. Thus no transformation is required and the linearity assumption of the data holds.

Homoscedasticity Assumption: The homoscedasticity or constant variance assumption can also be tested by examining the residual plot in Figure 4.2a. The uniform 'bar' shape of the residuals over time is an indicator of constant variance. If the variance were non-constant, the residual plot would take on a different shape such as a cone shape.

Normality Assumption: The assumption $\mathcal{R}_t \sim \mathcal{N}(0, \sigma^2)$ can be tested by plotting the residuals in a *quantile-quantile (QQ)* plot shown on the right in Figure 4.3. The residuals are plotted along the normal distribution line on the right. In this figure, residuals fall close to the normal distribution line, thus upholding the normality assumption. Also plotted on the left is the distribution of the data, which follows the red normal distribution line well.

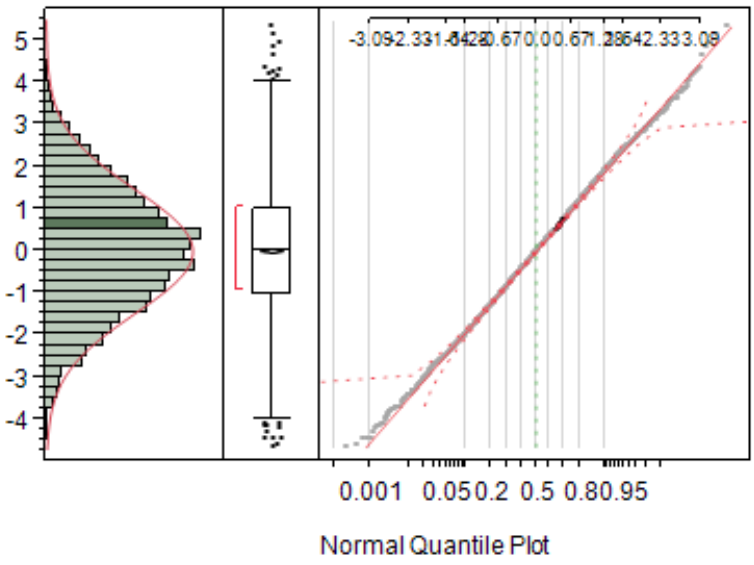


Figure 4.3: QQ plot of residuals with distribution.

Independence Assumption: The residuals $\mathcal{R}_t$ are collected from traces processing fixed key k and uniformly random plaintext x and are thus assumed to be independent of each other. In order to test this assumption, the Durbin-Watson (DW) test for

autocorrelation is applied [65]. It uses the following hypothesis test for autocorrelation parameter ρ:

$$H_0 : \rho = 0$$
$$H_1 : \rho > 0. \tag{4.23}$$

If the DW test statistic D is greater than the upper bound d_u which can be obtained in pre-computed tables [80], then the null hypothesis H_0 is accepted, thus showing no problem with independence in the data. For this regression, $d_u = 1.91$ and $D = 2.02$ at the 99% confidence level, thus showing uncorrelated residuals. In addition, the autocorrelation for the data set was found to be -0.0116, which is nearly 0. Thus the independence assumption holds.

Test for Outliers: Outliers in the data set can skew the linear regression by moving the fitted line toward them. They can be determined by examining the standardized residuals from the regression calculated by

$$\mathcal{R}_{t,std} = \frac{\mathcal{R}_t}{\sqrt{MSE}}, \tag{4.24}$$

where MSE is the Mean Square Error computed by SSE/($N-p$ observations). Observations with standardized residuals greater than 4 are potential outliers in the data. Figure 4.4 shows 10 observations with standardized residuals greater than 4. With N_1=5000 training traces, the effect of these 10 outliers was not significant enough to skew the regression line significantly. Removing these observations from the data did not significantly impact the R_a^2 value of the regression and hence were left in the training data set. However, for smaller training sets outliers can have significant impact leading to erroneous prediction.

4.4.2 Linear Regression Analysis of Other Methods

Method$_{\|\mathbf{b}\|}$described in [115] was used to find time instant with highest $\|b\|$. The data at this time instant was analyzed in a similar method as in Section 4.4.1. No linear regression assumptions were violated. The R_a^2 value was 9.6e-2 indicating a poor linear regression.

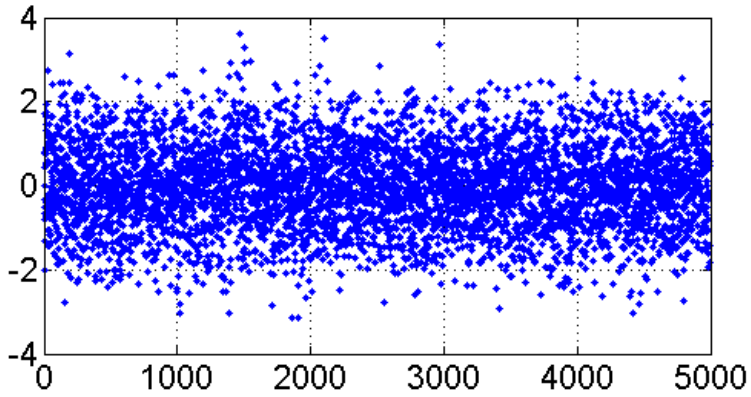
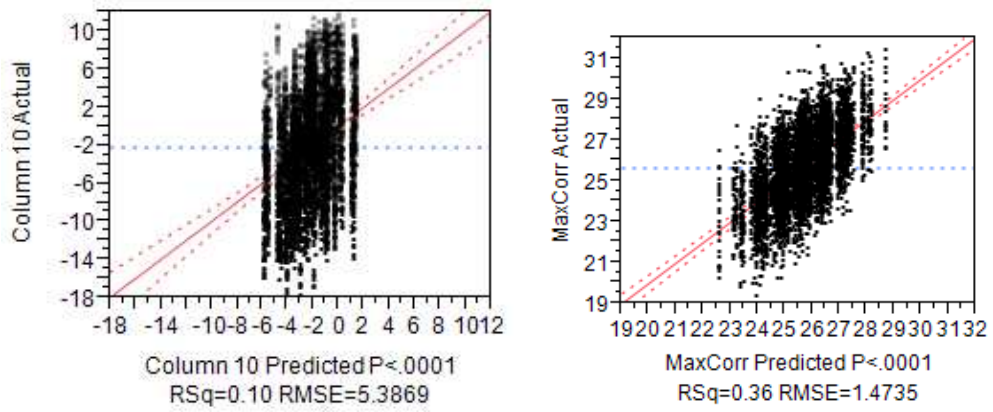


Figure 4.4: Plot of standardized residuals

Figure 4.5 shows the actual versus predicted values of the training observations for the linear regressions chosen with Method$_{\|\mathbf{b}\|}$and Method$_{\mathbf{R}_\mathbf{a}^2}$. Figure 4.5a shows almost no correlation between the actual measured values at that time instant and the predicted values by the linear regression model. In addition, Sum of Squared Lack of Fit (SSLF) [65] from the model using Method$_{\|\mathbf{b}\|}$was a very high 1.12e5 compared with the relatively low SSLF of 5.61e2 with the model using Method$_{\mathbf{R}_\mathbf{a}^2}$, indicating serious issues with lack of fit for this regression. We stress the correct key guess can still be attained even using estimators at time instants with low prediction accuracy when predictions over multiple test traces are accumulated as in (4.14). However, the number of required test traces is expected to be higher for poorer predictors.

No outliers were observed as determined by examining the standardized residuals. Method$_{\mathbf{sym}}$identified the same best time instant as the Method$_{\|\mathbf{b}\|}$. Method$_{\mathbf{CPA}}$and Method$_{\mathbf{R}^2}$identified the same best time instant as the Method$_{\mathbf{R}_\mathbf{a}^2}$. Thus the analyses of the regressions found using Method$_{\mathbf{R}_\mathbf{a}^2}$and Method$_{\|\mathbf{b}\|}$are applicable to the remaining three methods. It is thus concluded that five methods produced models that did not violate any linear regression assumptions.



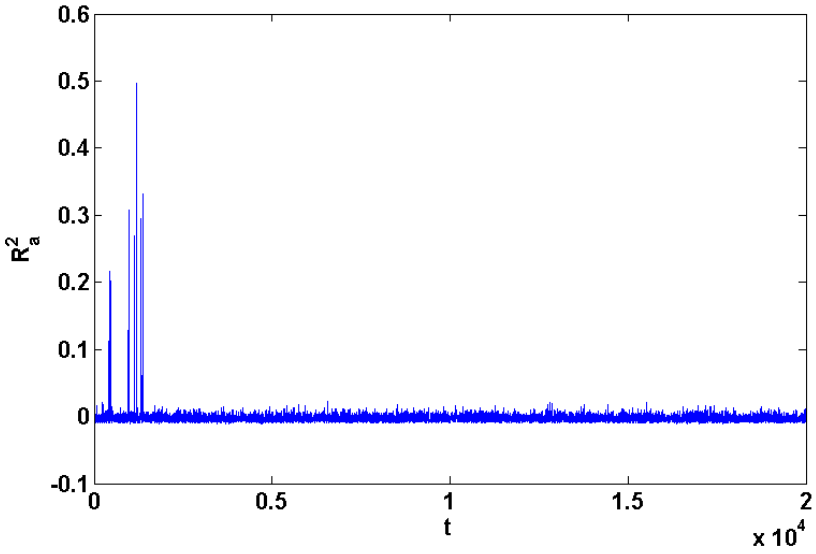
(a) (b)

Figure 4.5: (a) Plot of actual vs predicted values of the training traces using the best time sample with Method$_{\|\mathbf{b}\|}$, and (b) plot of actual vs predicted using the best time sample with Method$_{\mathbf{R_a^2}}$.

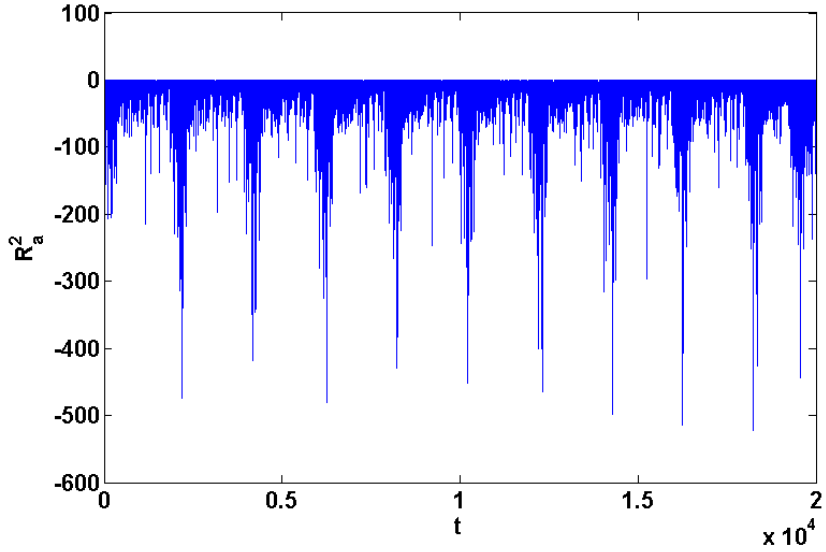
4.4.3 Linear Regression with Intercept

In this section, the importance of using the intercept in the linear regression is analysed. The intercept b_0 for a byte-wise Linear Regression Attack is ~60 times larger than the remaining model parameters and represents the power drawn from the chip that is not directly caused by the data, such as power drawn from operations on the remaining bytes, other electronic devices, environment, etc. In order to assess the importance of including the intercept in the regression, two models were tested one with an intercept and one without.

Figure 4.6 shows the R_a^2 values for the linear regression models with and without the intercept. From the equation for R_a^2 in (4.3), negative values indicate a larger Sum of Square Error (SSE) than Sum of Square Total ($SSTO$), indicating that the model estimate is worse than simply using the observation mean, and is thus a poor fit. It is clear from the figure that including the intercept in the model yields a much stronger regression.



(a)



(b)

Figure 4.6: (a) R_a^2 values for linear regression models with intercept, and (b) without intercept.

Recent work in [49] shows an improvement in Stochastic Attack (SA) performance when using a a model known as Stochastic Attack Offset Tolerant Method (SA-OTM) which does not include an intercept. However, in SA-OTM, the difference of traces is used to build the linear regression instead of the traces themselves. As a result, much of the non-data dependent information is removed, thus partially negating the need for an intercept. In addition, the profiling and key extraction phases are significantly changed in SA-OTM, all of which jointly contribute to the improved performance and it is not apparent how much improvement was from removing the intercept alone. Thus, although SA-OTM advances the state of the art of Linear Regression Attacks, its success comes from a number of factors and not simply from the removal of the intercept. Thus it cannot be concluded from [49] that removal of the intercept alone improves performance in Linear Regression Attacks.

4.4.4 Interaction Terms

As the PIC microcontrollers used in this research are 16-bit, processing two bytes at a time, it is possible that there can be an interaction between bits of adjacent bytes. In order to test this theory, the first two bytes of the intermediate function $x \oplus k$ are examined together. AES SubBytes table lookup for this microcontroller was restricted to 1 byte at a time so multi-byte analysis with $S(x \oplus k)$ function was not possible. A step-wise linear regression performs an investigation of all proposed model terms by systematically removing and adding terms to determine their benefit in improving the linear regression. This was performed with all 2-way interaction terms $g_i(x,k) \times g_j(x,k)$ for all $i, j \in \{0,1\}^{16}$ where $\times$ indicated multiplication, and each was examined for its importance to the prediction performance. The linear model with all interaction terms is estimated by

$$\begin{aligned} \tilde{h}_t^*(x,k) = b_{0t} + \sum_{i=1}^{u} b_{it}.g_i(x,k) + \\ \sum_{i=1}^{u}\sum_{j=1}^{u} b_{ijt}.\Big(g_i(x,k) \times g_j(x,k)\Big), \end{aligned} \tag{4.25}$$

and terms that are not useful toward prediction are removed by the step-wise process.

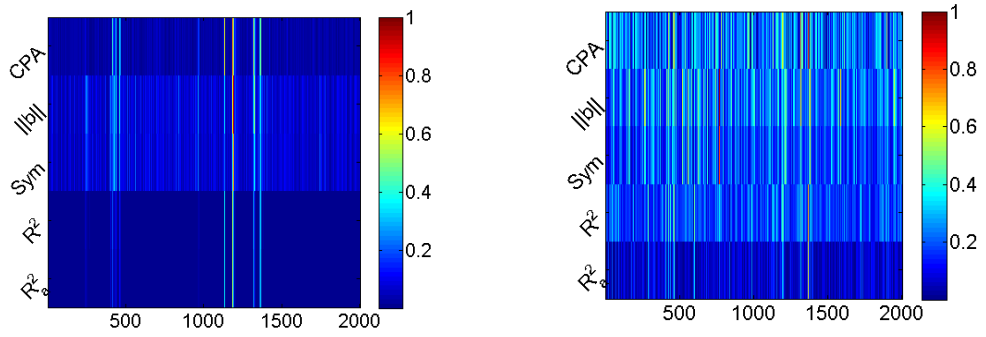
Several interaction terms were found that provided slight performance improvement, showing that information from one bit leaks into another. However, the benefit is minimal. Table 4.2 shows the SSE accounted for by $g_1(x,k)$ as well as several interaction terms. It is evident that the interaction terms do not account for nearly as much of variance reduction as the single term $g_1(x,k)$. Not including them does not significantly impact the linear regression prediction performance. Thus they are left out of the regression model and the original model in (4.9) is used for all subsequent experiments.

Table 4.2: SSE accounted for by $g_i(x,k)$ and interaction terms $g_i(x,k) \times g_j(x,k)$

Model term	SSE
$g_1(x,k)$	274.37
$g_3(x,k) \times g_1(x,k)$	0.92
$g_2(x,k) \times g_7(x,k)$	11.62
$g_4(x,k) \times g_{16}(x,k)$	0.04
$g_5(x,k) \times g_{17}(x,k)$	12.10

4.5 Comparison of Linear Regression Methods

Performance in the key-extraction phase is directly tied to accurately building the covariance matrix template. The five methods in Section 4.4 were used with the estimated leakage function (4.9) to determine their ability in accurately estimating the covariance matrix of the residuals. These methods identify the m time samples in the traces where there is information leakage. The residuals $\mathcal{R}_t$ where $t \in \{1, 2, \ldots, m\}$ from the training data at these m time instants is used to build the multivariate distribution of $\mathcal{R}$ defined in (4.13). A good selection of these m variables will give a closer approximation of $f(\mathcal{R})$ and consequently give better key prediction performance.



(a) (b)

Figure 4.7: Intensity plot of variable importance by using five different methods with 5000 training traces (a) , and with 100 training traces (b)

Using N_1=5000 training traces, the results of the variable importance methods are plotted in the intensity plot Figure 4.7a. For plotting purposes in Figure 4.7 only, in order to provide a fair comparison, variable importance results from the five methods were normalized to a range of 0 to 1 by dividing each by their respective maximum values. Figure 4.7a shows that although some points are identified as important by all five methods, Method$_{\mathbf{R_a^2}}$and Method$_{\mathbf{R^2}}$show better noise suppression evidenced by regions of very low values (solid blue) between spikes, as compared to the Method$_{\mathbf{||b||}}$, Method$_{\mathbf{CPA}}$and Method$_{\mathbf{sym}}$. This can be advantageous when attacking data sets with large amounts of non-data dependent noise.

Table 4.3 shows the results of the key extraction phase using the training data at times identified by the 5 different methods. Key extraction was performed using the sum of exponents in (4.15). Training sets came from the *Yellow1* microcontroller while test sets came from the *Yellow2* microcontroller, which is from the same part number, but different physical chips. The success rate for i = 100 iterations is simply the percentage of correctly

Table 4.3: Success Rate for 100 iterations using training and test traces from *Yellow1* and *Yellow2* microcontrollers, respectively. The number of training and test traces and the dimensionality m is varied between experiments.

Test setup	$\text{Method}_{\|\mathbf{b}\|}$	$\text{Method}_{\mathbf{sym}}$	$\text{Method}_{\mathbf{CPA}}$	$\text{Method}_{\mathbf{R^2}}$	$\text{Method}_{\mathbf{R_a^2}}$
100 Train 10 test, m=1	0.0506	0.0175	0.0887	**0.0925**	**0.0925**
100 Train 100 test, m=1	0.0519	0.0281	**0.0988**	**0.0988**	**0.0988**
100 Train 100 test, m=20	0.8912	0.8119	**0.9806**	0.9719	0.9719
500 Train 10 test, m=1	0.0838	0.0719	0.0919	**0.0944**	**0.0944**
500 Train 100 test, m=1	0.8331	0.7250	0.8850	**0.8894**	**0.8894**
500 Train 100 test, m=20	0.9975	0.9900	0.9956	**0.9981**	**0.9981**
5000 Train 10 test, m=1	**0.1019**	0.0969	**0.1019**	**0.1019**	**0.1019**
5000 Train 100 test, m=1	0.8862	0.8363	**0.9206**	**0.9206**	**0.9206**
5000 Train 100 test, m=20	0.9981	**1.0000**	0.9962	0.9994	0.9994

guessed sub-keys out of 16 over 100 iterations[1]. For each iteration, a random selection of

[1]AES-128 uses a 16 byte key

test and training traces is used out of 5000 trace pools. Thus these results show the 100-fold validation results for these experiments. Best results are highlighted in bold. For all experiments in this section, $N_1 = N_2$ for training traces.

Results show that performance is improved the most by raising the number of time samples m, effectively raising the dimensionality of the multivariate distribution. For example, with 100 training traces and 100 test traces, raising m from 1 to 20 improves success rate by almost an order of magnitude in $\text{Method}_{\mathbf{R}_a^2}$ from 0.0925 to 0.9719. Similar performance improvement is also seen for 500 and 5000 training trace cases. These results show that a multivariate distribution is more successful in Linear Regression Attacks than univariate. The probability distribution $f(\mathcal{R})$ estimated in the training phase in (4.13) is dependent on the covariance alone, as $\mathcal{R}_t$ has a zero mean for each $t \in \{1, \ldots, m\}$. Raising the dimensionality m provides greater definition to the covariance matrix allowing the probability estimate to better classify the correct sub-key from incorrect ones. It should be noted that even with as few as 100 training and test traces, with adequate number of dimensions m, the attack can still attain a success rate as high as 0.9806 (or 98.06%). As the linear basis functions are assumed to be known to the attacker and the *function* $h'_t(x, k)$ is estimated instead of the *function values* of $h'_t(x, k)$, only one template of $f(\mathcal{R})$ is required to be estimated. So successful attack is possible using a template generated with as few as 100 training traces in our experiments, which is fewer than the number of unique elements in the image of $\phi(x, k)$.

The next highest performance benefit is obtained by increasing the number of test traces. For 100, 500 and 5000 training traces with $m = 1$, raising the number of test traces in each case from 10 to 100 for each case improves performance by 17.63%, 969.96% and 990.13% respectively. This suggests that for these data sets linear regression classifier is not strong enough to correctly guess the correct sub-key for individual traces. In fact, if the attack is performed on N_3=5000 traces from the same test set with $\text{Method}_{\mathbf{R}_a^2}$, the correct

key was guessed for only 35 test traces or 0.70% correct classification rate. Only when information gain is accumulated over many test traces with the product of probabilities in (4.14), or sum of exponents in (4.15), does the attack classify the correct sub-key.

Finally performance benefit is also realized by raising the number of training traces. With 10 test traces and $m = 1$, raising the number of training traces from 100 to 500 and 100 to 5000 improves average success rate by 76.84% and 118.06% respectively. Increasing the number of training traces leads to a better estimate of $h_t(x, k)$ and $f(\mathcal{R})$, thereby improving prediction performance.

Overall, best performance in the majority of cases was shown for $\text{Method}_{\mathbf{CPA}}$, $\text{Method}_{\mathbf{R_a^2}}$, and $\text{Method}_{\mathbf{R^2}}$. $\text{Method}_{\mathbf{R^2}}$ and $\text{Method}_{\mathbf{R_a^2}}$ have identical performance in all cases. The R_a^2 metric rewards distributions where all independent variables $g_i(x, k)$ contribute meaningfully to the regression. In the byte-wise linear regression, each variable was found to contribute significantly to the $\alpha < 0.0001$ or with greater than 99.9999% confidence level. This is to be expected as each $g_i(x, k)$ represents a bit of the sub-byte of the leakage model, all of which contribute to the data dependent power measurement $h(x, k)$. As each independent variable is important to the regression, $n >> p$, and n and p are fixed for an attack, $R^2 \approx R_a^2$, which can be seen in the identical variable importance intensity plots for $\text{Method}_{\mathbf{R^2}}$ and $\text{Method}_{\mathbf{R_a^2}}$ in Figure 4.7a. It is expected, however, that $\text{Method}_{\mathbf{R_a^2}}$ and $\text{Method}_{\mathbf{R^2}}$ will show different results when these conditions are not met. Equation (4.3) as a function of n is a monotically increasing function, where $R_a^2 \rightarrow R^2$ as the difference between n and p increases. In addition, from equation (4.3) we see the R_a^2 as a function of the error term $e = \frac{SSE}{SSTO}$ is monotonically decreasing. Combined these two functions indicate that for small training sets as $n \rightarrow p$, the rate of decrease in R_a^2 is less for larger e. If R_a^2 is used as a variable importance metric, variables of high importance with large R_a^2 will decrease at a slower rate than variables of low importance as n decreases (or conversely, as p increases). For small n (or large p), variables of high importance will stand out more

with R_a^2 than R^2 and this is proven in Figure 4.7b where important variables are easier to identify using R_a^2 than R^2. Thus for smaller training data sets, $\text{Method}_{\mathbf{R_a^2}}$would prove more useful at identifying important points than $\text{Method}_{\mathbf{R^2}}$.

$\text{Method}_{\mathbf{CPA}}$shows similarity in success rates to that from $\text{Method}_{\mathbf{R^2}}$and $\text{Method}_{\mathbf{R_a^2}}$and is most likely a consequence from the fact that the Pearson coefficient of correlation is the signed square root of the coefficient of determination R^2 (for positive R_a^2 values). Hence both metrics find similar peaks when measuring variable importance, as shown in Figure 4.7a.

Table 4.4: Mean and variance of model parameter values b for time samples chosen by different methods.

	Mean	Variance
$\text{Method}_{\mathbf{\|\|b\|\|}}$	0.9152	0.0090
$\text{Method}_{\mathbf{sym}}$	0.8857	0.0120
$\text{Method}_{\mathbf{CPA}}$	0.8374	0.0026
$\text{Method}_{\mathbf{R^2}}$	0.8187	0.0035
$\text{Method}_{\mathbf{R_a^2}}$	0.8187	0.0035

$\text{Method}_{\mathbf{\|\|b\|\|}}$and $\text{Method}_{\mathbf{sym}}$performed best when dimensionality m was increased and when training set size increased but otherwise fell behind the other methods. These two methods contend that larger linear model parameters lead to a better regression. In a linear regression, if a model parameter is close to 0, this indicates that this parameter is not contributing to the regression. So linear models where parameters are all non-zero are stronger than those with models with zero-value parameters. However, in the case of Linear Regression Attack on a PIC microcontroller, if the standard HW power model is assumed which gives an equal weight for every bit, then estimated model parameters $|b_{1,t}, \ldots, b_{8,t}|$ for the byte-wise linear regression at time t should ideally be equal. The

strength of this assumption for the PIC is verified by the good performance of Method$_{\mathbf{CPA}}$ in Table 4.3 which uses the HW leakage model for CPA attack. With the HW assumption the $|b_t| = |b_{1,t}, \ldots, b_{8,t}|$ values greater than the common weight w would generate inaccurate predictions with the severity of performance degradation increasing as estimates of $|b_t|$ increase beyond w. In addition, the HW model gives equal weight to each bit and so linear models with less variation between model parameters adhere better to the HW model. Table 4.4 shows the mean and variance of the $|b_t|$ values for linear regressions from the best time sample chosen by the 5 methods. None of the methods has a perfect 0 variance. Two explanations are available, and the true cause is likely a combination of the two: 1) the HW model does not exactly fit the real hardware, and 2) model parameters b_t are not perfectly estimated with the training data due to the presence of noise unaccounted for by the regression model. Of further note, the variance in model parameters from Method$_{\mathbf{CPA}}$, Method$_{\mathbf{R^2}}$ and Method$_{\mathbf{R^2_a}}$ are between 3 and 4 times smaller than that for the other two methods. This divergence of the model parameter estimates b_t from the HW model of 0 variance results in slightly poorer performance of Method$_{\mathbf{\|b\|}}$ and Method$_{\mathbf{sym}}$ compared to the other methods. Figure 4.8 shows the Global Success Rate (GSR) as defined in [118] for the Linear Regression Attacks using the five methods to determine the m=20 variate distribution $f(\mathcal{R})$. 100 training traces each were used for generating $h^*_t(x,k)$ and the distribution estimate for $\mathcal{R}$. Method$_{\mathbf{CPA}}$, Method$_{\mathbf{R^2}}$ and Method$_{\mathbf{R^2_a}}$ best identified the time samples that best approximated $f(\mathcal{R})$, resulting in improved prediction performance.

4.6 Practical Performance Characterization of Linear Regression Attacks

In this section, several experiments are conducted to gain a better understanding of the performance of Linear Regression Attacks in various practical scenarios. These experiments reveal attack performance when training and target microcontrollers and collection equipment differ, as well as performance in noisy environments.

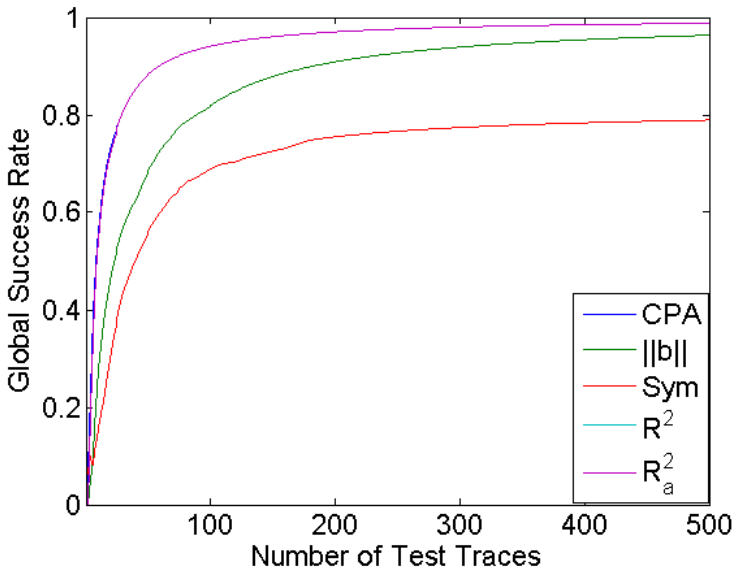
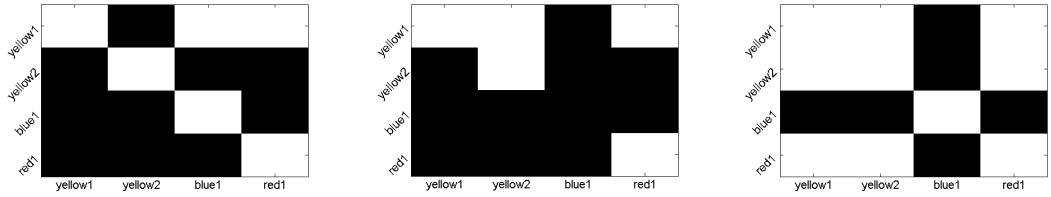


Figure 4.8: Global Success Rate with 100 training traces. Performance curves of Method$_{\mathbf{R_a^2}}$, Method$_{\mathbf{R^2}}$ and Method$_{\mathbf{CPA}}$ overlap.

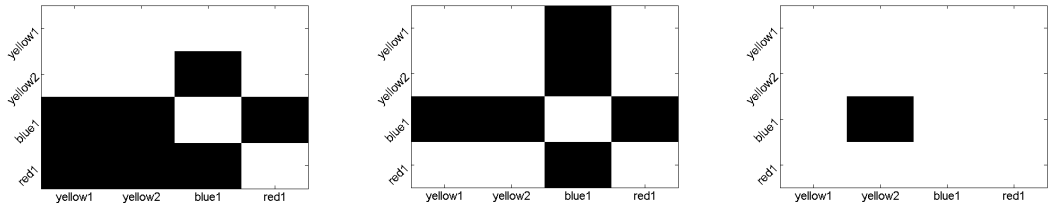
4.6.1 Profiling and Testing with Different ICs

The main purpose of profiling attacks such as Linear Regression Attack is to develop the ability to profile a microcontroller and attack a similar but not identical microcontroller. To test this, Method$_{\mathbf{CPA}}$ and Method$_{\mathbf{R_a^2}}$ were applied for profiling and key extraction on four microcontrollers: *Yellow1, Yellow2, Blue1 and Red1* as described in Section 4.3. With the hardware differences between the microcontroller models, it was hypothesized that *Blue1* would be the most difficult to attack as differences in the times when operations are executed are not expected to align well between *Blue* and other families. However, if the number of chosen time instants is increased, similarities in hardware architectures may turn up time instants that both architectures have in common and allow for meaningful profiling between hardware architectures. If enough common time instants are found, successful key guess is possible after profiling.



(a) (b) (c)

Figure 4.9: Attack results for the first byte of $S(x \oplus k)$ with Method$_{\mathbf{CPA}}$ using 50 test traces, and m = (a) 5, (b) 15, and (c) 35. Successful attack is shown in white.

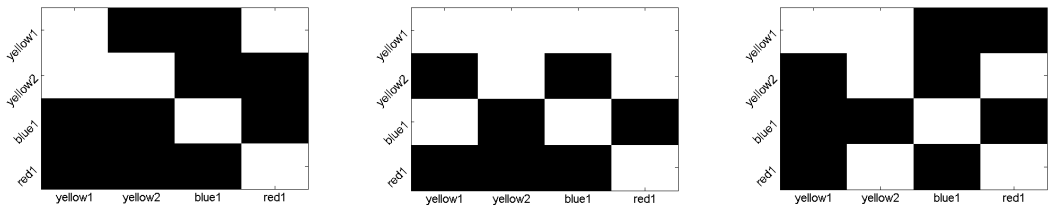


(a) (b) (c)

Figure 4.10: Attack results for the first byte of $S(x \oplus k)$ with Method$_{\mathbf{CPA}}$ using 500 test traces, and m = (a) 5, (b) 15, and (c) 35. Successful attack is shown in white.

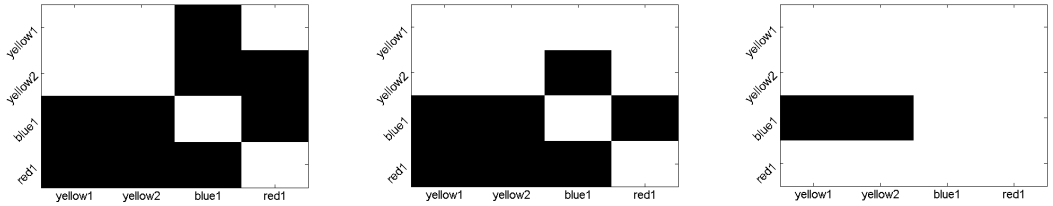
Each microcontroller was used to train and test all remaining chips for the first byte of the $S(x \oplus k)$ function. There was also a unique key used for encryptions on each microcontroller. Results with Method$_{\mathbf{CPA}}$ for 50 and 500 test traces are shown in Figures 4.9 and 4.10 respectively and those for Method$_{\mathbf{R_a^2}}$ are shown in Figures 4.11 and 4.12 respectively. A light square indicates a correct key guess while a dark square is an incorrect guess. The test cases are shown when m=5, 15 and 35 time samples are used in the distribution estimate. N_1, N_2=4000 training traces were used for all tests. Figures show failures when *Blue1* was used against other microcontrollers, which are overcome to some extent when m increases. Thus linear regression based attacks are sensitive to differences

in hardware configurations but this can be overcome by including more time samples of interest in the training data.



(a) (b) (c)

Figure 4.11: Attack results for the first byte of $S(x \oplus k)$ with Method$_{\mathbf{R_a^2}}$using 50 test traces, and m = (a) 5, (b) 15, and (c) 35. Successful attack is shown in white.



(a) (b) (c)

Figure 4.12: Attack results for the first byte of $S(x \oplus k)$ with Method$_{\mathbf{R_a^2}}$using 500 test traces, and m = (a) 5, (b) 15, and (c) 35. Successful attack is shown in white.

In general, increasing the number of time samples m resulted in better performance. This is because more time samples were found that were in common between hardware architectures. This has been shown to be true for template attacks as well [84]. Table 4.5 shows the number of time samples in common between microcontrollers by choosing time samples identified by Method$_{\mathbf{CPA}}$from each. 4000 training traces for profiling and m=35 were used in each case. *Blue1* has the least number of time samples in common with the other microcontrollers while *Yellow1* and *Yellow2* have the most. This directly corresponds

to the results where *Blue1* was the hardest to attack by profiling other microcontrollers, while *Yellow1* and *Yellow2* were easiest to attack with each other.

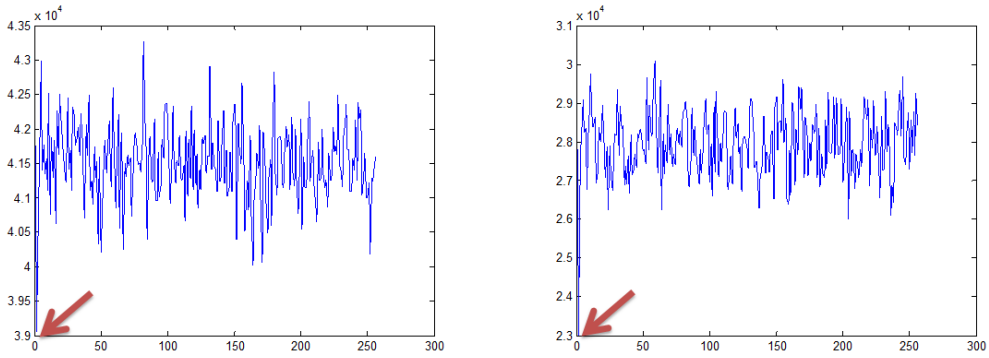
Table 4.5: Number of common time samples between different microcontrollers with m=35 and 4000 training traces.

	Yellow1	*Yellow2*	*Blue1*	*Red1*
Yellow1	35	28	10	20
Yellow2	28	35	7	21
Blue1	10	7	35	8
Red1	20	21	8	35

In certain cases, simply increasing the number of points does not always guarantee a better guess. In Figure 4.9, when m is increased from m=5 to 15 with $\text{Method}_{\mathbf{CPA}}$, performance actually decreases as indicated by a decrease in the number of light success squares. This is because as m increases, more time samples that are not in common between the test and training microcontroller are being included in the distribution estimate for the test microcontroller which degrades key prediction performance.

4.6.2 Profiling and Testing with Different Probes

To define the envelope of success for the the Linear Regression Attack, it was tested against data collected with two different probes: a RISCURE near-field EM probe [110] and a Rohde & Schwarz RS H 50-1 ring-type near-field EM probe [113]. Data from each probe was used to train and attack test traces collected from the other probe, using the same microcontroller. $\text{Method}_{\mathbf{R}_\mathbf{a}^2}$ was used in each case and the first byte of $S(x \oplus k)$ is investigated. Results were based on the 500 test traces and N_1, N_2 =4000 training traces for key-byte 2. Figure 4.13 shows the mean absolute value of the exponent of 256 possible hypotheses of the first key-byte from (4.15). The best key guess would show as the maximum value. The correct key value, highlighted by arrows is clearly the maximum value in the plot and



(a) (b)

Figure 4.13: (a) Attack results with Method$_{\mathbf{R_a^2}}$using loop probe training data against Riscure probe test data, and (b) results using Riscure probe training data against loop probe test data. Key byte guess highlighted by arrow is correct in both cases.

thus is correct. These results show the versatility of this attack, where the profiling of the probability distribution with the training data is accurate enough to be effective even when the equipment used to collect the test data is different.

4.6.3 Linear Regression Attack with Method$_{\mathbf{R_a^2}}$in Noisy Environments

In Section 4.5, it is noted that the coefficient of determination R^2 is the square of the Pearson coefficient of correlation used in CPA. As a result, the intensity plot of variable importance in Figure 4.7a shows many similar time samples with high importance for both of these metrics. We have shown that R_a^2 will suppress less important variables more than more important ones. This suggests that R_a^2 might perform better with noisy data than traditional CPA. To test this hypothesis, White Gaussian Noise (WGN) was added to the test data to achieve a desired Signal-to-Noise Ratio (SNR); the Linear Regression Attack was performed on this data using Method$_{\mathbf{CPA}}$and Method$_{\mathbf{R_a^2}}$to choose time instances of N_1, N_2 =4000 training traces and 500 test traces with m=15. Results in Table 4.6 show that the attack on key-byte 1 using Method$_{\mathbf{R_a^2}}$performs better in general for test data with

WGN, succeeding with fewer test traces and more noise than the attack with $\text{Method}_{\mathbf{CPA}}$ on the same data.

Table 4.6: Performance of Linear Regression Attack when selecting points using $\text{Method}_{\mathbf{CPA}}$ and $\text{Method}_{\mathbf{R_a^2}}$ on 500 test traces with WGN

SNR (dB)	Attack Type	100 Test Traces	50 Test Traces	30 Test Traces	15 Test Traces
100	$\text{Method}_{\mathbf{CPA}}$	Success	Success	Fail	Fail
100	$\text{Method}_{\mathbf{R_a^2}}$	Success	Success	Success	Success
25	$\text{Method}_{\mathbf{CPA}}$	Success	Success	Fail	Fail
25	$\text{Method}_{\mathbf{R_a^2}}$	Success	Success	Success	Fail
20	$\text{Method}_{\mathbf{CPA}}$	Success	Fail	Fail	Fail
20	$\text{Method}_{\mathbf{R_a^2}}$	Success	Success	Fail	Fail
15	$\text{Method}_{\mathbf{CPA}}$	Fail	Fail	Fail	Fail
15	$\text{Method}_{\mathbf{R_a^2}}$	Success	Success	Fail	Fail
5	$\text{Method}_{\mathbf{CPA}}$	Fail	Fail	Fail	Fail
5	$\text{Method}_{\mathbf{R_a^2}}$	Fail	Fail	Fail	Fail

4.7 Conclusions

Linear Regression Attacks allow the leakage function to be estimated rather than assumed, and this allows for a more tailored attack on a device that might not strictly follow leakage models such as the HW model. This work provides two main areas of contribution: statistical analysis of linear regression estimators, and practical experiments using Linear Regression Attack including a comparison between methods.

Statistical analysis of the linear regression is conducted in the training phase, where four different methods from literature and one new method were used to choose important

time instants where there is information leakage. Analysis revealed that including the intercept greatly improved prediction capability. In addition, it was shown that poor predictors could still guess the correct key but at a cost of larger required training and test sets.

In the comparison of attack performance with these five methods, $\text{Method}_{\mathbf{CPA}}$, $\text{Method}_{\mathbf{R_a^2}}$ and $\text{Method}_{\mathbf{R^2}}$ were found to outperform $\text{Method}_{\mathbf{sym}}$ and $\text{Method}_{\mathbf{||b||}}$ in most cases, with the difference in performance decreasing as number of training traces increased. Increasing the dimensionality of the probability distribution in the training phase from m=1 to m=20 resulted in an order of magnitude improvement in performance. Comparison of variable importance determined by the five methods show good overlap, with $\text{Method}_{\mathbf{CPA}}$, $\text{Method}_{\mathbf{R_a^2}}$ and $\text{Method}_{\mathbf{R^2}}$ giving higher weighting to time instants with better regression, as determined by their R_a^2 metric. In practical experiments, attack performance was better on hardware families with more time instants in common. Linear Regression Attacks were successful even when test and training traces were attained with different data collection hardware. Finally, $\text{Method}_{\mathbf{R_a^2}}$ performed better than $\text{Method}_{\mathbf{CPA}}$ in noisy environments, and was able to successfully guess the correct key-byte with as little as 50 test traces and SNR=15 dB.

V. Results: Random Forest SCA Application

This chapter presents the first application of the Random Forest classifier to the field of SCA. Research results are presented from the article in 'Random Forest Profiling Attack on Advanced Encryption Standard' as accepted to appear in the *International Journal of Applied Cryptography* [91]. The background section is shortened here to reduce redundancy with the other chapters in this document. This is followed by an investigation into the nature of the non-Gaussian distributed variables captured from a PIC device and their effects on CEMA attacks.

5.1 Introduction

Modern cryptographic algorithms such as the AES protect data using a mathematically rigorous process that effectively encrypts the sensitive plaintext information. The security of encryption algorithms like AES lie in their secret key which is usually stored within the cryptographic *black box* hardware which itself is assumed to be secure. Side Channel Analysis (SCA) is the study of physical characteristics of the algorithm implementation that tests this assumption. By monitoring various side-channels during the encryption or decryption process, side-channel attacks can effectively deduce the secret key without the need for invasive procedures such as chip depackaging to image and read device memory [2, 54, 75].

Profiling attacks on AES model the side-channel on a known training device when it is executing a key-dependent Intermediate Value (IV) calculation. The trained model can be used against devices of the same or similar model type to classify the unknown IV of a test device and thereby derive the corresponding unknown cryptographic key, assuming the leakage function is the same between devices.

The most popular profiling attack used today is the *Template Attack* which characterizes a training device using Bayesian Maximum Likelihood (ML) for a multivariate Gaussian distribution. This is an optimal classification assuming the distribution is in fact Gaussian and that the distribution function can be exactly derived with the training data [125]. In practice, the exact distribution of the side-channel is not known and uncertainty exists about its true dimensionality. In addition, the noise is the side-channel is not always Gaussian. Regardless, the assumption of a multivariate Gaussian distribution has served SCA well. Template Attacks have been shown to be effective against Atmel [103] and PIC [84] microcontrollers, ASIC circuits [72], and FPGAs [52]. In addition to Template Attacks, Stochastic Attacks [115] have used the multivariate Gaussian noise distribution assumption to successfully profile and attack microcontrollers. Although not a profiling attack, Mutual Information Analysis (MIA) is studied in [97] with the Data Encryption Standard (DES) on a 8 bit smart card as well as on a SecMat V3/2 ASIC testbed [43]. MIA was found to be successful with few traces using a parametric estimation.

In all of these works, a Gaussian distribution of the side-channel for a chosen IV is assumed and estimated from the collected data. In addition, the actual side-channel leakage of the IV (signal) is assumed to be fixed and the only randomness in the measurements entirely due to the noise. However, if this noise is non-Gaussian, then non-parametric classifiers would perform better in these situations. Recent work has sought alternative classifiers to ML used in Template Attacks. In [66], Random Forest, Support Vector Machine (SVM) and Self Organizing Maps (SOM) attack individual bit values in the Triple Data Encryption Standard (3DES) algorithm on an FPGA. Authors show varying levels of success per bit ranging from 50% to 97% correct classification rate. When attacking the individual bits of a single byte, the correct classification rates using Random Forest with Principal Component Analysis (PCA) were found to be between 5.80% to 15.33% more successful than using Template Attack with the minimum Redundancy maximum

Relevance (mRMR) filter. In [55], the Least Squares SVM (LS-SVM) was used with a Radial Basis Function (RBF) for non-linear classification by converting the original data set to higher dimensional space. Two-class problems were investigated consisting of detecting singular Hamming Weight (HW) and particular bit values. Three feature selection methods were used: Pearson Correlation, Sum Of Squared pairwise t-differences (SOST) and PCA. In all cases Template Attack either matched or outperformed LS-SVM. Better results were achieved in [51], where the Hamming Weight was classified successfully with SVM, requiring fewer attack traces with moderate to high noise.

These results lead to some obvious questions in this research. First, for a trace collected over a full encryption only a small time segment signal actually contains the leakage information for an IV. Furthermore since the time samples are the input variables to a classifier and the exact dimensions of data-dependent distribution are not known, can a successful attack still be accomplished? Random Forest is shown to be more tolerant to the "curse of dimensionality" than Template Attack, allowing greater success for very large dimensional data sets [22]. Second, how can the dimensions be correctly identified to enhance classification performance? SOST has been shown to significantly improve Template Attack performance by identifying time samples where information leakage occurs. Random Forest offers an alternative variable importance metric that allows identification of dimensions with the most discriminatory information for classification. These methods are tested in this research in dimensionality reduction experiments. Third, even given a close approximation of the correct dimensions, is the underlying distribution of the side-channel noise Gaussian for a chosen IV? And finally, if the distribution of the side-channel noise is Gaussian, do parametric attacks provide better results?

5.2 Background

In this section, background information is provided on side-channel leakage, Template Attack, Random Forest and SOST method of variable selection.

5.2.1 Side Channel Leakage

Side-channel leakage is any unintentionally emitted information from a physical device which is dependent on the data being processed by that device. For the case of a microcontroller running an encryption operation, side-channel leakage can be observed via frequency [41], time [62], Electro-Magnetic (EM) emanations [2] and the power consumption [61] of the device during an operation. All side-channel collected data can be modeled as

$$\mathcal{X} = d + n, \tag{5.1}$$

where d is the data dependent part generated by the device performing an operation (signal) and n is the non-data dependent part generated either intrinsically or by ambient conditions (noise) [24]. This research considers a microcontroller running an AES-128 encryption and the EM side-channel as the side-channel of interest.

5.2.2 Template Attack

Template Attacks profile the side-channel of a device as it is performing a key-dependent operation ϕ [24, 42]. For AES, ϕ can be any intermediate value calculation during the encryption process. Let S be the image of possible intermediate values generated by function ϕ. Profiling attacks such as Template Attack learn the leakage models for each value of $S=\{S_1, \ldots, S_c\}$, given a set of input vectors, thus converting the attack to a c-class classification problem. Commonly used methods for profiling attacks include learning the leakage models for the individual byte values (c=256) [24, 42, 84], Hamming Weight or Hamming Distance (c=9) [51], or bit values ($c = 2$) [55, 66] of S. For this research, byte-wise attacks are considered, which profile the side-channel leakage for each byte value of S, resulting in c=256 distinct values of S. If the M-dimensional input vector $\mathbf{x}$ is collected while the microcontroller is calculating $S_j \in S$, then the parametric Template

Attack assumes a multivariate Gaussian distribution for $p(\mathbf{x}|S_j)$ or

$$p\left(\mathbf{x}|S_j\right) = \frac{1}{(2\pi)^{\frac{M}{2}}|\Sigma_{S_j}|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu_{S_j}\right)^T \Sigma_{S_j}^{-1}\left(\mathbf{x}-\mu_{S_j}\right)\right\}. \tag{5.2}$$

The M-dimensional covariance matrix Σ_{S_j} and mean vector μ_{S_j} for each class S_j are estimated from the training data. Thus, 256 such *templates* of Σ_{S_j} and μ_{S_j} are built for byte-wise Template Attacks with the traces from the training device. One limitation of this method is that to invert Σ_{S_j}, the number of dimensions M must be less than or equal to the number of observations n_{S_j} in class S_j. To overcome this limitation, a pooled covariance assumption can be made where all classes share the same covariance matrix, which is the method used in this research, unless stated otherwise.

For the test device, the test traces $\mathbf{x}^*$ are used with the templates to determine the intermediate value S_j^* of the unknown key k^* using the Bayes rule

$$S_j^* = \arg\max_{S_j \in S} \frac{p(\mathbf{x}^*|S_j)P(S_j)}{p(\mathbf{x}^*)}, \tag{5.3}$$

where $P(S_j)$ is the prior probability of the intermediate value S_j and $p(\mathbf{x}^*)$ is given by

$$p(\mathbf{x}^*) = \sum_{j=1}^{256} p(\mathbf{x}^*|S_j)P(S_j). \tag{5.4}$$

The class with the highest probability using the estimated *template* distributions from the training phase becomes the class guess S_j^*. Since S_j^* for a known plaintext p has a one-to-one relationship to a unique key value k_j^*, the unknown key value k^* can be recovered by inverting ϕ which is provided through the AES decryption functions. In this way $p(\mathbf{x}^*|k_j^*)$ can be recovered from $p(\mathbf{x}^*|S_j^*)$. For the test data, a valid assumption is that the unknown key k^* is constant for all test traces. Probabilities over n traces can then be combined using maximum likelihood principle [72]

$$p(k_j^*|\mathbf{x}^*) = \arg\max_{k_j} \frac{\prod_{j=1}^{n} p(\mathbf{x}^*|k_j)\ P(k_j)}{\sum_{i=1}^{256}\left(\prod_{j=1}^{n} p(\mathbf{x}^*|k_j)\right)\ P(k_j)}. \tag{5.5}$$

5.2.3 Random Forest

The method of Bayes classification with equal prior probabilities, which is used in Template Attack, is optimal for classification provided the probability distributions $p(\mathbf{x}|S), \forall(S)$ are *exactly* known [125]. Also, there is the strong assumption that these probability distributions are Gaussian. This must be true for optimality to hold for Template Attack. When these conditions are not met, non-parametric classifiers such as Random Forest can be advantageous. Random Forest is an ensemble classifier introduced in [17], consisting of a group of decision trees which recursively partition a M-variable space. Each node of the decision tree splits a set of observations into two child nodes based on several possible criteria such as entropy reduction or information gain from the split [15], Gini impurity reduction [100] and Area-Under the Curve (AUC) [39]. We use entropy reduction for computational efficiency and high performance. At each node in a decision tree, a subset of $m \leq M$ randomly selected variables is investigated and for each a threshold is found that divides the data into two child nodes. For a c-class problem, the probability of class $i \in \{1, \ldots, c\}$ at node a is represented as $p(i|a)$ and is efficiently calculated by counting the number of observations for class i in a group of data. Then the impurity at node a can be calculated by finding the Shannon Entropy [125] in

$$H(a) = -\sum_{i=1}^{c} p(i|a)\log_2 p(i|a). \tag{5.6}$$

The best variable and best threshold are determined as the variable-threshold combination (v, t) that gives the highest information gain or

$$I(a; (v, t)) = H(a) - H(leftChild|(v, t)) - H(rightChild|(v, t)), \tag{5.7}$$

where a is the parent node, and *rightChild* and *leftChild* are the child nodes. With this definition of impurity, a *pure* node is one with minimum entropy with all observations from a single class.

The recursive algorithm for growing a tree is shown in Algorithm 5. Assume training set *Data* of N observations, each of $M-$dimensionals.

Algorithm 5 Decision Tree Node Splitting Recursive Algorithm

```
SplitNode (Data, Impurity)
if Impurity == minimum Impurity then
   return
else
   Calculate parent impurity H(a)
   for each variable v_i do
      for each possible value t_j do
         v_i.leftChild = {Data > t_j}
         v_i.rightChild = {Data ≤ t_j}
         Calculate v_i.leftChild impurity H(leftChild|(v_i, t_j))
         Calculate v_i.rightChild impurity H(rightChild|(v_i, t_j))
         Calculate impurity reduction for (v_i, t_j)
      end for
   end for
   BestNode = max( Impurity reduction for all variables and all thresholds )
   SplitNode( BestNode.leftChild, BestNode, H(leftChild) )
   SplitNode( BestNode.rightChild, BestNode, H(leftChild) )
end if
```

Decision Trees are grown on different randomly chosen data sets to decorrelate them from each other. In this way, each decision tree is grown on a slightly different data set, giving each a unique view of the variable space. Trees are grown to full length where all terminal leaves are pure, in order to minimize both bias and between-tree correlation. In the test phase, an observation is classified by each trained tree to reach a classification decision. Final class decision of the forest is reached by majority vote among the ensemble. As Random Forest determines a test class by voting among the trees of the forest, the probability of each class can be calculated by $p(\mathbf{x}|S_j) = v_{S_j}/N_t$ where v_{S_j} are the number of votes for class S_j and N_t are the total number of trees in the forest (i.e. total number of votes). Then (5.5) can combine probabilities from multiple test traces in a similar manner to Template Attacks.

The Random Forest classifier includes a built-in variable importance metric. For each node of each tree in the Random Forest, the reduction in entropy from the splitting of parent to children over a variable is recorded. The entropy reduction for each variable x_i is averaged over all nodes and all trees to give an entropy based variable importance metric for each variable. The Random Forest Entropy Importance method is abbreviated as *RFEI* from here on.

The number of trees is a key parameter in the Random Forest classifier. As trees are grown to minimize correlation, variation is expected in the tree class votes and a large enough forest is required to ensure the majority vote for the correct class can be attained. A forest that is too large increases computation time but is generally not harmful as Random Forests are largely immune to overtraining [125]. Traces from a PIC microcontroller were analyzed on a Random Forest of varying size to determine the proper forest size for this data. Test and training data were taken from the same device and the best 25 features as determined by RFEI were used. Figure 5.1 shows that training correct classification rate rises to 100% in as few as 12 trees, because trees are grown to full length. However, the classifier is not overfitted as more trees are added and test correct classification rate continues to rise until a steady state is reached after 600 trees. As results are expected to be more difficult when the training and test devices are different, a forest size of 1000 trees was chosen for all experiments.

Random Forest has gained popularity due to its non-parametric and non-linear properties, high performance in high dimensional data sets and ease of implementation. It has found application in classifications of genes [31], spectral data [74], geographical landmarks [85] and agriculture [89]. In empirical studies Random Forest has outperformed Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and Boosted Decision Trees in large dimensional data sets [85, 89, 109]. In [88], Random Forest was found

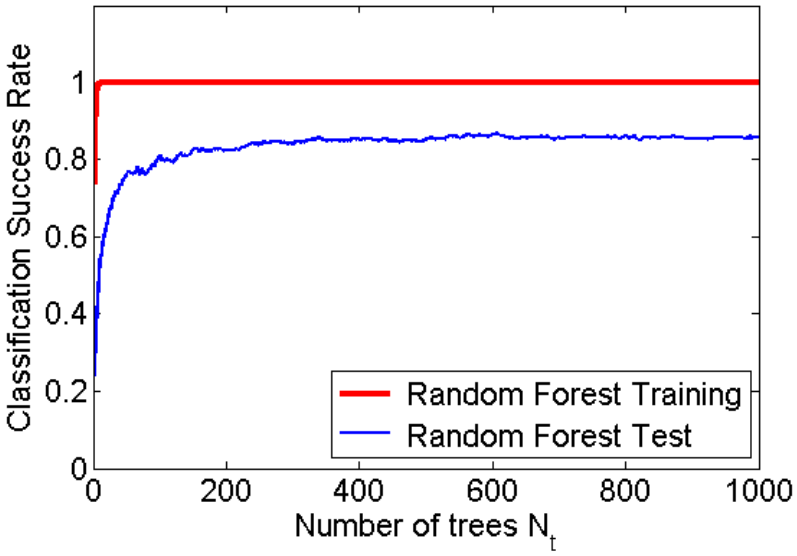


Figure 5.1: Training and test errors for Random Forest as size increases. Template Attack performance is plotted for reference.

to provide good posterior probability results when compared to SVM, Logistic Regression, KNN, Artificial Neural Networks and Naive Bayes classifiers.

Although the true convergence theory behind Random Forest is a subject of active research [13, 70], a few conclusions may be derived directly from its implementation. First, each tree is grown on a different data set, thereby reducing the influence of data outliers. Next, a random sampling of variables is analyzed at each node of each tree in the forest. From each sampling the variable most useful in classification is used for partitioning the data further. This variable filtering reduces the influence of non-data dependent variables and ensures that more useful variables are used in the classification, as mathematically proven in [12]. Random Forest determines decision regions strictly from the data alone without any assumption of an underlying distribution. In addition, as the decision space is split at each node, non-linear decision regions can be determined.

5.2.4 Sum-Of-Squared pairwise T-difference

Gierlichs et al. in [42] first used SOST as a means of identifying points in time where there is information leakage to aid in Template Attacks. For data from two sets (i, j), the t-Test takes into account their mean (m_i, m_j) and their variance (σ_i^2, σ_j^2) in relation to the number of observations from each set (n_i, n_j). Authors modified the t-Test implementation to provide the sum of squared pairwise t-differences, which was termed SOST, and given by [42]

$$\sum_{i,j=1}^{K} \left(\frac{m_i - m_j}{\sqrt{\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}}} \right)^2 \text{ for } i \geq j. \tag{5.8}$$

In this way, the SOST is calculated for each pairwise combination of classes. Regions of interest with high SOST values indicate where there is information leakage. SOST was found to greatly aid Template Attack in [42], where success rate improved from 23% to 100% when using SOST to identify important variables with 5000 training traces.

5.3 Data Collection and Analysis

5.3.1 Data Collection

Table 5.1: PIC device families and their part numbers.

Family	Device Numbers	Part Number
A	A1-A10	PIC24FJ64GA102 I/SP
B	B1-B10	PIC24FJ64GA002 I/SP
C	C1-C10	PIC24FJ48GA002 I/SP
D	D1-D10	PIC24FJ32GA002 I/SP

Experiments were conducted on four families of PIC microcontrollers from Microchip Technology Inc. Their model numbers along with our naming convention is shown in Table 5.1. Ten chips from each family were used for experiments. Devices from family A

differ from the remaining device families by including a number of on-board peripherals not present on the other device families. Families A and B have 64 KB of program memory, while families C and D have 48 KB and 32 KB respectively. A study of the variance of traces collected from each family in [84] showed that family A differed significantly from the remaining families and resulted in poorer cross-device Template Attack performance when using family A to attack devices from the remaining families and vice versa.

The EM measurements during the AES-128 encryption operation are the side-channel of interest for all experiments. Traces were initially collected at a 2.5GHz sampling rate with a 1 GHz in-line low pass filter on a Lecroy 104-Xi-A digital oscilloscope with a Riscure broadband EM probe [110]. Traces were then filtered with a 100 MHz 8th order Chebychev digital filter and downsampled to 250 MHz, and normalized [84]. This new equivalent sampling frequency is still well above the PIC instruction clock frequency of 14.74 MHz, thus meeting Nyquist sampling criteria. This yielded the highest correlation values in a correlation attack and outperformed direct sampling at 250 MHz. An Agilent E3631A power supply was used to minimize power fluctuations to the EM probe. Normalizing each time sample to a zero mean and unit variance was shown to provide superior results for cross device attacks in [84].

The different microcontrollers on the test board are spatially aligned by using a custom designed jig that can be fixed to the microcontroller board relative to the probe. The collected traces are temporally aligned by using a trigger signal that is sent by the microcontroller before the start of the encryption, which triggers the oscilloscope collection. Post-processing cross-correlation is used to ensure that the collected traces are well aligned. In order to minimize noise from external sources, at the time of collection, only the AES algorithm is running on the microcontroller. Serial communication with the collection computer is halted during the encryption process. 5000 training traces and 500 test traces were collected from each device. The best location on the chip for the EM

collection was determined by performing an X-Y scan, where the surface of the chip was divided into a grid of multiple sub-regions and a trace was collected at each region. The region with the highest intensity in the spectral band from DC-30 MHz was chosen to be the best location for data collection. This region was fixed for all devices used in this work.

5.3.2 Data Analysis

The collected traces are in the time domain, and as such the variables input to the classifier are the data collected at the given time samples. For IV operations that span multiple time samples, the side-channel leakage should theoretically be the same and so the variance between these variables will be entirely due to the noise. If the noise during the encryption process spans multiple time samples, then it can be modeled with a multivariate distribution. Template Attack hypothesizes that this multivariate distribution is Gaussian. However, distribution analysis $p(\mathbf{x}_i)$ of the variables in our data shows multiple variables exhibiting bimodal distribution shapes as shown in Figure 5.2. The figure also shows 2-variable distribution plots of $p(\mathbf{x}_{i,j})$ exhibiting distinct *multivariate* non-Gaussian shapes. Many more variables were found to show this bimodal distribution; only these arbitrarily chosen variables are shown here for clarity. A Kolmogorov-Smirnoff test (KS-test) for normality was used to determine how closely variables resembled a standard normal distribution. This test has the null hypothesis that a variable is normally distributed for a given significance level α. A very low α value of $1e-20$ was chosen to identify those variables that were clearly not normally distributed. It was found that this low α value when used with the KS-test distinguished those variables that clearly displayed the bimodal distribution seen in Figure 5.2. This method will be used in subsequent tests in this work to distinguish Gaussian and non-Gaussian variables. 32,102 of a total 50,000 variables rejected the null hypothesis of normality at this α level, and clearly showed a bimodal distribution. Classical profiling attack theory assumes that the side-channel leakage of the data-dependent signal d_{S_j} in (5.1) is constant when processing data S_j, and that variation

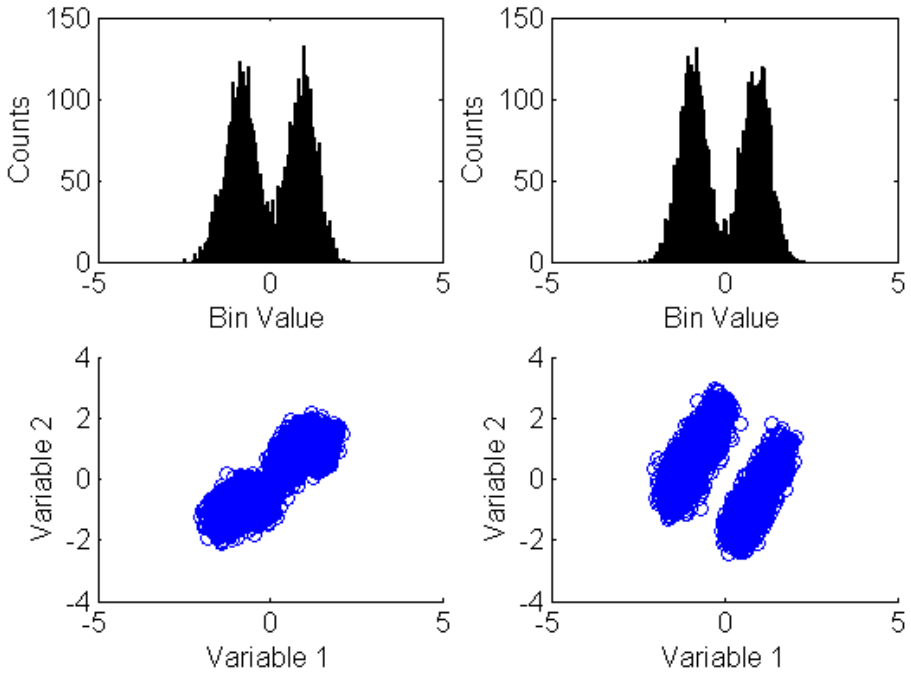
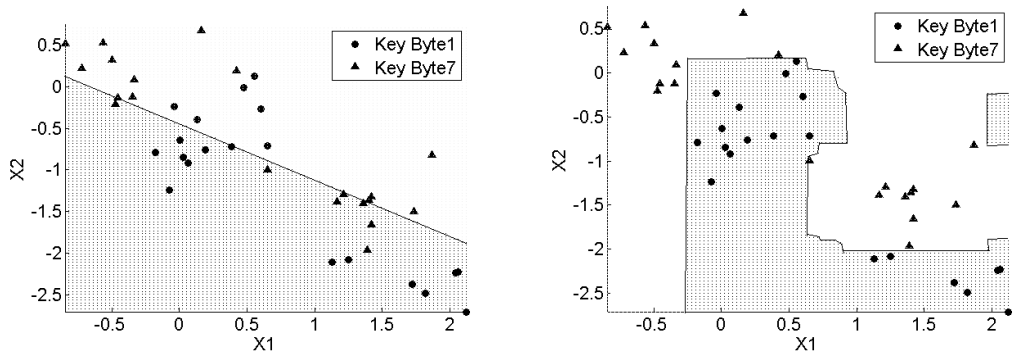


Figure 5.2: 1-D and 2-D distributions for an arbitrarily chosen sample set of two PIC data variables. KS-test for normality revealed 32,102 out of 50,000 total variables exhibited similar distribution shape.

in measurement for a class S_j comes from multivariate Gaussian noise centered at d_{S_j} [72]. However, we show here that in certain cases such as ours, this noise can be non-Gaussian.

Figure 5.3 shows the arbitrarily chosen 2-variable space of two key byte values 1 and 7. Only two variables and two classes are shown for clarity. The figure shows the bimodal nature of variables in Figure 5.2 is also clearly present for $p(\mathbf{x}|Byte_1)$ and $p(\mathbf{x}|Byte_7)$. This behavior has been verified to exist with each of the 256 classes in our byte-wise attack, as well as with other variables. Larger shaped dots in bold are the locations of the training data used to generate the models. Light and dark shaded regions in the 2-variable space show the classification regions assigned by each respective classifier, i.e., they show how the classifier would partition the 2-variable space given the training set. At the time of



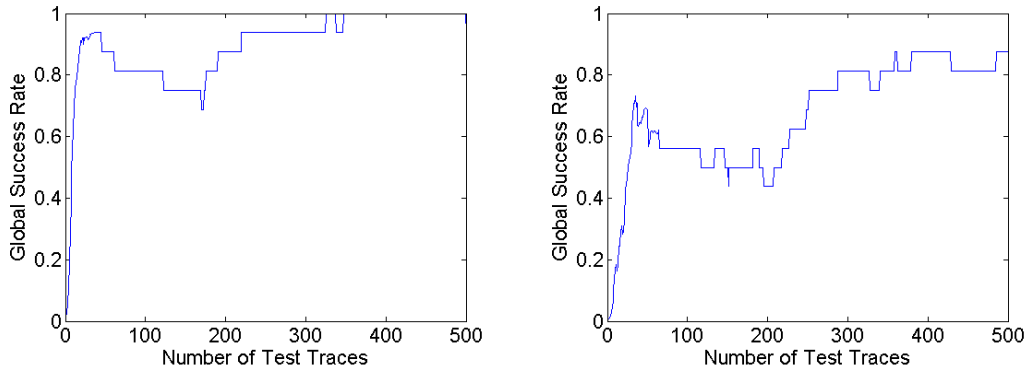
(a) (b)

Figure 5.3: Arbitrarily chosen two variable decision space for two IV byte values as determined by (a) Template Attack, and (b) Random Forest. Light and dark shaded regions represent the decision regions determined by each classifier, and bold shapes represent the training observations used to determine the decision regions.

collection, no other functions are running concurrently with AES, and so the cause of the non-Gaussian noise is not certain. Regardless, these non-Gaussian properties were witnessed for the collections of 40 PIC microcontrollers, demonstrating that non-Gaussian noise can exist in side-channel collected data.

5.3.3 Variable Importance and Dimensionality Reduction

In this research the large number of time samples collected for each trace lead to the daunting task of identifying variables of interest. A common challenge with SCA is not knowing the true dimensions of the distribution, i.e., the time samples where the sensitive information leakage occurs. As discussed in Section 5.2.3, Random Forest has a built-in variable filtering feature that uses only the best of an m variable subsample at each decision tree node. It has shown good performance even when the number of dimensions is much higher than the number of observations. Assuming no prior knowledge of the underlying encryption algorithm implementation, Random Forest was used with full

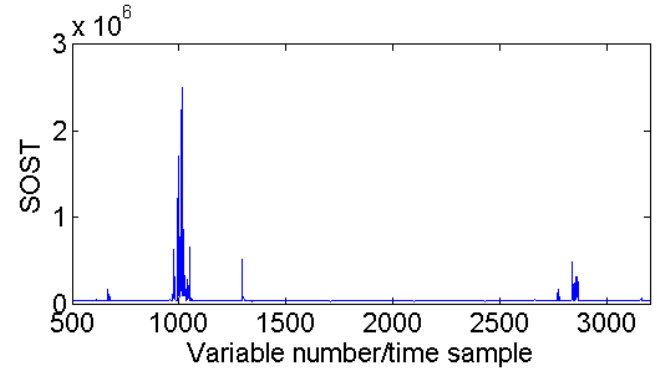


(a) (b)

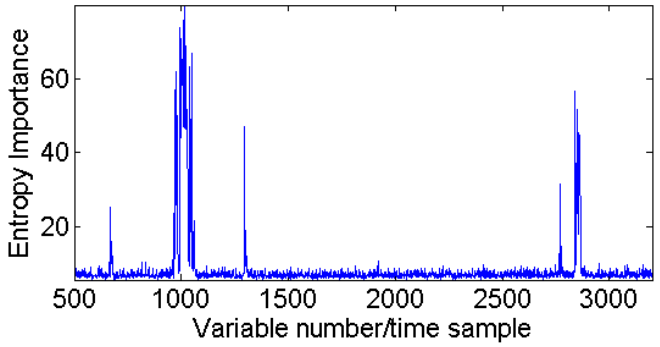
Figure 5.4: Global Success Rate with Random Forest using full 50,000 variable traces with no variable reduction: (a): same training and test device, (b) different training and test device, but from the same family.

dimensional traces without variable reduction. When training and test traces were from the same device all 16 AES key bytes were correctly identified. When training and test devices were different but from the same family (cross-device attack), 15/16 AES key bytes were successfully found. Global Success Rate (GSR) as defined in [120] is used in Figure 5.4 to show performance as a function of number of test traces required for successful attack. Random Forest performance has been found to improve when less important variables are removed from the training set [5], and so dimensionality reduction should be investigated.

Principal Component Analysis (PCA) has been used successfully as means for feature selection and reducing redundant information and has shown good success with side-channel attacks [7, 11, 84]. Another commonly used method is to use CPA with a power model such as HW, i.e. HW-CPA [72, 84, 92]. Variables useful in profiling attacks have relatively high correlation using CPA, provided the device follows the leakage model.



(a)



(b)

Figure 5.5: Variable importance as determined by two methods: (a) SOST, and (b) RFEI.

In this work, the use of RFEI and SOST was examined to identify variables of interest. Random Forests have widely been used with variable importance metrics with good performance [17, 32]. Similarly, Template Attack has shown improvement with SOST identified variables in [42]. Figure 5.5a shows the SOST Importance for variables 500-3200, and Figure 5.5b shows the RFEI value for these variables. Magnitude of the variables on the y-axes are different, but can be ignored as these plots are meant to show *relative* variable importance, i.e. how important variables are in context to each other. Both methods identify similar variables as important, but differ in the importance they assign to the variables. In particular, RFEI assigns significantly higher importance than SOST to the

Table 5.2: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack for the first 3200 variables.

Train Device	Test Device			
	A	B	C	D
A	100.00	53.13	54.13	51.37
B	31.50	99.94	100.00	100.00
C	37.44	100.00	99.94	100.00
D	36.56	100.00	100.00	100.00

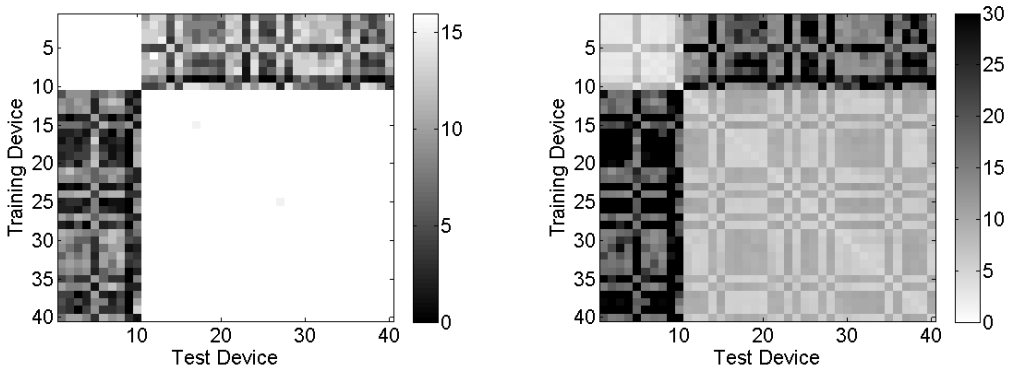
variables around time samples 650, 1300 and 2800. Thus the top ranked variables identified by each method will differ and have a corresponding effect on attack performance. For this work, the top ranked RFEI and SOST variables will be used for dimensionality reduction.

5.4 Results

5.4.1 Profiling Attack Performance Without Variable Reduction

Less than 1% of the time samples collected from the AES leakage traces are useful in classifying the output of the SBox for the first round for any given key-byte, which is the operation chosen to attack. In many practical scenarios, the attacker may not have the luxury of knowing the exact encryption algorithm implementation used by a device. Therefore, even if the general leakage time-frame can be guessed, a key challenge for the attacker is knowing the exact time when the operation of interest is taking place. For Template Attack using probability estimation, high-dimensional classification is only possible by significantly increasing the number of training traces to keep the covariance matrix in (5.2) non-singular.

In Section 5.3.3, Random Forest successfully classified AES key bytes without any variable reduction. Further expanding the testing to larger cross-device experiments, an



(a) (b)

Figure 5.6: Average results over 100 iterations from Random Forest profiling attack on AES-128 for the first 3200 variables with no variable reduction. (a) shows the number of correctly classified key bytes, (b) shows the minimum number of test traces required for the correctly guessed key bytes to achieve 90% posterior probability.

assumption was made that the SBox output of the first round for all bytes was generated in the first 3200 time samples. This is the largest number of dimensions that could practically be used for probability estimation with 5000 training traces even assuming a pooled covariance matrix for the variables. Experiments were conducted by training each of 40 PIC microcontrollers from 4 different families and attacking the remaining devices. Thirty test traces were drawn randomly from a pool of 500 and used to determine a key guess. This sampling was repeated 100 times to gain confidence in the results. A key-guess was considered a success if the posterior probability of the true class was greater than 90% after 30 test traces. Figure 5.6a shows an intensity plot of the number of correctly guessed key bytes out of 16 for each combination of test and training device. The first 10 devices are from device family A, followed by 10 devices each from families B, C and D. As stated in Section 5.3.1, device family A differs the most in hardware peripherals than the remaining families. As a result, classification is poorer when devices from family A are used against

those from other families. Table 5.2 shows the average number of correctly guessed key bytes out of 16 for each device family combination. Figure 5.6b shows minimum number of test traces required for the correctly guessed key bytes. For cases where no key-bytes were successfully guessed, minimum number of test traces is set to 30 which is the sample size. It is apparent that for Test-Training device combinations with high numbers of correctly guessed key-bytes, very few test traces are required. We note that these results are with a very high 3200-dimensional input space.

Template Attack with probability estimation was not able to correctly identify any key guesses for any combination of devices and hence results are not presented here.

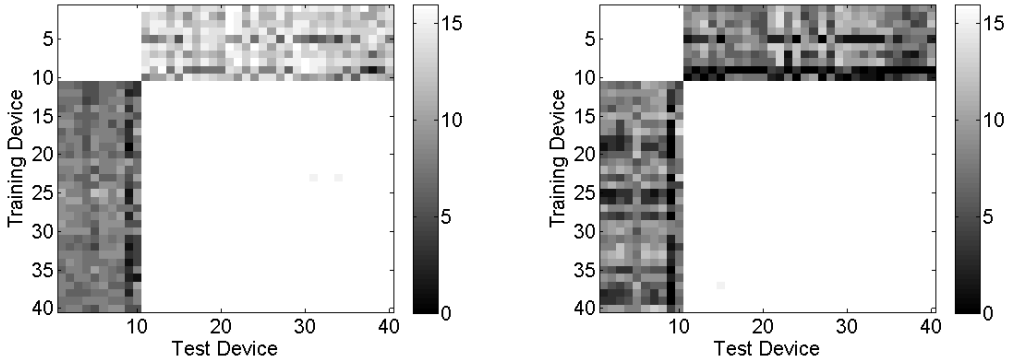
5.4.2 Profiling Attack Performance With RFEI Variable Reduction

Results improve for both Template Attack and Random Forest when selecting only those variables where there is information leakage, shown in Figure 5.7. The best 25 variables were chosen for each training microcontroller with RFEI. The cross-device attack with both classifiers was repeated on these reduced sets of variables. Again, 30 test traces were chosen randomly from a set of 500 traces to attack a key byte, and this was repeated with replacement 100 times to gain confidence. Table 5.4 shows the average success rate of correctly guessed key bytes out of 16 for each training and test family combination. The results show near-perfect performance for both Template Attack and Random Forest when families B, C and D were used against each other, similar to Figure 5.6. Similar correct key success rate is achieved by Random Forest and Template Attack when devices from families B, C and D (train) are used to attack devices from family A (test). Differences in performance are much more apparent when devices from family A are used against families B, C and D, where nearly two-fold performance improvement with Random Forest over Template Attack. This high level of success shows that Random Forest is generalizing well even for devices that differ in their hardware specifications.

The similarity in performance between the two classifiers despite non-Gaussian data is further explored. The 25 variables were investigated with the KS-test for normality described in Section 5.3.2, with α=1×10^{-20} for device A1. Of the 25 variables, 10 were found to be clearly non-Gaussian and exhibiting a bimodal distribution, while 15 did not reject the null hypothesis for normality at that alpha value. Visual inspection on these 15 variables showed approximately Gaussian shape to their distribution with 5000 training traces. Table 5.5 shows the minimum number of traces required to achieve the corresponding success rate in Table 5.4. Results show Random Forest requiring fewer traces to achieve the same success rate as Template Attack with this 25 variable set for majority of the cases. So for example, even if both attacks correctly extract all 16 bytes from a particular training-test device pair, Random Forest requires fewer test traces than Template Attack. Template Attack uses the Gaussian variables and to some extent the non-Gaussian variables to successfully attack the PIC microcontroller. However, Random Forest fully utilizes its non-linear properties for all variables and achieves the same success rate with fewer test traces.

5.4.3 Profiling Attack Performance With SOST Variable Reduction

It is possible that Random Forest generated RFEI variables can benefit Random Forest more than Template Attack. Therefore, an experiment is conducted using SOST as an alternative method for variable selection. Results are presented in Table 5.6. Random Forest performance is only slightly improved when compared with RFEI selected variables in Table 5.4, but improvement is much more significant for Template Attack. For example, average success rate improves from 40.88% to 69.14% when devices from family A are used to attack devices from family B. The number of the top 25 variables identified by SOST that were non-Gaussian averaged over 40 training devices for 16 key-bytes was found to be a very low 0.0031. Thus, almost all of the top 25 variables identified by SOST



(a) (b)

Figure 5.7: Number of correctly classified key bytes over 16 bytes of AES-128 with RFEI variable reduction on 40 PIC microcontrollers using (a)Random Forest, and (b) Template Attack.

are Gaussian (by our relaxed definition) for any device and any key byte, which helps to improve Template Attack performance.

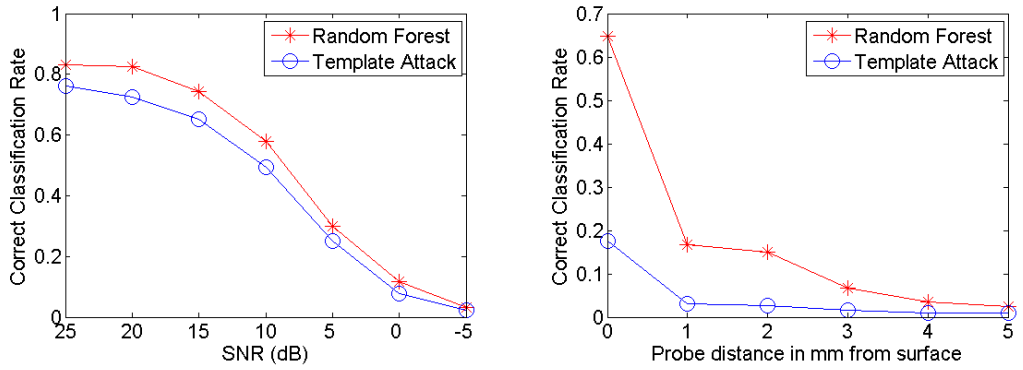
5.4.4 Performance with Gaussian Noise

Results shown in the previous sections indicate that the non-parametric classifier Random Forest can outperform the parametric Template Attack in the presence of non-Gaussian noise. In this section the effect of increasing Gaussian noise $n_{Gaussian}$ on Random Forest and Template Attack performance is discussed.

If the side-channel collected data $\mathcal{X}$ in (5.1) is truly non-Gaussian, non-parametric classifiers such as Random Forest perform well. The caveat is that the non-Gaussian noise $n_{non-Gaussain}$ from the device should be high enough that the overall measured $\mathcal{X}$ still remains non-Gaussian in the presence of Gaussian noise $n_{Gaussian}$. As the Gaussian noise power of $n_{Gaussian}$ increases, Random Forest performance logically drops to the same level or below that of parametric classifiers such as Template Attack. This is shown in the following experiment. 5000 training traces and 500 test traces were collected from a PIC device and

designated as the signal. The output of $\text{SBox}_{round1,byte1}$ is targeted here. Dimensionality reduction was performed to use the best 25 RFEI variables. Additive White Gaussian Noise (AWGN) was added to achieve a desired Signal-to-Noise Ratio (SNR). The correct classification rate for each attack method is shown in Figure 5.8a. As SNR decreases with higher power AWGN, in addition to the expected performance drop for both classifiers, the difference in correct classification rate between both attack methods also decreases.

Results seen in these simulations were verified in Figure 5.8b with real Gaussian noise generated by increasing the distance from the EM probe to the PIC surface. As distance increases, the power of the Gaussian noise $n_{Gaussian}$ from the environment increases while relative signal power d, as well as non-Gaussian noise generated from the chip, decreases. This leads to a similar trend in performance drop as seen with the simulated AWGN.



(a) (b)

Figure 5.8: Correct classification rate for collected leakage signal with (a) simulated AWGN, and (b) for real noise added by increasing probe distance from PIC surface.

Further tests are conducted with the larger population of devices to determine the effect of Gaussian and non-Gaussian variables in classification. Of the two variable importance methods, SOST requires the use of class parameters m_i (mean) and σ_i^2 (variance). RFEI on the other hand determines importance based on the reduction in entropy from using a

variable, and thus behaves in a more non-parametric fashion than SOST. Investigating the top 100 variables identified by SOST for the 40 training devices, the average number of non-Gaussian variables are shown in Table 5.3. The table shows that RFEI is more likely to rank non-Gaussian variables higher than Gaussian. Therefore, for further investigation, RFEI is chosen to identify two new sets of variables: a) the top 25 Gaussian variables *RFEI-25G* and b) the top 25 non-Gaussian variables *RFEI-25NG*. These variables were then used with both Template Attack and Random Forest to train and test the population of 40 devices.

Table 5.3: Number of Non-Gaussian variables in the top RFEI and SOST 100 variables, averaged over 16 bytes and 40 devices.

	Device Family			
Importance Method	A	B	C	D
SOST	32.54	3.51	5.31	6.28
RFEI	48.89	21.89	22.83	24.97

Success rates are shown in Table 5.7 for RFEI-25G variables and Table 5.8 for RFEI-25NG variables. They are similar for both classifiers with Gaussian variables as from the top 25 SOST variables in Table 5.6, confirming earlier analysis that SOST gives higher weighting to Gaussian variables. There is significant improvement in Template Attack performance with Gaussian variables versus using mixed RFEI variables in Table 5.4. For non-Gaussian variables, the opposite is true. Success rate significantly decreases for both classifiers with RFEI-25NG variables, with Random Forest performing slightly better than Template Attack. These results show that non-Gaussian variables were not as useful to classification as Gaussian variables.

Surprisingly Template Attack for certain scenarios performs very well with RFEI-25NG variables. For example, the average success rate with devices from family D

for training and devices from family B for test is 97.38%. Table 5.10 shows the misclassification rates for each classifier for RFEI-25NG variables. Misclassification rate is the number test traces whose key bytes were correctly identified by the classifier. In this way, results from each trace are independent which differs from the success rate using (5.5), where results are an accumulation over many traces. Tables 5.8 and 5.10 show that Random Forest typically has lower misclassification rates than Template Attack, but similar success rates. Misclassification rates for RFEI-25G variables are also provided in Table 5.9 for completeness. Template Attack is able to consistently show a high posterior probability for the correct key byte over multiple test traces, even if it is not the highest posterior probability. This effect accumulates when calculating the key guess with (5.5) to provide a high success rate, even with non-Gaussian variables. This shows that Template Attack is robust against non-normality and given sufficient test traces, can guess the correct secret key with high degree of success.

Finally, in all experiments to this point, a pooled covariance matrix is used with Template Attack which is necessitated by the need to provide an invertible matrix to (5.2) with just 5000 training traces and dimensions of sizes 3200 or 25. With 5000 traces and 256 classes, there are an average of 19.53 observations per class assuming a uniform distribution. If the number of dimensions is reduced to less than 19 then standard Template Attack can be used where the covariance matrix is estimated for each class. We reduce the number of dimensions to 5, choosing the best Gaussian variables as chosen by SOST (*SOST-5G*). This provides the most favorable conditions possible for Template Attack for our given data set. Results of classification testing are shown in Table 5.11. They show that Template Attack performance now more closely matches that of Random Forest, often beating Random Forest for certain cases. Values in bold show cases when Template Attack outperformed Random Forest.

Thus, non-parametric classifier performance approaches that of parametric classifiers in the presence of increasing Gaussian noise and Gaussian variables were more useful to classification than non-Gaussian. Both classifiers were able to achieve high success rates even with non-Gaussian data, and this particularly highlights Template Attacks robustness against non-Gaussianity. Further, given a reduced variable set, Template Attack with templates using individual class estimates for mean and covariance approach the performance of Random Forest.

5.5 Conclusion

In this work, a case where side-channel data was found to have many non-Gaussian variables was shown. For this data set and these conditions, the non-parametric classifier Random Forest outperformed the parametric classifier Template Attack. Random Forest can solve the common challenge of not knowing exactly when the information is being leaked. Using the side-channel measurement of the full encryption operation with 50,000 variables, Random Forest was able to successfully find all 16 key-bytes when the training and test device was the same, and 15/16 key-bytes when they were different. Expanded cross-device attacks with a smaller 3200 variable set allowed Random Forest to achieve success rates as high as 100% while Template Attack was not able to find any key bytes correctly. With RFEI and SOST variable reduction methods, Random Forest and Template Attack performance improved. Random Forest for both cases was better at generalizing during training to be able to attack devices that are more physically dissimilar to each other, achieving as high as two-fold performance improvement over Template Attack. SOST variables were found to improve Template Attack performance more than Random Forest performance, mostly due to the SOST preference for Gaussian variables. Template Attack performance was also found to be robust when only non-Gaussian variables were used for classification, achieving success rates as high as 98.31% despite having corresponding misclassification rates as high as 99.41%.

Table 5.4: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with RFEI variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	100.00	77.19	80.87	73.19	100.00	40.88	47.75	39.50
B	41.88	100.00	100.00	100.00	45.06	100.00	100.00	100.00
C	49.50	100.00	100.00	99.88	43.56	100.00	100.00	100.00
D	42.12	100.00	100.00	100.00	43.44	99.94	100.00	100.00

Table 5.5: Average number of traces required to achieve 90% success rate for Random Forest and Template Attack with RFEI variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	2.52	9.15	9.06	8.97	4.64	8.83	8.97	8.64
B	7.19	4.05	4.10	4.29	9.33	6.22	7.03	7.20
C	7.46	4.49	3.94	4.82	8.66	6.54	6.25	7.34
D	7.16	4.20	4.28	3.78	8.91	6.52	7.35	6.55

5.6 Subsequent Research

In [91], the 40 PIC microcontroller data set collected by Cobb et al. in [25] was analyzed. For a particular device, 32,102 out of 50,000 total variables were found to exhibit non-Gaussian distribution, as measured by the one-sample Kolmogorov-Smirnoff hypothesis test for normality with α=1e−20. The remaining 39 devices also showed similar numbers of non-Gaussian variables. A new study was performed to understand the spatial

Table 5.6: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with SOST variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	100.00	80.50	83.00	75.00	100.00	59.12	59.44	54.37
B	43.31	100.00	100.00	100.00	47.56	100.00	100.00	100.00
C	51.31	100.00	100.00	99.75	46.00	100.00	100.00	100.00
D	40.69	100.00	100.00	100.00	45.15	99.94	100.00	100.00

Table 5.7: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with RFEI-25G variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	100.00	79.25	81.81	75.68	99.94	57.25	58.12	54.56
B	41.43	100.00	100.00	100.00	45.50	100.00	100.00	100.00
C	49.75	100.00	100.00	99.75	43.94	99.94	100.00	99.94
D	40.25	100.00	100.00	100.00	40.44	99.94	100.00	100.00

dependence on the number of non-Gaussian variables in a collection. This can provide further information on possible sources of the non-Gaussian noise in the collection.

5.6.1 Power Model Based Side Channel Theory

In [72], it is theorized that a microcontroller follows the Hamming Weight or Hamming Distance power models. For a microcontroller that follows the HW power model,

Table 5.8: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with RFEI-25NG variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	100.00	33.00	42.06	39.00	82.06	19.69	30.75	25.81
B	16.25	99.50	99.19	99.81	18.87	98.50	91.94	98.31
C	26.50	98.44	98.31	99.06	32.44	95.75	90.75	95.62
D	36.50	99.56	99.38	99.50	39.12	97.38	90.94	97.88

Table 5.9: Average misclassification rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with RFEI-25G variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	55.37	95.92	94.69	95.38	99.34	99.59	99.58	99.59
B	95.96	74.78	78.28	77.97	99.56	99.41	99.40	99.41
C	94.84	80.42	72.49	80.27	99.57	99.39	99.41	99.40
D	95.87	78.96	79.61	71.39	99.53	99.41	99.40	99.40

authors state that the distribution of variables with high information leakage will follow a multimodal distribution, with the modes centered around the power consumption for the hamming weights. For the 256 possible values of an 8 bit number, the HW distribution is not uniform. This is shown in Figure 5.9, where numbers with hamming weights 3, 4, and 5 are more common others. In their experiment with an 8-bit microcontroller using

Table 5.10: Average misclassification rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with RFEI-25NG variable reduction.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	66.08	97.55	97.13	97.25	99.30	99.49	99.52	99.51
B	98.33	89.63	91.62	90.93	99.49	99.28	99.30	99.30
C	98.07	92.12	89.76	91.21	99.52	99.32	99.32	99.35
D	97.62	91.06	90.94	87.54	99.46	99.25	99.26	99.28

Table 5.11: Average success rate percentage over 16 bytes and 100 iterations for Random Forest and Template Attack with SOST-5G variable reduction. Separate class estimates of covariance matrices are used for Template Attack.

Train Device	Test Device							
	Random Forest				Template Attack			
	A	B	C	D	A	B	C	D
A	93.69	52.94	62.00	52.25	93.69	42.38	53.69	46.31
B	20.37	88.56	90.00	90.19	**21.31**	**91.12**	89.31	**91.69**
C	40.06	83.87	85.00	81.31	38.06	**86.12**	**88.31**	**83.06**
D	25.12	81.50	79.69	90.06	23.12	78.94	76.62	86.94

Correlation Power Analysis (CPA), Mangard et al. collect the power emitted during an 8-bit register look-up operation. The histogram of the variable at time sample 362 ns for 51,200 collected traces is shown in Figure 5.10a. Results show a multi-modal distribution with modes claimed to be centered around the HW leakage for the byte being attacked as theorized in Figrue 5.10b. With this theory, and the assumption of multivariate Gaussian

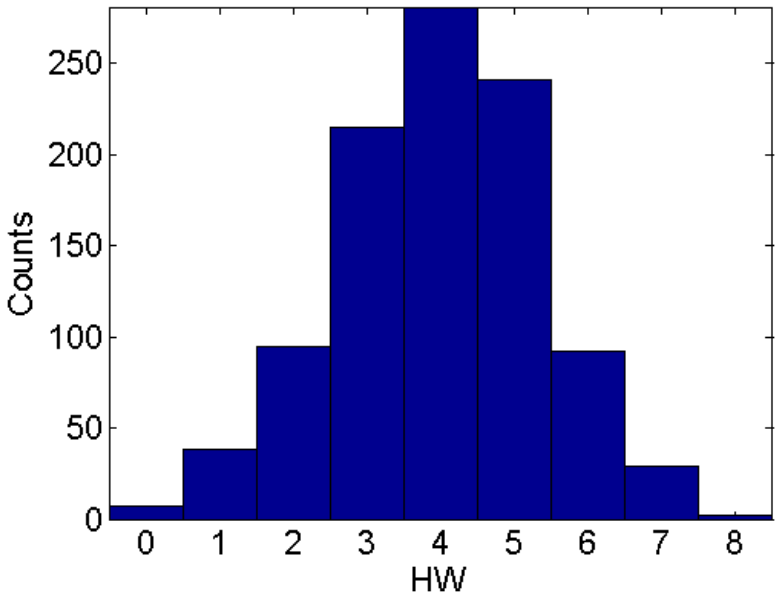
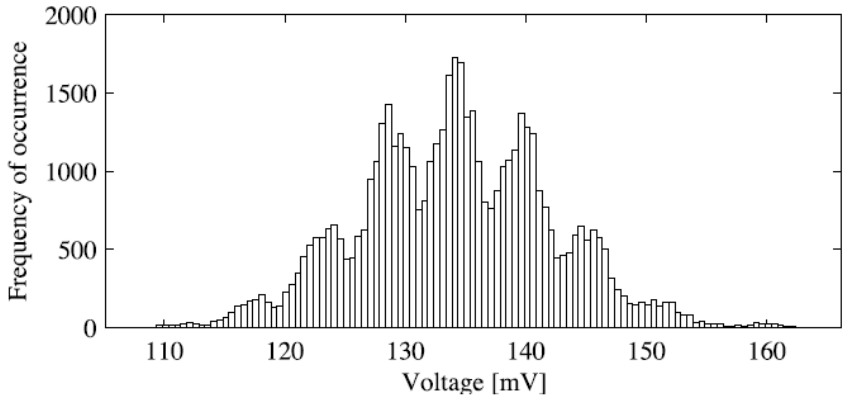


Figure 5.9: Histogram of HW for 1000 random 8-bit integers

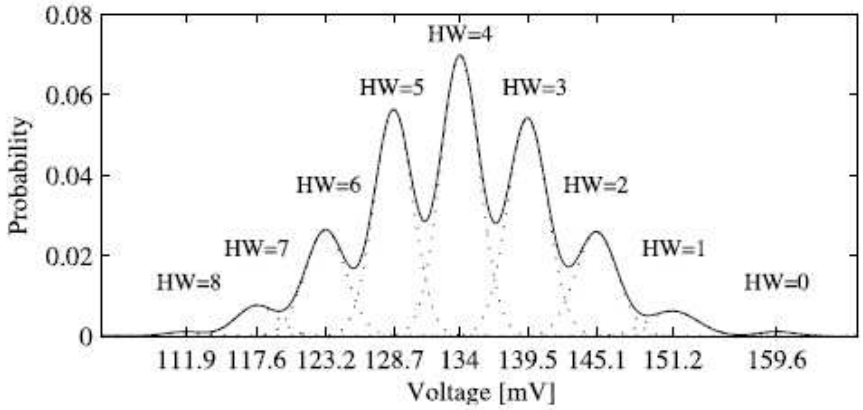
noise in the side-channel, areas on the microcontroller with high information leakage will have more non-Gaussian variables than areas with low information leakage. We test this theory by performing an *XY scan* of a PIC microcontroller and testing for non-Gaussian variables at each location.

5.6.2 Spatial Distribution Analysis with Constant Gain Control

Cobb et al. collected the 40 PIC dataset with unintentional EM emissions over the same location on each chip in [25]. In order to gain insight on the relevance of data collection location on the chip towards producing non-Gaussian variables, an *XY scan* of a PIC microcontroller is performed from the original set of 40, namely device C1 described in Section 5.3.1. An XY scan uses the Riscure XY table described in [111] to systematically place the Riscure EM probe at different locations on the device. A scanning grid of 5×25 was used, yielding 125 separate collection locations. For conciseness, the original 40 PIC dataset collected by Cobb et al. is called the *40PIC* dataset, and the new XY scan of device



(a)

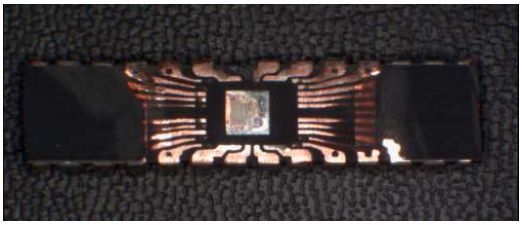


(b)

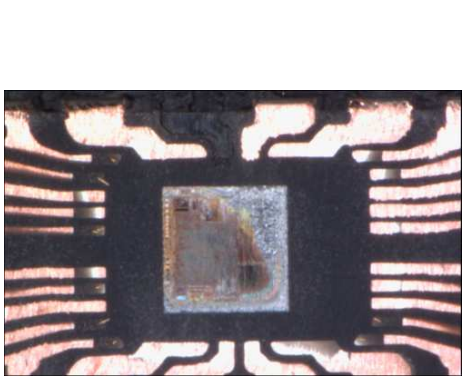
Figure 5.10: (a) Measured distribution from the power measurements of an 8-bit microcontroller at time sample 362ns for 51,200 traces [72], and (b) the theorized underlying distributions of the HW at that time sample [72].

C1 is called the *XY* dataset. A PIC microcontroller of the C device family (not from the *40PIC* device set) was delayered using a belt-sander power tool to determine the underlying structure and its effect on the generation of non-Gaussian variables. Figure 5.11 shows this device with three zoom settings. The CPU core is located in the middle of the PIC package with bonding wires seen extending to the external pins. These wires, made of relatively

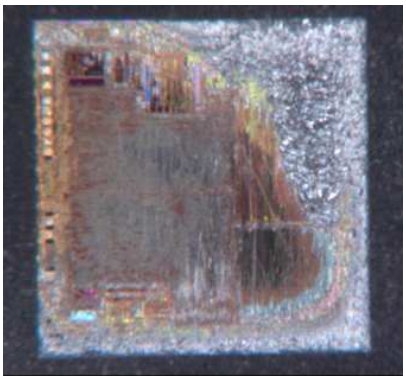
long linear metallic structures, behave like antennae and will radiate EM signals better than the comparitively short transistors located in the CPU core.



(a)



(b)



(c)

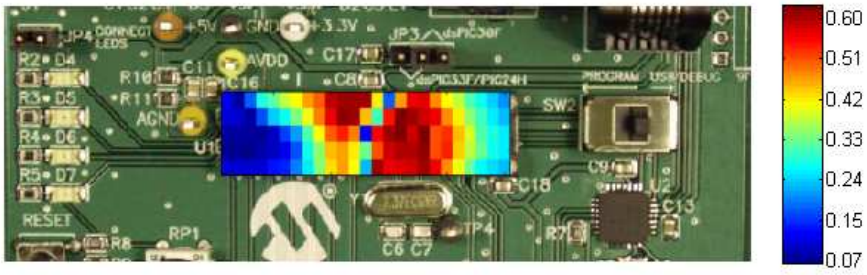
Figure 5.11: PIC microcontroller delayered and shown at three zoom settings (a), (b), and (c).

Every attempt was made to duplicate the setup by Cobb et al. [25] in the *40PIC* dataset. In addition to using chip C1 from the original collection, the same Riscure EM high sensitivity probe (model number HS130) is used. The same AES encryption C and assembly code are used. The same model Lecroy 104-Xi-A digital oscilloscope with the same sampling frequency of 2.5 Gsps. The same number of traces (5000) were collected. For post-collection data processing, the variables are low-pass filtered with a 100 MHz 8th order Chebychev digital filter and downsampled to 250 MHz, as was done in the *40PIC* collection [25]. The Riscure voltage probe contains a metal flap around the collection

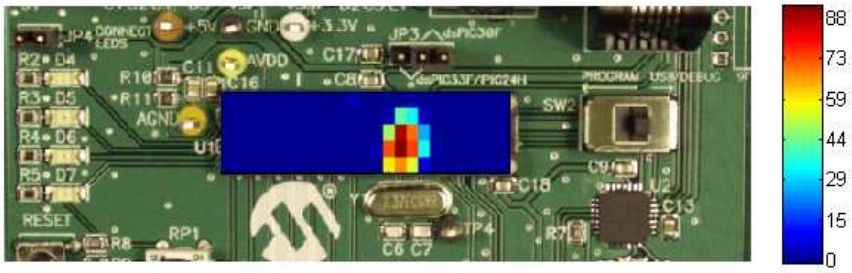
region which can either be raised or lowered during collection. This flap is raised in this collection to match the setting used during the *40PIC* collection.

One of three PIC-16 demonstration boards that house the microcontroller are used. It is uncertain which of the three were used by Cobb et al. in the original *40PIC* collection. However, clock frequency of the external oscillator was measured and verified to match that listed in the datasheet [78] to the second decimal place. The Agilent power supply was not available during the *XY* dataset collection, so a generic power supply is used. However, voltage is regulated with a surge protector. Oscilloscope vertical scale is set to 1V per division. It is not certain what vertical scaling was used in the *40PIC* collection. The entire *XY* dataset collection setup is placed on an Electrostatic Discharge (ESD) mat to reduce the effect of static electricity. This setup was not available during the original *40PIC* collection. As 5000 traces are collected at each of 125 locations, only 50,000 variables are collected, which after 10x downsampling during post-processing gives 5000 final variables per location. This is one-tenth the number of variables collected in the *40PIC* dataset. The first byte of the SubBytes output, which is the AES intermediate value under investigation, is processed by the microcontroller within these first 5000 variables. This reduction in variable count was necessary to complete data collection and analysis in a timely manner.

In Chapter 4, it is shown that the PIC follows the HW power model well. The maximum correct-key correlation from a CEMA attack can provide a measure of how much side-channel leakage is occurring at a particular location on a chip. CEMA attacks remove the requirement for separate training and test datasets, which are required for more powerful profiling attacks, and are therefore the fastest and most versatile attack for the multi-location testing performed here. At each location, a known-key CEMA was performed with 5000 traces attacking the first byte of the first round SubBytes AES operation.



(a)



(b)

Figure 5.12: (a) The maximum correlation of known key CEMA attack from an XY scan of device C1, and (b) corresponding number of variables identified by KS-test with α=1e – 20 after Matlab decimate post-processing.

Figure 5.12a shows the maximum correlation to the HW power model after a CEMA attack. Results have been overlayed on the location of the PIC on the demonstration board. Two locations are shown as regions of high information leakage on the chip. The area to the right of the CPU core where wires connect it to the external pins show regions of high information leakage. This is likely due to EM emissions from the wires connecting the

CPU core to power pin 8 (V_{SS}) and clock pins 9-12 (OSCI,OSCO,SOSCI,SOSCO) shown in Figure 5.13 [79]. The power and clock pins have been shown in literature to be good sources of side-channel leakage for cryptographic functions The high leakage region to the upper left of the CPU core is less clear. The external pins located here (pins 22-24 in Figure 5.13) are standard Input-Output (IO) ports and are not used in our implementation. Capacitor C8 is a 100 nF capacitor in-line with the power supply [78]. This suggests that it is a power regulating circuit which could account for side-channel leakage.

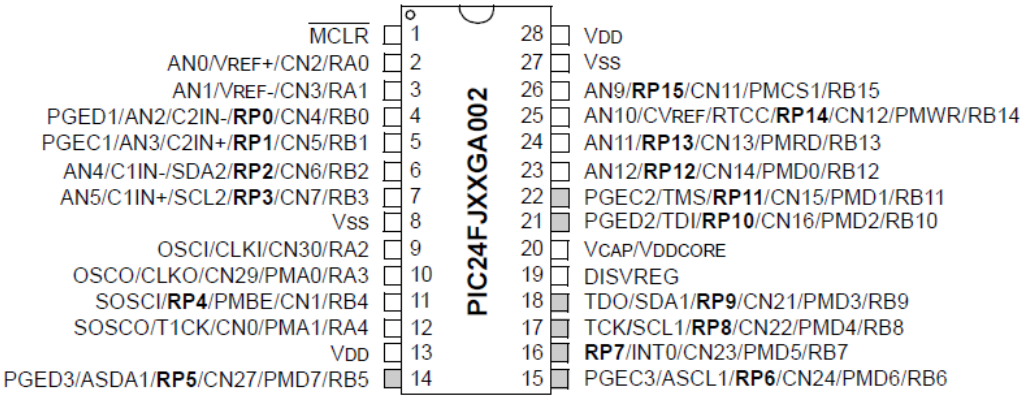


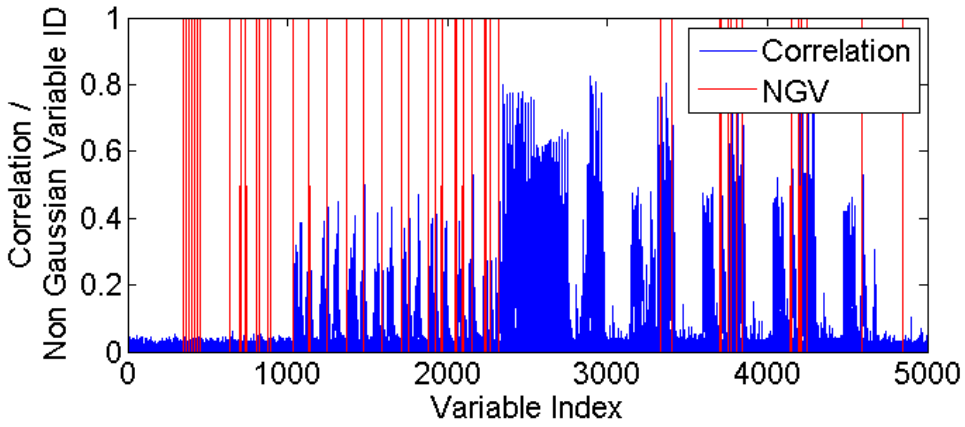
Figure 5.13: Pin diagram of PIC microcontroller [79].

Shown in Figure 5.12b shows the number of non-Gaussian variables out of 5000. The count is substantially lower than the 32,102/50,000 identified from the *40PIC* dataset. The reason for this difference could not be determined as of this time. However, the intensity map in Figure 5.12b reveals several useful insights. The region of high information leakage in Figure 5.12a to the right of the CPU corresponds well to the region of high non-Gaussian variable count. Location 116 (row 4, column 16) was the location with the largest number of non-Gaussian variables (88). This matches the existing theory that variables with large information leakage are multi-modal and non-Gaussian.

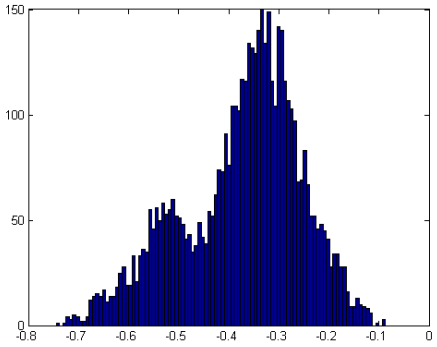
Surprisingly, the region on the PIC to the upper left of the CPU, near capacitor C8, does not show many non-Gaussian variables. This is in contrast to the existing theory [72]. The exact cause for this difference is not known. Figure 5.14a shows a plot of the correlation from a CEMA attack against all 16 bytes of the SubBytes AES operation. Each byte was attacked separately and the results are overlayed on each other. This gives an indication of all the time samples where that operation was being processed by the microcontroller. Overlayed in red are the locations where non-Gaussian variables (NGV) were identified. Results identify several locations where there is there is a good match between non-Gaussian variable and high CEMA correlation. Several time samples before variable 1000 however identify non-Gaussian variables in areas where the SubBytes operation is not identified by CEMA. However, other operations are occuring here that are not shown in this CEMA attack, namely the AddRoundKey operation. Each non-Gaussian variable was histogrammed and analyzed for their distribution. While some showed clearly irregular and often multi-modal shape as in Figure 5.14b, others showed more Gaussian shape with an extended tail, as in Figure 5.14c.

5.6.3 Spatial Distribution Analysis with Dynamic Gain Control

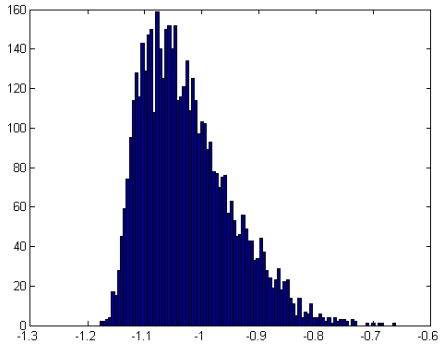
The one sample KS-test calculates the Cumulative Distribution Function (CDF) of a collected variable to the CDF of the normal distribution. The collected side-channel emissions are sampled with 8-bit resolution on the oscilloscope. For regions with very low emitted signal at constant gain control, only a few of these bits are used, leading to a very poor resolution collected signal for that variable. This lowered sampling resolution can lead to the full range of a low powered signal not being captured, which can lead inaccuracies in the KS-test. Therefore a method of dynamic gain control is used, which was developed by Montminy in [83] and modified here. This method adjusts the vertical scale on the scope based on the current measured signal to ensure that as much of the measurement range as possible is being used to sample the incoming signal without signal clipping. Signal



(a)



(b)

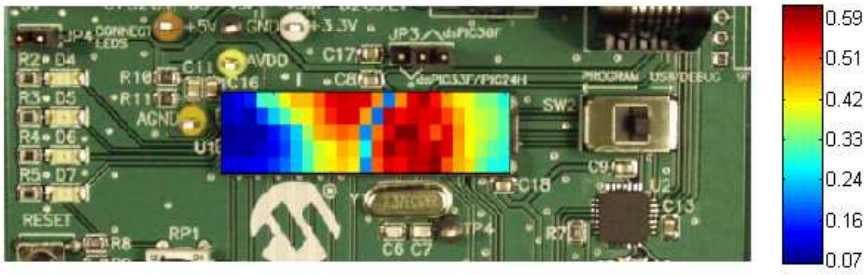


(c)

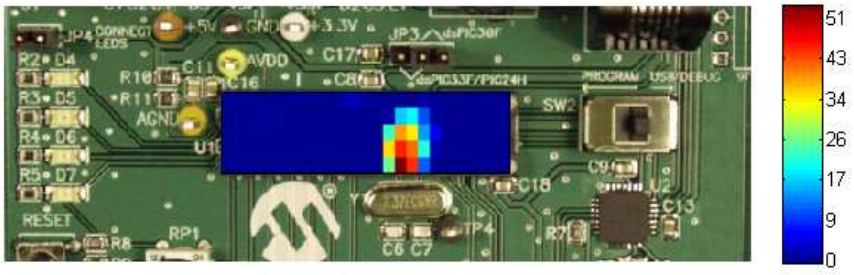
Figure 5.14: (a) Correlation from a CEMA attack to the power model for SubBytes calculation at round 1 for all 16 AES bytes. (c) Histogram of a non-Gaussian variable showing clear irregular shape, and (b) histogram of non-Gaussian variable showing a shape similar to a Gaussian distribution but with a longer tail.

clipping is when a signal is too large to be measured at a given vertical scale, and its value is capped at the maximum or minimum values of the sampling resolution.

Maximum correlation after a CEMA attack at each location on the XY scan is shown in Figure 5.12a. Results are improved over Figure 5.12a by using higher range resolution



(a)



(b)

Figure 5.15: (a) The maximum correlation of known key CEMA attack from an XY scan of device C1 with dynamic gain control, and (b) corresponding number of variables identified by KS-test with α=1e − 20 after Matlab decimate post-processing.

when the input signal is amplified to account for low input power. The plot of the number of Non-Gaussian variables is shown in Figure 5.15b. Here, results remain very similar to those in Figure 5.12b.

For cases with and without dynamic gain control, the number of non-Gaussian variables identified are much reduced compared to those from the *40Pic* dataset, and the

multi-mode nature is not as easily identified as those in Figure 5.2 from the *40Pic* dataset. The exact cause for this is unknown at present and requires further study. Frequency analysis around the PIC and nearby peripheral devices might lead to further insight. In addition, the effect of this more Gaussian collection on Template Attack and Random Forest profiling attacks could be determined in future experiments.

VI. Results: Random Forest RF-DNA Application

This chapter presents research results on two ZigBee datasets: the *ZigBee Cross-Environment* dataset, and the *ZigBee Cross-Receiver* dataset. Research on the Cross-Environment dataset included in 'Improving ZigBee Device Network Authentication Using Ensemble Decision Tree Classifiers with RF-DNA Fingerprinting' is accepted and under revision in the *IEEE Transactions on Reliability Special Section on Trustworthy Computing*. Thereafter, additional results are presented on the Cross-Receiver data set submitted to *Military Communications Conference MILCOM 2014*. In both cases, certain text was omitted to reduce redundancy between other chapters in this document.

6.1 Introduction

ZigBee devices based on the IEEE 802.15.4 standard [10] have gained popularity in a variety of applications as devices of choice for low-cost, low-power and low-complexity communication applications [60]. Per the IEEE standard, ZigBee devices operate as either Full Function Devices (FFD) capable of functioning as a Network coordinator, or as Reduced Function Devices (RFD) capable of communicating only with an FFD. ZigBee networks can operate as decentralized mesh networks [68] such as Star, Peer-to-Peer or Cluster Tree Topology, allowing new devices to be discovered and incorporated easily into an existing network. ZigBee devices represent a 2 orders of magnitude lower-power alternative to Bluetooth communications for short message burst communications [9, 60]. The low-cost, low-power attributes of ZigBee have greatly increased their popularity and they have been widely adopted for monitoring and control in industrial and building [38], healthcare [58], and security system [126] applications.

The increased vulnerability and growing popularity of ZigBee in critical applications motivates research aimed at devising robust security measures. ZigBee operation is secured

through the Advance Encryption Standard (AES) for Media Access Control (MAC) and Network (NWK) payload encryption. Message integrity can be enabled using a Message Integrity Code (MIC) that provides protection against simple replay attacks. However, due to the resource constraints and distributed network topology inherent to ZigBee, several vulnerabilities have been recently discovered [33, 99, 127]. In addition, a Python-based tool called KillerBee was released in 2009 that increases the exposure of ZigBee and other IEEE 802.15.4-based Wireless Personal Area Networks (WPAN) to attack [130]. Specifically, KillerBee simplifies sniffing and injecting traffic, packet decoding and manipulation by rogue devices.

Several methods have been proposed to use physical characteristics of devices enhance authentication. Mitchell and Chen [81] detect network intrusion by modeling behavior of nodes in a Cyber Physical System and determine Mean Time To Failure (MTTF) rates of the network for various attack scenarios. Physically Uncloneable Functions (PUFs) provide hardware authentication by inserting additional circuits into devices that take advantage of the natural randomness of the manufacturing process and are nearly impossible to duplicate [122]. Zhang, et al. [132] model network attack behavior using Support Vector Machines with additional network modules and use these models to classify possible network intrusion.

Similarly, RF-DNA fingerprinting has been used to enhance Physical (PHY) layer security of wireless networks through identification and verification of wireless devices. In contrast to other methods, RF-DNA fingerprinting is a passive technique that does not require additional network hardware for security and authentication. For device identification (ID), RF-DNA can be used to reliably distinguish between known network devices, even if these devices are identical device model from the same manufacturer. The RF-DNA verification process also enables discrimination between authorized devices and unauthorized rogue devices attempting to gain network access to conduct spoofing

attacks. In related work, RF-DNA fingerprinting exploited preamble features in 802.16e WiMAX [107] and 802.11a WiFi [48] signals using a parametric Multiple Discriminant Analysis/Maximum Likelihood (MDA/ML) classifier and non-parametric Generalized Relevance Learning Vector Quantization-Improved (GRLVQI) and Learning From Signals (LFS) classifiers. The MDA/ML classifier was also adopted and used in preliminary demonstrations to 1) differentiate authorized ZigBee devices at varying *SNR* [102] and 2) detect the presence of unauthorized rogue devices [35] using verification. ZigBee devices are often employed in locations that make their RF emissions susceptible to interference from multipath reflection and other operating devices; such conditions can induce non-Gaussian conditions that are reflected in RF-DNA features used for device discrimination. This effectively degrades MDA/ML performance which is inherently based on underlying Gaussian distribution assumptions for the input features [25]. In addition, lack of built-in variable identification methods in MDA/ML require the use of external methods for Dimensionality Reduction Assessment (DRA).

Random Forest (RndF) and Multi-Class AdaBoost (MCA) are two robust ensemble learning classifiers not based on assumed or known input data distributions [18, 125, 133]. They have good performance in large dimensional, multi-class problems [31, 91, 109]. In fact, non-parametric RndF and MCA ensemble decision tree classifiers enhance device identification and verification performance using RF-DNA input features. Results are generated here using experimentally collected ZigBee emissions and compared to parametric MDA/ML and GRLVQI methods that have been previously shown to be effective for RF-DNA fingerprinting applications [47]. This work builds on previous research [94] where identification and verification of ZigBee emissions with RndF and MCA were first investigated.

6.2 Background

6.2.1 Signal Collection Methodology

Signal collection, RF-DNA fingerprint generation, model generation and device testing are conducted collectively through an Air Monitor system comprising of a receiver and computational platform. The signal collection methodology outlined in [36] is maintained herein. All signal collections are performed with an Agilent E3238S-based receiver [1] that down-converts signals to near-baseband, digitizes them using a 12-bit Analog-to-Digital Converter (ADC) and stores the samples as complex In-phase and Quadrature (I-Q) components. Subsequent post-collection processing is performed at a sample frequency of f_s=11.875 Msps using an 8th-order Butterworth filter having a bandwidth of W_{BB}=1.0 MHz. The authorized device data set used for model development consists of N_p=3000 preamble responses collected from each of N_D=4 like-model CC2420 IEEE 802.15.4 compliant devices operating at 2.4 GHz. The N_p=3000 responses per device includes 1000 responses collected under each three different experimental ZigBee Transceiver-to-collection Receiver (Tx-Rx) scenarios, including: 1) the Tx and Rx in a Ramsey STE3000 anechoic enclosure (*Cage*), 2) the Tx and Rx having a clear line-of-sight (*LOS*) down an indoor hallway, and 3) the Tx and Rx on opposite sides of indoor wall (*Wall*). The rogue device data set consisted of an additional N_p=1000 preamble response collected from each of N_D=9 devices operating under varying experimental Tx-Rx scenarios.

An amplitude-based detection threshold of T_D =6.0 dB was used to detect ZigBee burst leading edges. This provided reliable identification and extraction of the desired preamble region of interest. The experimentally observed preamble duration of $T_p \approx 129\mu$sec was consistent with IEEE 802.15.4 specification [56]. The average *SNR* of detected bursts for each of the experimental scenarios was *SNR* ≈ 50.0 dB (*Cage*), *SNR* ≈ 40.0 dB (*LOS*), *SNR* ≈ 30.0 dB (*Wall*).

6.2.2 *Statistical RF-DNA Fingerprint Generation*

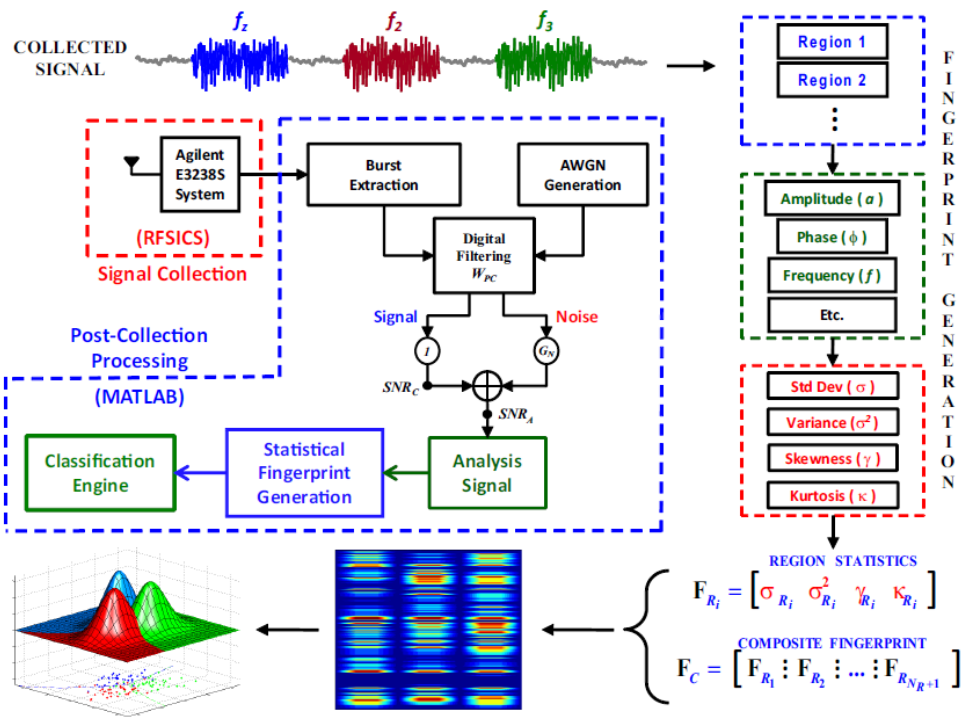


Figure 6.1: RF-DNA Fingerprint Generation Process [129].

Figure 6.1 shows the signal collection and RF-DNA fingerprint generation methodology. A statistical RF-DNA fingerprint ($\boldsymbol{F}$) is derived from the signals instantaneous amplitude (a), phase (ϕ) and/or frequency (f) responses as described in [129]. The corresponding response sequences $a[n]$, $\phi[n]$ and $f[n]$ are generated from the complex signal samples within the region of interest. These sequences are centered (zero mean) and normalized (divide by maximum value) prior to calculating statistical RF-DNA features of variance (σ^2), skewness (λ) and/or kurtosis (k) within selected signal region(s) of interest. The regional fingerprint markers for the signal are generated by 1) dividing each response sequence into N_R contiguous equal length sub-sequences, 2) calculating N_S statistical metrics for each sub-sequence, plus the entire region as a whole for $N_R + 1$ total regions, and 3) arranging

the metrics in a vector of the form

$$F_{Ri} = \left[\sigma_{Ri}^2 \lambda_{Ri} k_{Ri}\right]_{1\times 3}, \tag{6.1}$$

where i=1, 2, ..., N_R + 1. The N_R + 1 regional vectors from (1) are concatenated to form the composite fingerprint vector given by the concatenated vectors

$$\boldsymbol{F} = \left[F_{R1}|F_{R2}|F_{R3}|\ldots|F_{R(N_R+1)}\right]_{1\times N_S(N_R+1)}. \tag{6.2}$$

For consistency with results in [102] and [36], all performance assessments conducted here are based on a total of N_R + 1=81 subregions within each of the $a[n]$, $\phi[n]$ and $f[n]$ responses and N_S=3 statistics for each subregion. According to (6.2), the resultant RF-DNA fingerprints used for assessment contain N_F=(3x3x81)=729 total features.

6.2.3 Classifier Description

Four different classifiers are considered for comparative assessment using RF-DNA fingerprints extracted from experimentally collected ZigBee emissions. It is important to note that when direct comparisons are made the results are based on *identical* RF-DNA feature sets into being submitted to classifiers. A brief description of each classifier is provided in the following sub-sections.

6.2.3.1 Multiple Discriminant Analysis/Maximum Likelihood (MDA/ML)

MDA/ML is a parametric classification process consisting of a transformation (MDA) followed by a parametric classification decision (ML). MDA is a multi-class form of Fishers Linear Discriminant Analysis that seeks the linear subspace that maximizes inter-class mean distance while reducing intra-class variation and improving overall class separability. This transformation effectively reduces the N_F dimensional input data to an (N_c1) dimensional space where N_c is the number of classes. Given that MDA uses linear eigenvector decomposition to generate transformed features, and assuming Gaussian input features, the transformed features remain Gaussian as well. However, if non-Gaussian input is provided the transformed features are not guaranteed to be Gaussian.

The MDA transformed features are classified via a ML decision assuming a multivariate Gaussian distribution of projected data. A mean vector μ_i is estimated for each class, along with a pooled covariance matrix Σ that is used for all classes. Likelihood estimation is calculated for a multivariate Gaussian distribution given by

$$P(\mathbf{x}|i) = \frac{1}{(2\pi)^{N_F/2}|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_i)\Sigma^{-1}{}_i(\mathbf{x}-\mu_i)^T\right\}, \tag{6.3}$$

where T denotes transpose. A sample from the input *testing* set is projected via MDA and class likelihood values are assigned using Bayesian decision theory, where the conditional posterior probability is given by

$$P(i|\mathbf{x}) = \frac{P(\mathbf{x}|i)P(i)}{P(\mathbf{x})}. \tag{6.4}$$

Assuming equal prior probabilities $p(i)$ for each class and equal $p(x)$ for all classes, these terms in (2.11) can be ignored, giving

$$p(i|\mathbf{x}) \propto p(\mathbf{x}|i)\ . \tag{6.5}$$

Thus, the posterior probability under these conditions is proportional to the conditional probability $p(\mathbf{x}|i)$ which can be estimated during the training phase. The final class estimate of a test observation is assigned to the class i with highest probability, or

$$p(i|\mathbf{x}) > p(j|\mathbf{x}),\ \forall i \neq j\ . \tag{6.6}$$

The class yielding highest probability is assigned as the class estimate for the unknown testing input. As typically large dimensional distributions can have very small probabilities [59], log-likelihood is used here. Given a *known* distribution, ML is known to provide optimal classification performance [125].

6.2.3.2 Generalize Relevance Learning Vector Quantized-Improved (GRLVQI)

GRLVQI is a neural learning algorithm that iteratively adjusts prototype vectors to define class boundaries [63]. A system of K prototype vectors is arbitrarily chosen for

the data set. Weight vectors are assigned to a first layer of neurons with each neuron representing one class. The weight vector of a class neuron is iteratively updated (*learned*) to reduce the training data distance of that class in a manner similar to supervised K-Nearest Neighbor processing. LVQ was generalized to GRLVQ in [114] using gradient descent to minimize drifting of the prototype vectors from their optimal locations. Hammer modified the algorithm to incorporate variable relevance ranking [44] which was key to successful implementation of feature DRA work in [106]. GRLVQI was successfully used for RF-DNA fingerprinting work in [47] involving identification and verification of WiFi and WiMAX devices. In addition to vector class labels, GRLVQI provides a distance metric that provides a measure of "confidence" in the class estimate. Device ID verification performance was improved in [106] by modifying the confidence measure to account for prototype vector angle as well as distance.

6.2.3.3 Ensemble Learning Classifiers

Ensemble learning classifiers combine multiple weaker classifiers to reach a unified classifier that is stronger than individual components. Early ensemble learning algorithms trained the same classifier using different training data subsets and bootstrap aggregation or bagging [16]. When these training subsets generated significantly different classifiers, the accuracy was found to improve [16]. The Random Subspace method also manipulates the training space by randomly selecting a subset of variables to train different classifiers [53]. As implemented here, RndF is a decision tree ensemble learning algorithm that combines bagging with the random subspace method to grow de-correlated decision trees to form the ensemble [17]. Forests of multivariate trees such as Random Forest Random Combination [17], Oblique Random Forest [73], and Rotation Forest [64] have been suggested for data sets with correlated input variables.

Boosting is a method for iteratively training a weak classifier on successively challenging training observations. Initially, a weak classifier is trained using an unweighted

subset of training observations chosen randomly with a uniform distribution. Observations not used for training are classified and incorrect guesses are given higher weighting. The process is repeated for successive iterations by choosing training observations from a weighted distribution. Thus, training observations that are more challenging to classify are given higher preference in later classifier training iterations.

Advanced methods have been developed for generating ensembles that selectively choose models for a given training set. For example, forward stepwise selection is used in [23] for model selection and can be optimized for performance metrics such as accuracy, cross entropy, mean precision or area under the ROC curve. The authors in [23] demonstrate that component models such as Support Vector Machine (SVM), Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Decision Trees and others can be used together in an ensemble. The bagged ensemble selection method in [23] is further developed in [123] using bagged groups of models with ensemble selection occurring within each bag.

In this research, RndF and MCA ensemble classifiers are chosen for demonstration. Univariate decision trees are chosen as the weak classifiers over multivariate trees as a study of RF-DNA fingerprint variables shows only a few have variables have high correlation; for 530,719 unique pair-wise combinations of 729 variables ($\mathbf{x}_i \neq \mathbf{x}_j$) only 15% showed correlation higher than 0.5. Further, as part of this research training models are constructed at each of 16 different SNRs. Therefore, RndF and MCA methods were chosen here over more complex ensemble selection methods to maintain manageable classifier training times while ensuring reliable proof-of-concept demonstration.

Random Forest (RndF) RndF is a decision tree ensemble learning algorithm where the class estimate is determined by a majority vote [17]. Decision trees are grown to full depth, i.e., until all terminal leaves are pure (contain only one class), to minimize prediction bias. The correlation between trees is minimized by using a different randomly selected

training data sample (with replacement) for each decision tree; the non-selected training observations are called the Out-Of-Bag (OOB) observations. At each decision tree node, a random subset of m variables is used to further minimize inter-tree correlation. The best variable from the subset of m is used to split the data into two child nodes. Various metrics have been used to determine the best node including Shannons entropy [15], Gini impurity [74] and Area Under the Curve (AUC) [39]. Shannons entropy was selected here given that it yields similar performance to Gini Entropy [100] and is more computationally efficient than the AUC method for multi-class problems. Although RndF is a non-parametric classifier, posterior probabilities can be determined as $p(\mathbf{x}|i)=N_i/N$ where i is the class label, N_i is the number of trees that voted for class i, and N is the total number of trees in the ensemble.

The RndF classifier provides a built-in Variable Importance (VI) metric called permutation importance. Once a forest is grown, the Out-Of-Bag Error (OOBE) is calculated for each tree as the misclassification percentage of the OOB set. This is averaged over all trees in the forest to give an overall baseline OOBE. Each variable in the training data is randomly permuted and the resultant forest OOBE recalculated. The difference in OOBE between the permuted data and the baseline OOBE is stored as the permutation importance. More important variables yield a larger OOBE difference relative to the baseline when they are permuted giving a relative importance ranking of variables.

Multi-Class AdaBoost (MCA) AdaBoost is a boosting algorithm designed to iteratively train weak classifiers based on misclassifications that occur during the previous iteration [20]. Each of the N_{tr} training observations starts with a weight of $1/N_{tr}$ and a weak classifier such as a decision tree T_i is grown on an in-bag sample of the training set. The entire training set is then classified by T_i and the weights of misclassified observations are increased while decreasing the weights of correctly classified observations. The random in-bag sample of observations is chosen by this weighted distribution. Thus, MCA focuses

on harder to classify observations by giving them a higher weight and making them more likely to be used during the next iteration classifier T_{i+1}. Weights of all observations for classifier T_i are used to generate a single weight α_i and a final class vote is determined using a weighted majority vote given by

$$C(x) = \underset{k\in\{0,1\}}{\operatorname{argmax}} \sum_{i=1}^{N} \alpha_i \cdot I\left(T_i(x) = k\right), \tag{6.7}$$

where N is the total number of decision trees in the ensemble, and I is the indicator function which takes on a value of 1 if the class vote from T_i for observation x is k and 0 otherwise. AdaBoost was originally developed for 2-class problems and subsequently extended to multi-class applications using a multi-class exponential loss function [133].

The RndF and MCA classifiers have been empirically shown to provide high classification performance using high-dimensional input data sets. Good RndF and MCA performance was demonstrated in [109] with performance measured by accuracy, Root Mean Square Error (RMSE), AUC and average across these metrics for the datasets IMDB (685K variables), Spam (405K variables), DSE (195K variables), and Cite (105K variables). RndF was used in [31] with microarray datasets for Adenocarcinoma (9,868 variables), prostrate (6,033 variables), and brain (5,597 variables). RndF has also been applied in side-channel analysis of spectral data (50K variables), with results in [91] demonstrating successful classification of all 16 bytes of the encryption key. In addition to their success in high-dimensional problems, the non-parametric properties of RndF and MCA make them excellent alternatives to MDA/ML in RF-DNA fingerprinting for ZigBee devices. For results presented here, these classifiers were coded in Mathworks Matlab software, with the decision tree node split algorithm applied in C++ to reduce computation time. Results of this implementation are compared to original RndF results in [17] using the *glass, breast cancer, Pima diabetes, sonar, vowel, ionosphere, zip code* and *letters* datasets. In each case, classification performance using the RndF and MCA implementations here was equal to or better than prior work. In addition, the RndF and MCA implementation

performance was comparable to that of the *Weka* and *R* versions of RndF and MCA for the spectral dataset described in [91].

6.3 Results

6.3.1 Data Set Definitions

For this research, two primary datasets are used and designated as the *Inst* and *Stat* datasets. The *Inst* dataset consists of observations with 5760 variables, made up of 1920 variables each from the instantaneous amplitude ($a[n]$), phase ($\phi[n]$) and frequency ($f[n]$) responses. The *Stat* dataset consists of observations with 729 statistical variables derived from the *Inst* dataset, with 243 variables each from RF-DNA amplitude, phase and frequency features $\boldsymbol{F}$ per (6.1), (6.2). Both *Inst* and *Stat* datasets contain 3000 observations from each of N_D=4 authorized devices for a total of 12000 observations per dataset, as described in Section 6.2.1. Equal size training and test sets are constructed with 6000 observations each as summarized in Table 6.1. For both *Inst* and *Stat* datasets, White Gaussian Noise (WGN) is added to achieve a desired $SNR \in [0, 30]$ dB in 2dB steps.

Table 6.1: *Inst* and *Stat* dataset summaries.

	Number of Test Observations	Number of Training Observations	Total Number of variables
Inst	6000	6000	5760
Stat	6000	6000	729

6.3.2 Random Forest Performance Using Instantaneous Responses

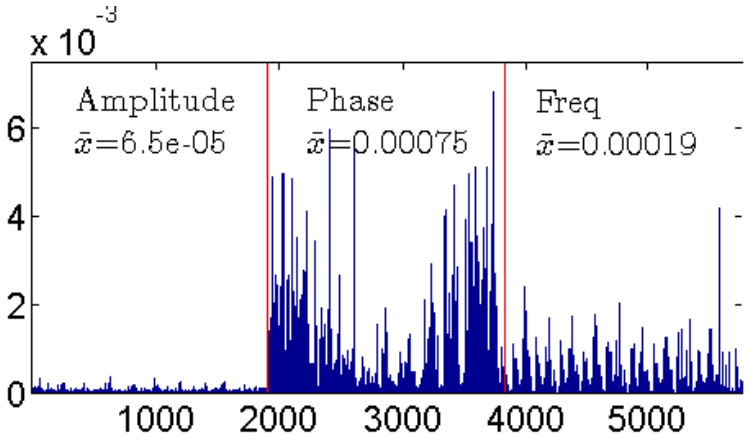
As noted in Section 6.2.3.3, RndF has been shown to perform well in large dimensional problems. Thus RndF was introduced here to directly classify authorized ZigBee devices using instantaneous amplitude ($a[n]$), phase ($\phi[n]$) and/or frequency ($f[n]$) samples of collected signals. The *Inst* dataset samples were used as input variables to

RndF for all $SNR \in [0, 30]$ dB considered. Figure 6.2a shows the permutation VI values as generated by a 1000 tree RndF from the *Inst* training dataset at SNR=10 dB. Larger RndF ensembles were experimentally used for VI tests and impact on resultant classification performance was negligible. It is evident that phase variables are the most important for ZigBee device classification followed by frequency and lastly amplitude variables. This coincides well with the results from [35] shown in Figure 6.2b, where KS-Test based VI of phase statistical RF-DNA features from the *Stat* dataset at SNR=10 dB were found to be dominant. The KS-test VI is the p-value from pair-wise KS-tests of observations for each of M variables given by Algorithm 6.

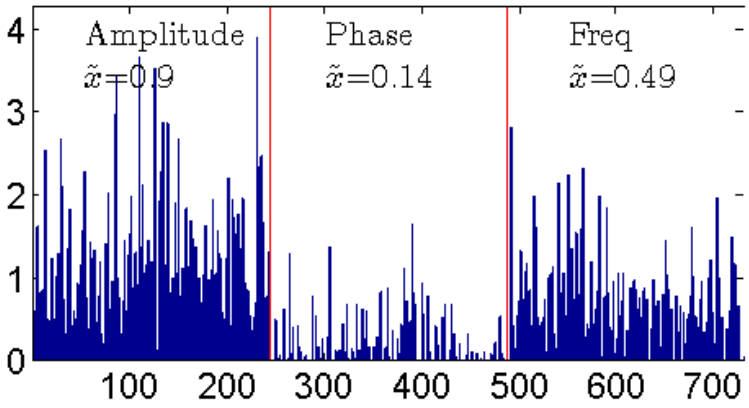
Algorithm 6 KS-Test for VI pseudo code

```
for Each variable v = 1 → M do
  for i = 1 → c classes do
    for j = (i + 1) → c classes do
      x_i = Observations from class i, variable v
      x_j = Observations from class j, variable v
      p(variable) = p(variable) + (p-value from KS-test(x_i, x_j))
    end for
  end for
end for
```

Figure 6.3a shows the $\%C$ vs. SNR for N_D=4 authorized ZigBee devices using the *full-dimensional Inst* dataset with RndF for each response type. In agreement with VI results of Figure 6.2a, the phase-only variable set achieves the $\%C$=90% benchmark at the lowest SNR=12.0 dB; the combined VI and $\%C$ results strongly indicate that $\phi[n]$ variables contain the most discriminating information. Figure 6.3b shows classification performance with DRA using the best 10%, 25%, 50%, and 100% of the 1920 instantaneous phase; the *full-dimensional* 5760 variable results are provided for comparison. In each case, the subspace size chosen at each node is m= greatest integer of $\sqrt{\text{Total Number of variables}}$. The most important features for DRA were selected based on the 1000 tree RndF model developed at SNR=10 dB. Most notably, Figure 6.3b results show 1) good agreement between full-



(a)

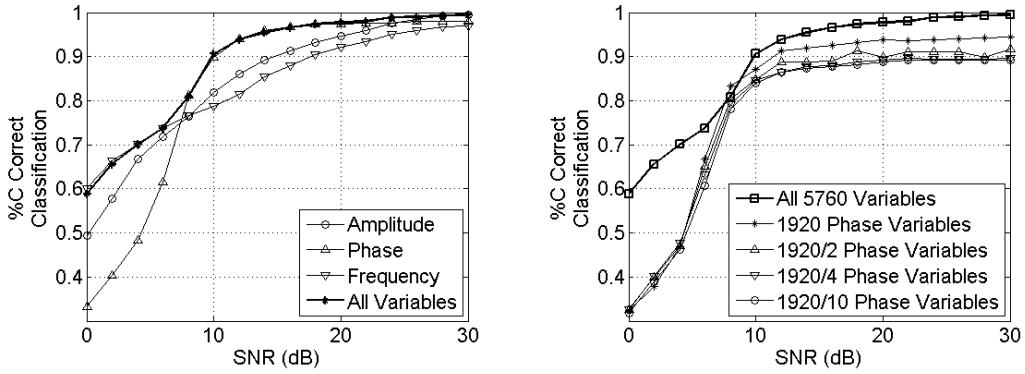


(b)

Figure 6.2: (a) RndF VI using *Inst* features at *SNR*=10 dB with higher VI indicating greater importance and (b) corresponding KS-Test *p*-values for *Stat* features at *SNR*=10 dB with lower *p*-value indicating greater importance.

dimensional and phase-only performance, to include statistically identical performance for $SNR \geq 6.0$ dB using one-third the variables, and 2) a reduction of approximately 5% in *%C* for other phase-only DRA subsets. As a dimensional reduction to one-third of the full-dimensional 5760 variables yields minimal *%C* performance degradation over the range of

SNR considered, all subsequent RF-DNA fingerprinting results and analysis are based on the phase-only 1920 variables.



(a) (b)

Figure 6.3: RndF correct classification performance for (a) the 1920 *Inst* variables from individual amplitude $a[n]$, phase $\phi[n]$ and $f[n]$ responses as well as the combined 5760 *full dimensional* (Full Dim) set , and (b) top-ranked 10%, 25%, and 50% DRA $\phi[n]$ subsets as well as the total 1920 $\phi[n]$ variables. Results from the 5760 variable Full Dim set is also shown for reference.

RF-DNA fingerprint features have been shown to successfully accumulate discriminating information from raw collected signals into smaller dimensions for classification [35, 47, 106] by calculating only 3 statistics (variance, skewness, kurtosis) over selected regions of interest spanning several time samples. This RF-DNA feature generation method is used with the Stat dataset in the remainder of this paper.

6.3.3 Variable Importance Comparison

The VI rankings generated by RndF processing are compared with VI rankings provided by pre-classification KS-Test metrics in [35] and GRLVQI post-classification feature relevance ranking methods. In each case, the training *Stat* dataset is used for VI generation. The RndF, KS, and GRLVQI methods are independently used to identify

the top-ranked 25 of N_F=243 (81 subregions × 3 statistics per subregion) phase-based statistical RF-DNA fingerprint features (variables). Collectively considering the top 25 variables identified by all three methods, there were a total of 47 unique variables identified, 18 of which were found by two or more methods. Figure 6.4 shows a Venn diagram of the top ranked 25 variables identified by each method. Variables found by two or more methods are shown by their respective intersecting regions. The top-ranked 25 variables identified by each method are used for RF-DNA fingerprinting results in Section 6.3.4 using N_D=4 authorized ZigBee devices.

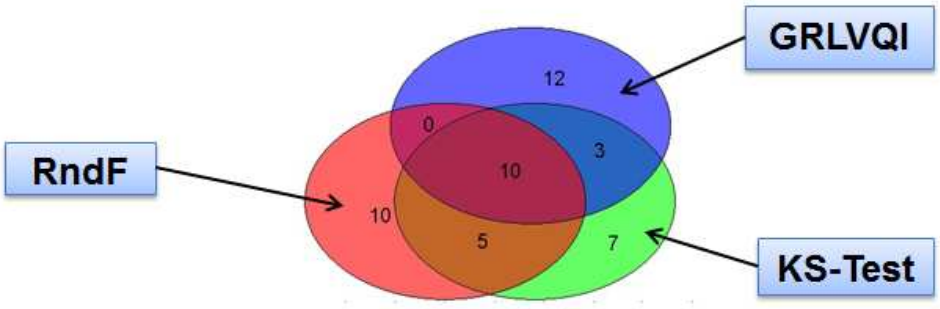
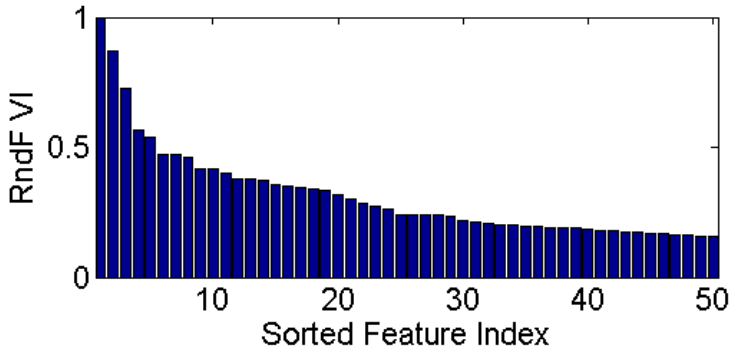
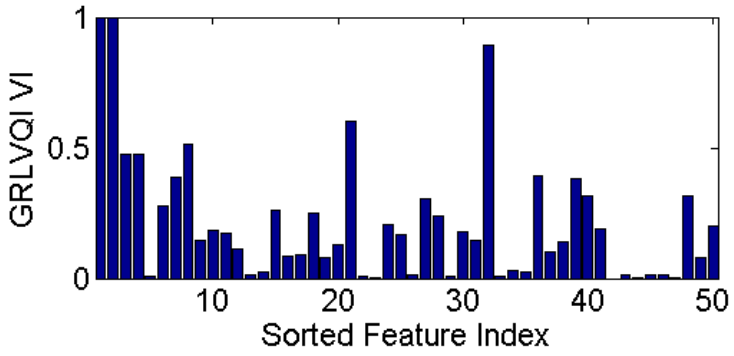


Figure 6.4: Relationship of top-ranked 25 variables identified by RndF, KS Test, and GRLVQI methods. Number of variables found by two or more methods are shown in intersecting regions.

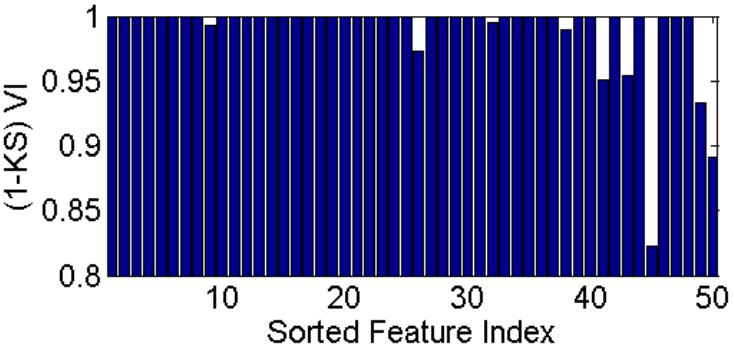
Figure 6.5a shows the normalized (divide by maximum value) top-ranked 50 variables identified by RndF VI. Figure 6.5b and Figure 6.5c show VI values for the same variables assigned by GRVLQI VI and (1-KS-test) VI methods; use of (1-KS-test)-VI here is a matter of convenience for making a "larger VI is better" comparison with the other methods. Results clearly show differences in the relative importance assigned to variables by the three methods.



(a)



(b)



(c)

Figure 6.5: (a) Normalized VI values for 50 top-ranked, sorted RndF variables and (b) corresponding GRLVQI VI (Middle) and (c) (1-KS Test) VI (Bottom) values.

6.3.4 RF-DNA Authentication Results

Figure 6.6 shows OOBE at *SNR*=0.0 dB using RndF using best 25 variables identified by RndF. Different subspace selection sizes *m* are used for decision tree node training. This low *SNR* was chosen as it has a high misclassification rate and performance differences are more easily visible. Based on these results, a subspace size of *m*=5 $\left(\sqrt{\text{Total Number of variables}}\right)$ was deemed sufficient for demonstration and used for all subsequent testing.

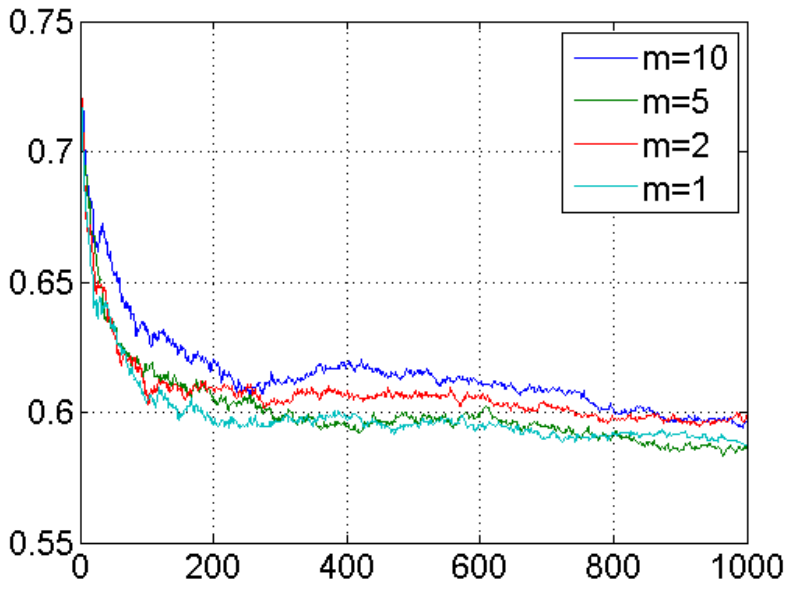


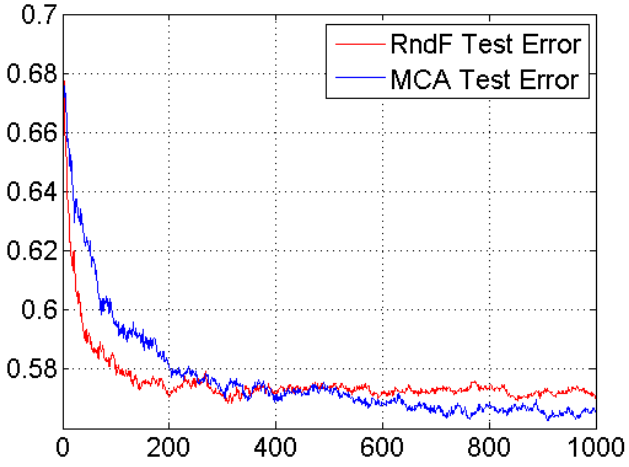
Figure 6.6: Out-Of-Bag Error (OOBE) for RndF with top-ranked 25 RndF variables and different subspace sizes *m* using *Stat* dataset at *SNR*=0.0 dB.

Figure 6.7a shows RndF and MCA test error for RF-DNA fingerprinting at *SNR*=0.0 dB using only the best 25 variables identified by RndF and N_P = 6000 test observations. Both curves reflect a general downward trend until approximately 900 trees where the misclassification rate begins to level out. Based on these results, a total of N_T=1000 trees were used for subsequent RndF and Adaboost ensemble method processing. Figure 6.7b

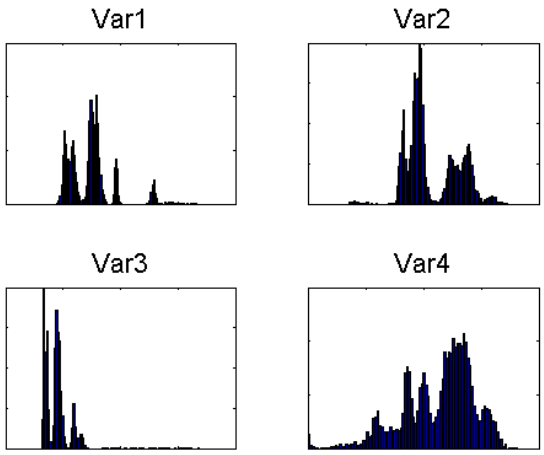
shows distributions for four arbitrarily selected fingerprint variables at SNR=30.0 dB. Histograms for all 25 top-ranked variables are included in Appendix B chapter for completeness and show that this non-Gaussian shape is prevalent in each. The dataset at this SNR is chosen here as it contains the least amount of additive noise. The histograms clearly illustrate the distinct non-Gaussian distribution of ZigBee features being considered. Parametric classifiers such as MDA/ML that inherently rely on Gaussian distributions will yield large estimation error and degraded classification performance. Non-parametric and non-linear classifiers such as GRLVQI, RndF and MCA are expected to perform better than MDA/ML on variables with irregular distributions such as these.

Figure 6.8 shows $\%C$ vs. SNR for N_D=4 authorized ZigBee devices using MDA/ML, RndF, Multi-Class AdaBoost (MCA) and GRLVQI classifiers using the top 25 variables selected by each of the three VI methods described in Section 6.3.3 (KS, GRLVQI, and RndF). While more than half of the 47 variables (29/47) were uniquely identified by one particular VI selection method (i.e., they were not found by any other VI method), classification results were similar for each of the four classifiers regardless of variable selection method.

Results show that RndF and MCA ensemble methods generally outperform MDA/ML and GLRVQI methods for all SNR considered, with MCA being slightly better than RndF. The arbitrary $\%C$>90% benchmark is achieved for RndF and MCA ensemble methods at SNR=12.0 dB, while GRVLQI and MDA/ML methods require SNR=18.0 dB and SNR=30.0 dB, respectively. The benefit of ensemble methods is reflected in a performance "gain" of approximately G_S=6.0 dB (GRLVQI) and 18.0 dB (MDA/ML). Here gain is defined as the reduction in required SNR, expressed in dB, for two methods to achieve a given $\%C$ performance. Viewed in a manner consistent with expectations for an operational scenario with fixed SNR conditions, the RndF and MCA $\%C$ performance is 9%-19% better than MDA/ML and 3%-20% better than GRLVQI for the range of SNR



(a)



(b)

Figure 6.7: (a) Random Forest and MCA test error performance at *SNR*=0.0 dB, and (b) representative histograms for four arbitrarily selected variables showing distinct non-Gaussian distributions.

considered. Collectively, these results suggest that ensemble methods are more robust

means for correctly authenticating ZigBee devices on networks operating in much noisier environments.

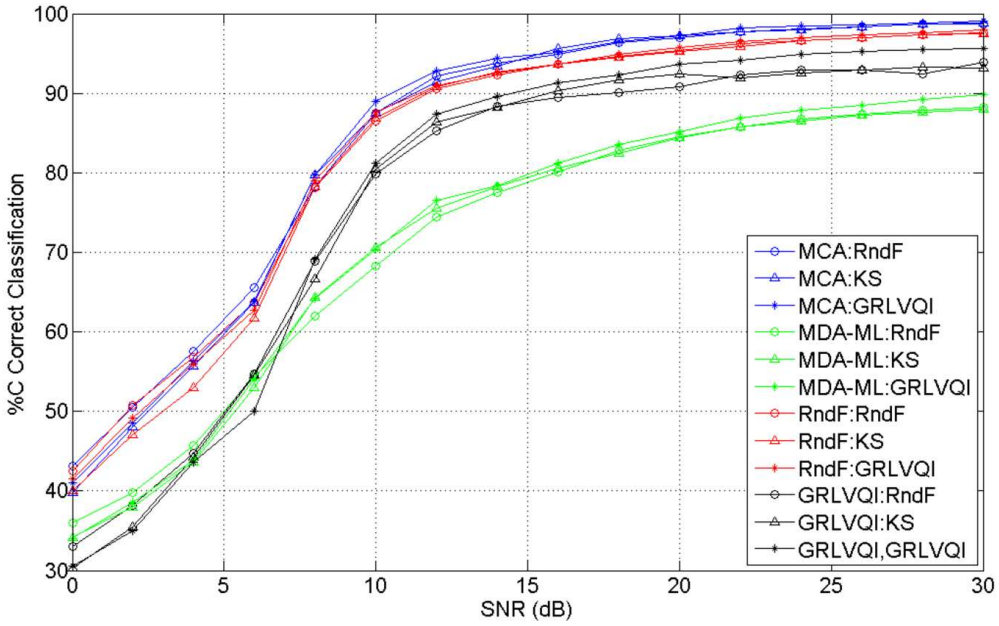


Figure 6.8: Percent correct classification ($\%C$) versus SNR for all classifiers using top-ranked 25 variables selected by each of the three VI methods as indicated. Legend A:B indicates results for classification Method A using variables selected by Method B.

6.3.5 RF-DNA Verification Results

Once a given classifier has been trained on *authorized* network devices, it can be used to identify *unauthorized* rogue devices attempting to gain network access. In this case, rogue devices impersonate authorized devices by presenting false bit level credentials matching those of an authorized device a common approach for spoofing attacks. Rogue detection and rejection presents considerable challenge given their emissions were not available during classifier training. This was successfully addressed in [25, 35] using a biometric-based verification process with a similarity measure (test metric) reflecting a given rogue device "looks like" each of the authorized devices. For

related experiments here, four authorized ZigBee devices are used for classifier training according to Section 6.3.4 and nine previously unseen rogue devices are introduced to gauge classifier verification performance. Test metrics provide a level of confidence in the classifiers decision and are generated for each rogue device. For MDA/ML and RndF assessment, the test metrics included posterior probability estimates for each rogue device. For GRLVQI assessment, the combined angle-distance metric in [106] is used. The weighted voting scheme of MCA does not support generation of a convenient test metric and its verification performance is not assessed.

For results here, each of the nine rogue devices are presented as having a claimed bit-level identity matching each of the four authorized devices, for a total of 36 rogue assessment scenarios. For each rogue-authorized device pair, 1000 bin histograms were generated using test metrics reflecting 1) how much the rogue device "looks like" the authorized device (Rog:Auth), and 2) how much the authorized device "looks like" itself (Auth:Auth). A representative rogue-authorized histogram pair is shown in Figure 6.9 where the span of histogram bin values is determined by the authorized device test metric values.

Histograms such as shown in Figure 6.9 are used to generate Receiver Operating Characteristic (ROC) curves by varying a threshold (left-to-right) across the distributions and calculating True Verification Rate (TVR) and False Verification Rate (FVR). TVR reflects a correct decision whereby an authorized device is present and correctly granted network access. FVR corresponds to Rogue Accept Rate (RAR) and reflects an incorrect decision whereby a rogue device is present and errantly granted network access. The resultant ROC curve for a given rogue-authorized device pair is simply generated by plotting FVR vs. TVR, with higher TVR and lower FVR points reflecting better performance. ROC curve assessments here are based on an arbitrary benchmark performance range defined by TVR>0.9 and FVR<0.1. For rogue-authorized scenarios that

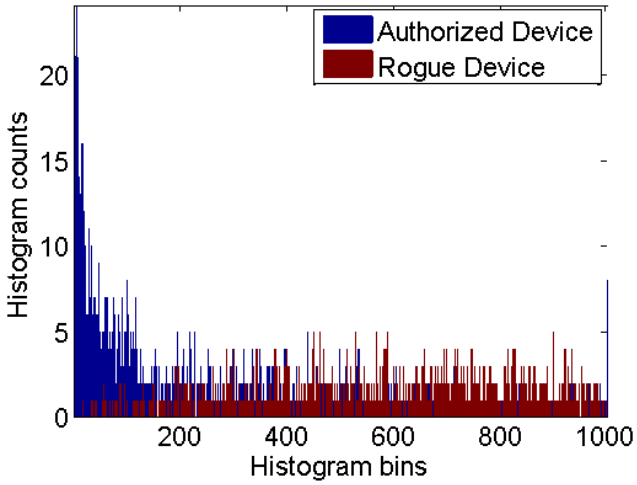


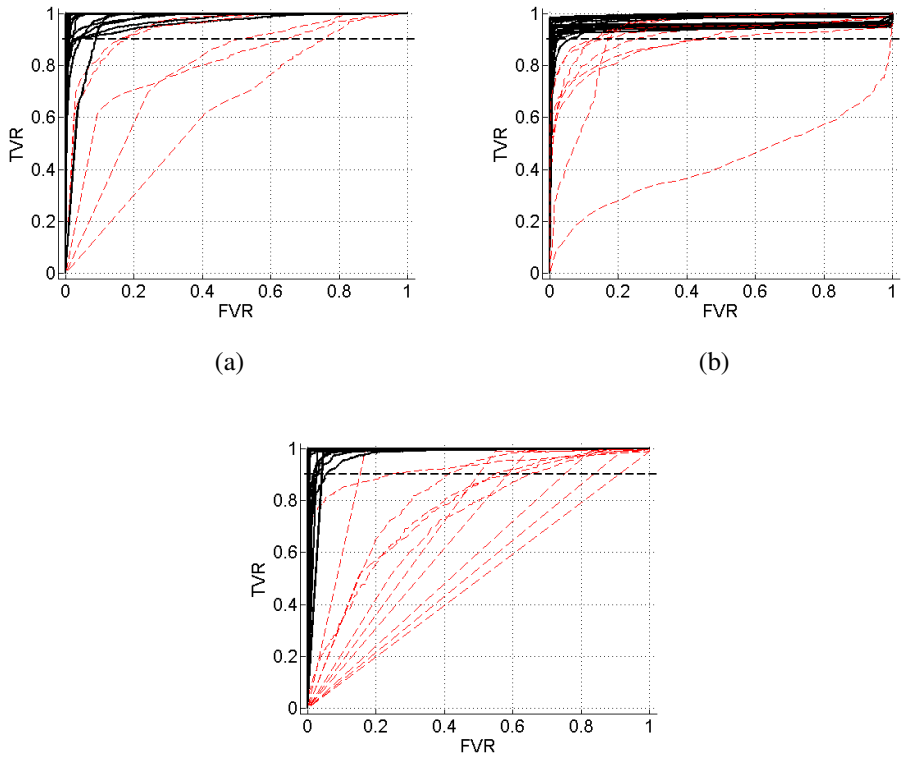
Figure 6.9: Representative histograms of MDA/ML posterior probability test metrics for a rogue-authorized device pair at *SNR*=12.0 dB. Legend A:B notation indicates test statistics reflecting how much Device A looks like Device B where Device B is the claimed ID.

do not meet these criteria, the rogue is deemed to have successfully "spoofed" the network. Verification ROC curves are presented here using the top 25 ZigBee features selected by GRLVQI at *SNR*=10.0 dB. Results for both RndF and KS-Test feature selection were also investigated and yielded similar results.

6.3.5.1 Fixed Correct Classification Performance

The three classifiers were trained using data sets yielding equivalent benchmark classification performance of $\%C$=90%. Using classifiers with equal classification performance in this way allows a normalized comparison of network intrusion detection performance by rogue ZigBee devices. From Figure 6.8, this required using RndF trained at *SNR*=12.0 dB, GRLVQI trained at *SNR*=18.0 dB, and MDA/ML trained at *SNR*=30.0 dB. The top 25 ZigBee features selected by GRLVQI at *SNR*=10.0 dB were used by each classifier. Figure 6.10 shows resultant ROC verification performance for each of the three classifiers at $\%C$=90%. Solid lines represent rogue scenarios satisfying the TVR>0.9 and

FVR<0.1 benchmark (spoofing thwarted) and red dashed lines represent those that did not (spoofing successful). As shown, RndF processing correctly rejected 31 of 36 rogue access attempts, which exceeds by MDA/ML (20 of 36) and GRLVQI (25 of 36).



(a) (b)

(c)

Figure 6.10: Receiver Operating Characteristic (ROC) curves showing verification performance for 36 total rogue scenarios using the GRLVQI identified best 25 input features with fixed $\%C$=90% using (a) RndF at SNR=12.0 dB, (b) GRLVQI at SNR=18.0 dB, and (c) MDA/ML at SNR = 30.0 dB. Solid lines represent rogue assessment scenarios satisfying the TVR>0.9 and FVR<0.1 benchmark (spoofing thwarted) and red dashed lines represent scenarios that do not (spoofing successful).

Similar verification tests were run using the top 25 input features as identified by VI methods RndF and KS-Test at SNR=10.0 dB. The number of rogue scenarios that

were correctly detected satisfying the TVR>0.9 and FVR<0.1 benchmark are shown in Table 6.2. This table shows that GRLVQI identified features provide the highest verification performance, as determined by the highest number of rogue scenarios successfully detected. In addition, for each VI method with equal correct classification performance, RndF was able to identify the most number of rogue scenarios.

Recalling from Section 6.3.4 that RndF processing provides approximately G_S=6.0 dB (GRLVQI) and 18.0 dB (MDA/ML) of "gain" at equivalent $\%C$=90%, RndF provides enhanced network intrusion detection capability of rogue ZigBee devices in a much noisier environment; a considerable advantage in real-world RF environments where high SNR may not be realized for ZigBee applications.

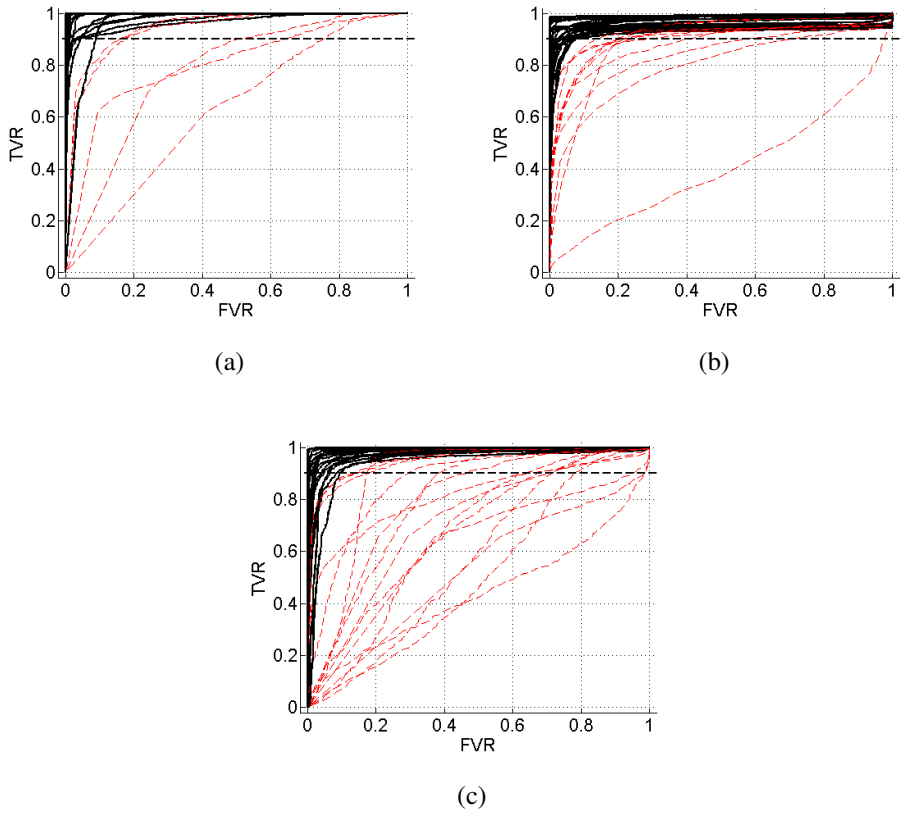
Table 6.2: Number of rogue scenarios out of 36 total that were correctly rejected based on the arbitrary TVR>0.9 and FVR<0.1 benchmark for fixed $\%C$=90% correct classification performance

Classifier	Variable Importance Method		
	RndF	KS-Test	GRLVQI
RndF	30	29	31
GRLVQI	24	24	28
MDA/ML	20	25	25

6.3.5.2 Fixed SNR Performance

Verification performance was also investigated using a fixed SNR for all three classifiers; this is consistent with real-world environments where receiver and background noise power is fixed and verification is required at a given location. Figure 6.11 shows results for fixed SNR=12.0 dB data sets using all three classifiers and the same top-ranked 25 variables from GRLVQI used for fixed $\%C$ verification results in Section 6.3.5.1. RndF processing is again superior and correctly rejects 31 of 36 rogue spoofing attempts at the

TVR>0.9 and FVR<0.1 benchmark; only 25 of 36 (GRLVQI) and 20 of 36 (MDA/ML) rogue spoofing attempts are successfully rejected.



(a) (b)

(c)

Figure 6.11: Receiver Operating Characteristic (ROC) curves showing verification performance for 36 total rogue scenarios using the GRLVQI identified best 25 input features with *fixed SNR*=12.0 dB using (a) RndF at *%C*=90%, (b) GRLVQI at *%C*=86%, and (c) MDA/ML at *%C*=75%. Solid lines represent rogue assessment scenarios satisfying the TVR>0.9 and FVR<0.1 benchmark (spoofing thwarted) and red dashed lines represent scenarios that do not (spoofing successful).

As in Section 6.10, verification tests were run using the top 25 input features as identified by VI methods RndF and KS-test at *SNR*=10.0 dB. The number of rogue scenarios that were correctly detected satisfying the TVR>0.9 and FVR<0.1 benchmark

are shown in Table 6.3. Similar results are seen, except the verification performance of RndF over the other two classifiers is even more pronounced using fixed SNR conditions.

The superiority of RndF processing in this lower SNR environment is even more significant considering average correct classification results at SNR=12.0 dB in Figure 6.8, which includes $\%C \approx 90\%$ (RndF), $\%C \approx 86\%$ (GRLVQI), and $\%C \approx 75\%$ (MDA/ML); once again, a considerable advantage in real-world RF environments where high SNR may not be realized for ZigBee applications.

Table 6.3: Number of rogue scenarios out of 36 total that were correctly rejected based on the arbitrary TVR>0.9 and FVR<0.1 benchmark for fixed SNR=12.0 dB channel conditions.

Classifier	Variable Importance Method		
	RndF	KS-Test	GRLVQI
RndF	30	29	31
GRLVQI	17	18	25
MDA/ML	11	20	25

6.4 Conclusion

The proliferation of ZigBee devices in commercial and military infrastructures has increased the urgency for improving protection. Attacks involving rogue devices that impersonate authorized devices by presenting false bit-level credentials can be especially devastating. Relative to traditional bit-level protection, RF-DNA fingerprinting provides a PHY layer defensive measure that can be employed against these attacks. The work presented here extends previous PHY fingerprinting work using 1) statistical RF-DNA features extracted from ZigBee emissions collected under real-world conditions and possessing non-Gaussian attributes due to multipath reflections and other device

interference, and 2) non-parametric Random Forest (RndF) and Multi-Class AdaBoost (MCA) ensemble classifiers that improved classification by 9%-19% over MDA/ML and 3%-20% over GRLVQI across the range of SNR considered. Furthermore, the better trained ensemble classifiers significantly enhanced device ID verification performance when unknown rogue devices were presented to the network. For operation at a *fixed* $\%C$=90% correct classification accuracy, RndF correctly detected 31 of 36 rogue scenarios at SNR=12.0 dB. This exceeds GRVLQI rogue detection performance which detected 28 of 36 rogue scenarios at SNR=18.0 dB and MDA/ML which detected 25 of 36 rogue scenarios at SNR=30.0 dB. Considering the SNR differences for *fixed* $\%C$=90% operation, RndF emerged as the superior alternative and provided a "gain" of G_S=18.0 dB and G_S=6.0 dB relative to MDA/ML and GRLVQI processes, respectively. Alternately, for operation under *fixed* SNR conditions, RndF ID verification processing detected 11 more rogue scenarios than MDA/ML and 6 more rogue scenarios than GRLVQI. In real-world applications where RF emissions crowd the 2.4 GHz spectral region, the SNR is relatively low and ensemble methods are envisioned as providing a more robust alternative for authenticating ZigBee device IDs and enhancing network security.

6.5 Multi-Receiver Data Set Evaluation

Experiments conducted with the ZigBee *Cross-Environment* datasets evaluated the performance of RF-DNA fingerprinting to classify and verify authorized ZigBee devices using an advanced high-cost Agilent receiver. A further evaluation was conducted to determine the cost-performance trade-off of using a low-cost receiver instead. Presented in the following sections is the summary of results which are under review at the *Military Communications Conference MILCOM 2014* [95]. Certain sections from [95] have been omitted here to avoid redundancy with previous sections in this chapter.

6.6 Introduction

ZigBee devices using the IEEE 802.15.4 standard for Wireless Personal Area Networks (WPAN) have found wide-spread adoption in a variety of applications, from security systems [126], to health-care [58], to industrial and building control [38]. Their increased popularity in sensitive and high-value areas has understandably led to security and vulnerability concerns. ZigBee security is provided through the Advanced Encryption Standard (AES) for MAC, NWK and APS layers. Security researchers however have revealed several methods to exploit vulnerabilities in the key-exchange process [33, 99, 127]. In addition, it has been proven that power consumption of an authorized wireless sensor node can be passively monitored to determine the secret encryption key through Side-Channel Analysis (SCA) methods [76, 108]. The ZigBee alliance has countered these threats by adopting AES in either Counter (AES-CTR) or AES-CCM* mode which are more resistant to SCA attacks. However, AES-CTR mode has theoretically and practically been proven vulnerable to SCA eavesdropping attacks in [57] where the full AES-CTR key was successfully recovered. Thus key recovery in ZigBee devices is entirely possible which easily allows an attacker to insert rogue devices within an existing network.

RF-fingerprinting offers an intriguing PHY layer countermeasure to spoofing attacks and rogue device identification. Physical differences between devices manifest themselves into measurable differences in their transmitted signals. RF-fingerprinting uses machine learning methods to exploit these differences and reliably identify wireless devices [48, 96, 102, 105, 107]. RF-fingerprinting is generally considered a robust countermeasure as it uses random physical variations between devices for identification, which are prohibitively complex for an attacker to duplicate.

Most RF-fingerprinting methods require the use of large, expensive receivers such as the Agilent E3238S [94], or high-speed oscilloscopes [30, 96]. Recent research has shown success in RF-fingerprinting through the use of low-cost USRP receivers [87, 105].

However, a comparison between high and low cost receivers for performance trade-offs has not been performed to date. This research bridges this information gap by comparing performance differences in RF-fingerprinting between a high-cost receiver and a low-cost USRP-based receiver. Six devices of the same manufacturer and model type are used for testing. This model uniformity represents the most challenging case for device identification, and performance will likely only improve for cases when device models differ. The Random Forest (RndF) classifier is used to investigate amplitude, phase and frequency features and their respective contributions toward classification. Performance is characterized by correct classification rate (%C) for device identification, and number of Rogue device scenarios identified for network intrusion detection.

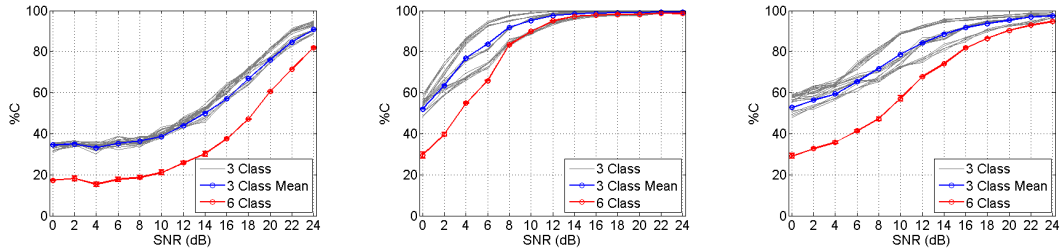
The remainder of this paper is organized as follows: Section 4.2 provides background information on RndF and feature generation. Section 6.7 describes the methodology for data collection, and device authentication and verification. Section 6.8 provides experimental results and analysis for experiments on device authentication and verification. Finally, our conclusions from this work are detailed in Section 6.8.

6.7 Methodology

RF-DNA fingerprinting methodology for data collection, device authentication and verification used here is similar to those used in Sections 6.2.1, 6.3.4 and 6.3.5. Differences in methodology are presented in the following sections.

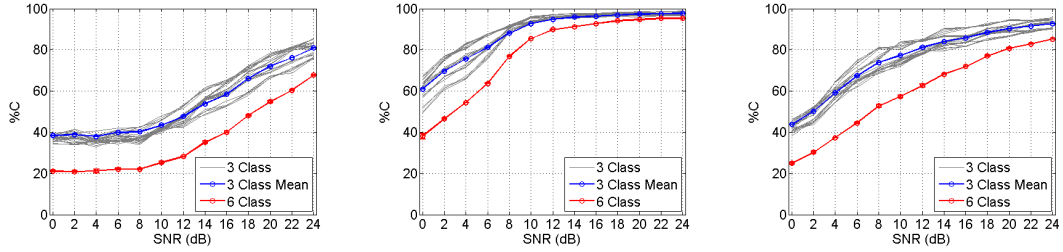
6.7.1 Data Collection and RF-DNA Fingerprint Generation

The data set originally defined in [121] is used in this research. The high-cost receiver used for data collection is the National Instruments NI PXIe-1085 system and the low cost receiver is the NI USRP-2921. Six Atmel RZUSBsticks are used as the transmitters. In order to provide a meaningful comparison, as many parameters as possible are kept constant during signal collection. Both receivers collect observations simultaneously and



(a) (b) (c)

Figure 6.12: Classification results for $SNR \in$[0,24]dB using the *PXIe Receiver* with (a) amplitude, (b) phase, and (c) frequency RF-DNA statistical features.



(a) (b) (c)

Figure 6.13: Classification results for $SNR \in$[0,24]dB using the *USRP Receiver* with (a) amplitude, (b) phase, and (c) frequency RF-DNA statistical features.

are at equal distance from the transmitter. Transmit power is set to 1 mW and a common VERT2450 receiver antenna with 3dBi gain is used for both receivers.

Both systems record I/Q data as 16-bit integers, sampled at 20 Msps. 600 transmission preambles are sampled from each RZUSBstick. Transmission detection from background noise is accomplished through amplitude-based leading edge detection using a -6dB threshold. As outlined in the IEEE 802.15.4 standard [56], the first 128 μs of each transmission constitutes the preamble. At 20 Msps the first 2560 I/Q samples span the preamble region of each transmission. The transmitter operating frequency is IEEE 802.15.4 channel 26 (2.480 GHz) for all collections to mitigate nearby IEEE 802.15.4

traffic (2.401-2.473 GHz). The collected SNR is approximately $SNR_C \approx$30 dB for PXIe and $SNR_C \approx$24 dB for the USRP.

Inter-device variability in USRP RF-fingerprints was previously noted in [104]. To mitigate possible variability in collection center frequency due to clock skew, the collection receiver center frequency was set 3 MHz below the transmission center frequency (2.477 GHz versus 2.480 GHz). The collected transmission was then down-converted to baseband using gradient-based frequency estimation performed with Mathworks Matlab software. A W_{BB}=1 MHz 8th-order Butterworth filter was used to remove background noise outside the IEEE 802.15.4 channel, resulting in a low-noise, baseband representation of the collected transmission. Background noise filtering was not discussed in [104], which may have contributed to the erratic RF fingerprinting performance reported therein. Given the relatively small W_{BB} bandwidth, the collected signals were downsampled by one-half to 10 Msps which satisfies Nyquist sampling criteria while reducing computational burden.

After down-sampling, 1280 preamble samples were converted into three 1280 length sequences of instantaneous amplitude ($a[n]$), phase ($\phi[n]$) and frequency ($f[n]$). These sequences were divided into N_R=32 equal length sub-sequences corresponding to the 32 preamble bits. Including the original full-length sequences for fingerprint features, and considering three RF-DNA statistical features (σ^2), (γ) and (k), there were a total of $(N_R + 1{=}33) \times 3 \times 3{=}297$ RF-DNA statistical variables used for assessment.

6.7.2 Device Authentication

Device authentication is the classification of authorized devices with RF-DNA fingerprinting variables. In order to assess authentication performance in high and low noise environments, the collected signals were added to like-filtered Additive White Gaussian Noise to achieve the desired $SNR \in$[0,24] dB in 2dB steps. All 6 devices are classified with the Random Forest algorithm at each SNR, with 300 observations each used for training and testing sets. Further rogue identification tests in Section 6.7.3 were based

on random selection of three *authorized* devices with the remaining devices designated as *rogue* devices; each possible selection of 3 devices from the population of 6 devices (denoted as ${}_6C_3$) is also evaluated for classification accuracy. Given the random nature of RndF, 5-fold cross-validation was used to ensure statistical confidence in the results. For all experiments in this research, the RndF consisted of an arbitrarily chosen N=1000 decision trees.

6.7.3 Device Verification

Once a given classifier has been trained on authorized network devices, it can be used to identify unauthorized rogue devices attempting to gain network access in a process termed *device verification*. Rogue devices impersonate authorized devices by presenting false bit-level credentials matching those of an authorized device - a common approach for spoofing attacks. Rogue detection and rejection presents considerable challenge given their emissions are not available during classifier training. This was successfully addressed in [25, 35] using a biometric-based verification process with a similarity measure (test metric) reflecting how much a given rogue device "looks like" each of the authorized devices. For related experiments here, three ZigBee devices are randomly chosen as authorized devices for classifier training. The remaining three unseen rogue devices are introduced to the classifier to gauge verification performance. Test metrics provide a level of confidence in the classifiers decision and are generated for each rogue device. For RndF assessment, the test metrics are the posterior probability estimates for each rogue device. Each combination of ${}_6C_3$ are tested to generalize results for the entire population of devices tested.

We define 1) how much the rogue device "looks like" the authorized device as *Rog:Auth*, and 2) how much the authorized device "looks like itself as *Auth:Auth*. As described in [35] test metrics generated by the classifier for an authorized device are used to generate a 1000 bin histogram (Auth:Auth). These bins from an authorized device are

used with the test metrics of the rogue device to generate a new histogram (Rog:Auth). With these histograms, traditional ROC processing is used to calculate the True Verification Rate (TVR) and False Verification Rate (FVR). Rogue devices achieving the arbitrarily chosen thresholds of TVR>0.9 and FVR<0.1 on the ROC plot are designated to have been correctly identified by the classifier. Devices that do not meet these thresholds are designated to have successfully spoofed the network. Three rogue devices impersonating each of the 3 authorized devices give 9 unique Rog:Auth scenarios. The number of rogue scenarios correctly identified is recorded for each case.

6.8 Results

6.8.1 Device Authentication Results

Figure 6.12 shows authentication results for the data collected with the PXIe receiver. Results for each combination of randomly selected 3 authorized devices (${}_6C_3$=20 combinations) are shown (3-class), as well as the mean for all 20 combinations. Also shown are the authentication results for the full 6 device population (6-class). Results for this 6-class set are lower than those for the 3-class case as more classes tend to crowd the variable space, making classification more difficult. As each measurement is repeated for 5-fold cross-validation, 95% confidence intervals over the 5 folds are plotted for the 3-class and 6-class mean. The variation across the 5-folds is very small causing the error bars to be smaller than the marker size.

RF-DNA fingerprint phase variables were found to be the most useful for device authentication, achieving a correct classification rate %C>90% on average for the 3-class cases at SNR≥8 dB, and for the 6-class case at SNR≥10 dB.

In a similar fashion, USRP results are shown in Figure 6.13. Again, RF-DNA fingerprint phase variables are the most useful for device authentication, reaching %C>90% benchmark at SNR≥10 dB. For the 3-class case, USRP authentication performance differs from PXIe performance between $\%C$=-8.87% to 3.75% over all SNRs tested, with an

average difference of 0.76%. Negative %C differences represent cases where classification is more successful with USRP versus PXIe collected data at that SNR. For the 6-class case where all ZigBee devices in the population are tested, %C differences between PXIe and USRP collected data range from -8.46% to 6.58%, with an average %C difference across all SNRs tested of 2.22%. Thus for both 3-class and 6-class cases, there is slightly lower authentication performance when using the lower cost USRP-based receiver.

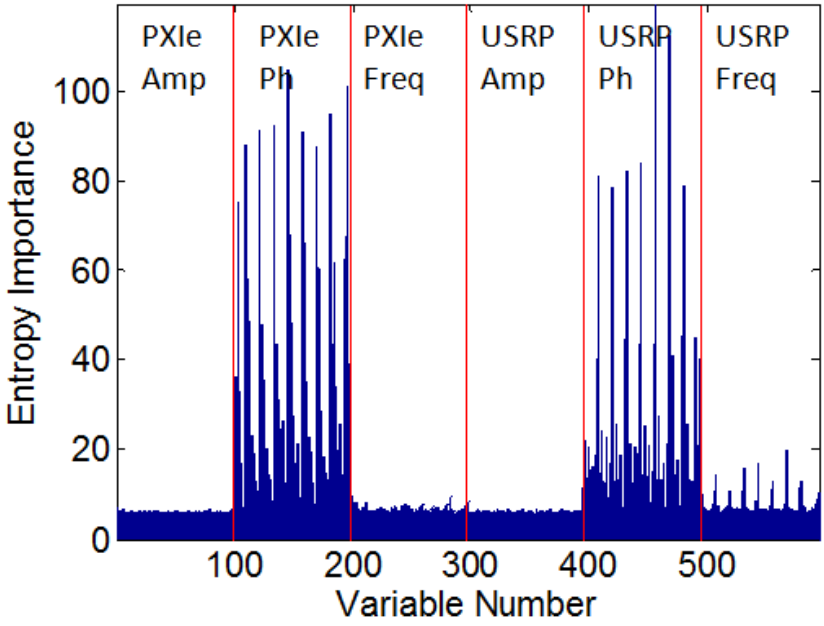


Figure 6.14: RFEI of PXIe and USRP variables. Results indicate phase variables are most important for classification, with PXIe phase variables being more important than USRP phase variables.

Random Forest Entropy Importance (RFEI) is used to determine the relative importance of USRP and PXIe collected variables. Data from each receiver at SNR=10 dB is used, as device authentication results show that both receivers achieve >90% %C at this SNR. The observations from each receiver at this SNR are concatenated such that the first 297 RF-DNA variables of each observation are from the PXIe receiver and the

last 297 RF-DNA variables are from the USRP receiver. A Random Forest of 1000 trees is grown for the 6 classes using the full 600 observations to determine relative variable importance, as shown in Figure 6.14. Merging the data from both receivers in this way gives Random Forest access to both data sets simultaneously, allowing it to jointly determine how important each receiver is toward device classification. Results show that the phase variables for both receivers are the most important, with Random Forest favoring the PXIe phase variables slightly more than the USRP phase variables. Judging by the number of peak important variables greater than an arbirtrary chosen threshold of 50, Random Forest favors the PXIe phase variables slightly more than the USRP phase variables. Additionally, USRP frequency variables are found more useful than the PXIe frequency variables. Finally, it is evident that for both receivers, amplitude variables are relatively unimportant for RndF classification. Given the high relative variable importance for phase variables, only phase variables are subsequently considered device verification tests.

6.8.2 Device Verification Results

Verification tests are conducted by randomly assigning three ZigBee devices as authorized and the remaining three as rogue. All ${}_6C_3$=20 such combinations are tested to determine how many rogues are correctly identified by data collected by each receiver. Data sets are chosen at the SNR at which average 3-class device authentication results in Section 6.8.1 achieve %C>90%. Only for RF-DNA fingerprint phase variables are considered. For the PXIe receiver, data at *SNR*=8 dB is used, while for the USRP data at *SNR*=10 dB is used. Figure 6.15 shows the number of rogue scenarios correctly identified with the ROC identification threshold corresponding to TVR>0.9 and FVR<0.1. Results are averaged over 5-folds to account for performance differences from the randomness in Random Forest.

Further investigation reveals that some devices have very similar hardware properties, making their corresponding Rog:Auth combinations difficult to differentiate. This is

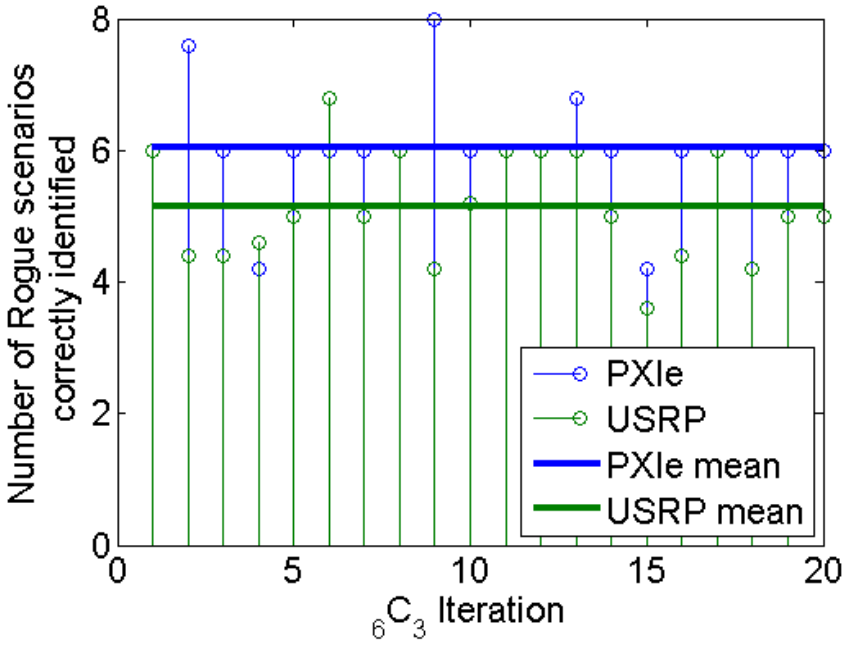


Figure 6.15: Number of rogue scenarios correctly identified out of a maximum possible 9 (averaged over 5-folds). A total of 20 rogue-authorized device combinations are tested. Average identified rogue scenarios over all combinations for each receiver are shown in bold.

evidenced for example in iteration 15 in Figure 6.15, where the average number of rogue scenarios correctly identified is a low 3.6 out of a possible 9. The number of times these Rog:Auth scenarios are correctly identified over 20 combinations is accumulated and averaged over 5 folds for each receiver data set in Tables 6.4 and 6.5. Several Rog:Auth combinations are found to have very low success rates. For example, the Rog:Auth combination of Dev1:Dev4 (row 4, column 1) is not correctly identified with either the PXIe or USRP receivers for fold. In this case, low success rate is more due to the similarity in their hardware rather than caused by receiver accuracy or classifier performance. These similarities are to be expected when testing the hardest case scenario where all devices are from the same model type, as was done here. On the other hand, certain cases such

Table 6.4: Average percentage of Auth:Rogue scenarios out over ${}_6C_3 = 20$ combinations and 5 folds that are correctly identified based on ROC TVR>0.9 and FVR<0.1 performance for the NI PXIe receiver.

		Rogue					
		Dev1	Dev2	Dev3	Dev4	Dev5	Dev6
Authorized	Dev1	*	100%	100%	0%	100%	100%
	Dev2	100%	*	60%	100%	70%	0%
	Dev3	87%	83%	*	97%	0%	83%
	Dev4	0%	100%	100%	*	100%	100%
	Dev5	50%	50%	0%	50%	*	50%
	Dev6	100%	0%	70%	97%	67%	*
	Avg	67%	67%	66%	69%	67%	67%

Table 6.5: Average percentage of Auth:Rogue scenarios out of over ${}_6C_3$=20 combinations and 5 folds that are correctly identified based on ROC TVR>0.9 and FVR<0.1 performance for the NI USRP receiver.

		Rogue					
		Dev1	Dev2	Dev3	Dev4	Dev5	Dev6
Authorized	Dev1	*	100%	100%	40%	80%	100%
	Dev2	100%	*	0%	100%	63%	0%
	Dev3	83%	0%	*	83%	0%	83%
	Dev4	0%	100%	97%	*	20%	100%
	Dev5	50%	53%	0%	13%	*	50%
	Dev6	100%	0%	46%	100%	50%	*
	Avg	67%	51%	31%	67%	43%	67%

as Dev2:Dev1 are correctly identified in all combinations for both receivers. In this case the Rogue devices differ greatly from the authorized devices, making rogue identification highly successful. Again, this success is due to device hardware differences and does not depend on the receiver accuracy.

On average, the PXIe receiver collected data correctly identified 67.11% of rogue scenarios, versus 57.11% rogue scenarios with the data from the USRP receiver. The 95% confidence intervals are ±18.55% and ±19.33% respectively, indicating that the rogue identification performance for the two receivers is not statistically different at this alpha level. These verification results show in a practical sense the performance-to-cost trade-off from using a lower cost USRP receiver at the risk of more rogue devices spoofing the network by impersonating authorized devices.

6.9 Conclusion

This research provides a comparison of RF-DNA fingerprinting performance using classification variables generated from high-cost PXIe and low-cost USRP receiver collections. By testing all combinations of ZigBee devices in a given population, biased results from favorable device selection are avoided, and results can be generalized for the entire population. RFEI results show that the process favors RF-DNA phase variables over amplitude or frequency variables. PXIe-collected RF-DNA phase variables are given higher Entropy Importance relative to corresponding USRP-collected phase variables. Device authentication results show an average $\%C$=0.76% higher classification when using PXIe collected RF-DNA phase variables over USRP for the 3-class case over all SNRs tested, and $\%C$=2.22% higher for the 6-class case. Device verification results show an average of 10% more Rog:Auth scenarios being identified with the PXIe collected data versus the USRP-collected data. Thus, as expected, performance is degraded when using the lower cost receiver. However, the performance decrease is not statistically significant at the 95% confidence level. In addition, the magnitude of performance drop is small and may

be acceptable given the much lower cost of the USRP receiver. These experiments consider the most-challenging case when all ZigBee devices are from the same manufacturer and model number. Results are expected to improve for both receivers when using rogue devices that are of a different model type than the authorized devices.

VII. Conclusions and Future Work

This chapter summarizes the key doctoral research contributions and provides several recommendations for future research.

7.1 Research Summary

The cyber-physical realm offers unique opportunities for both offensive and defensive measures that cannot be implemented in software-only systems. The field of Side Channel Analysis (SCA) offers unique offensive opportunities to extract the secret key from a microcontroller through the aid of side-channel leakage. This leakage can only be masked but not eliminated given that it is a by-product of the physical implementation of the cryptographic algorithm in hardware. In contrast, RF-Distinct Native Attribute (RF-DNA) fingerprinting exploits physical differences in transceiver hardware to uniquely identify trusted RF hardware in a distributed network. These physical differences are random and unique to each microcontroller, manifesting themselves as slight errors in their transmitted signals. They are naturally generated and nearly impossible for an attacker to duplicate. As such, they provide a high quality authentication method for identifying authorized devices on distributed networks, and conversely prevent unauthorized devices from entering the network.

This research advances the fields of SCA and RF-DNA fingerprinting through three main contribution areas: 1) Linear Regression Attack (LRA), 2) Random Forest (RndF) SCA Profiling Attacks, and 3) RF-DNA Fingerprinting of ZigBee devices with RndF. Main contributions in each area are described in the following sections.

7.1.1 Linear Regression Attack

A novel method was developed that combined the Linear Regression SCA attack with the adjusted coefficient of determination R_a^2 [92]. By using R_a^2 to determine the quality of

the fit of the model estimate to the collected SCA data, better estimation of the multivariate noise distribution was achieved. This new method was compared with four others from literature. These methods differed only in how they determined the fit of the linear model to the measured data. Previous work used metrics that are not known to be good indicators of a good model estimate fit to the measured data, such as the L^2-norm of the model parameters [115], and the largest absolute parameter value [50]. R_a^2 gives an indication of the variance unaccounted for by a linear model estimate. By using a metric that is more relevant to measuring the fit of data to linear model estimates, greater success was attained.

A first ever assessment of linear regression conditions was conducted verifying the conditions of linearity, homoscedasticity, normality, independence, as well as conducting outlier assessment, effect of interaction terms and importance of using the intercept in the model. Results revealed that the effect of outliers was diminished as the number of traces increases. Additionally, the intercept b_0 in the model is necessary for proper fit, which is in contrast to experiments in [49] where the intercept was not used.

This method was compared with 4 others from literature under varying conditions of test set size, training set size, and number of variables. In 8 out of 9 cases, the new method achieved the highest success rate. Attack performance showed an order of magnitude improvement when the dimensionality of the distribution estimated in the training phase was increased from 1 to 20, giving greater than 98% success rate with as few as 100 training and test traces. The attack with R_a^2 was proven successful when training and test data were collected with different probes and when training and attack hardware differed, showing good performance under real-world conditions. It was also found to be more successful at extracting key bytes from a noisier environment, achieving success with as few as 50 test traces at Signal to Noise Ratio (SNR)=15 dB. Finally, the R_a^2 metric was theoretically and practically shown to be the best out of those tested at identifying time samples with high information leakage, even when the number training traces and test was very low.

7.1.2 Random Forest SCA Profiling Attacks

Side-channel theory has historically assumed a Gaussian noise model around a fixed data leakage value in the measurement. Through this work with 40 PIC microcontrollers, it is shown in [91] that non-Gaussian variables can be present in EM collected traces and these can warrant the use of non-parametric classifiers. RndF was used for the first time in a byte-wise profiling attack and compared to the traditional parametric Template Attack [24].

The property of RndF to handle high dimensionality well was used to solve the case when the attacker does not know when the information is being leaked by the hardware. In this case, the full 50,000 variable set was used by RndF to extract all 16 bytes of the AES key when the training and test devices were the same, and 15/16 when they were different. When coarse data reduction was used with a 3200 variable set, RndF was still able to achieve success rates as high as 100% on cross-device attacks with 40 PIC microcontrollers. Template Attack was not able to extract any key bytes under any cross-device combination. Random Forest Entropy Importance (RFEI) was utilized to define time samples where there was information leakage, and this method was compared to Sum-Of-Squared pairwise T-difference (SOST) which in literature was found to be favorable to Template attack. Variable reduction with RFEI found many non-Gaussian variables to be useful for classification, while SOST ranked Gaussian variables higher. RndF for both cases was better at generalizing during training to be able to attack devices that are more physically dissimilar to each other, achieving as high as two-fold performance improvement over Template Attack. Template Attack performance was also found to be robust when only non-Gaussian variables were used for clarification, achieving as high as 98.31% despite having corresponding misclassification rates as high as 99.41%.

Further study into the effect of data collection location on the chip (probe spatial variation) was performed by scanning the PIC microcontroller over a grid of 5×25 locations with 5,000 traces collected at each location. The number of non-Gaussian variables was

counted at each location. Correlation Electro-Magnetic Attack (CEMA) attack was used for comparison to reveal the relative usefulness of each location for gathering SCA collections. Results showed two SCA "hot-spots" for collection - one near wires connecting the CPU die to the external pins, and another near a possible voltage regulating capacitor. However, non-Gaussian variables were found to be present only over the wires and not over the capacitor. Despite best efforts to duplicate the hardware setup used to collect the original 40 PIC SCA dataset, the same number of non-Gaussian variables were not discovered in this research.

7.1.3 RF-DNA Fingerprinting on ZigBee Devices with Random Forest

ZigBee distributed networks have become increasingly popular for a wide range of applications, from building control, to healthcare systems and critical infrastructure. However, their remote nature requires enhanced security to protect them from rogue devices attempting to gain unauthorized network access. In this work, RndF is combined with RF-DNA fingerprinting to further the state of the art by improving performance in high-noise environments for authenticating authorized devices and identifying rogue devices. Two main bodies of work were introduced that explored data collected in two primarily different ways, including 1) the Cross-Environment (XEnv) dataset based on emissions collected with a Agilent E3238S receiver and 2) the Cross-Receiver (XRx) dataset based on emission collections with a NI USRP and PXIe receivers.

The RndF and Multi-Class AdaBoost (MCA) ensemble decision classifiers were used in [93, 94] with the XEnv dataset. This dataset models the RF-DNA fingerprints from a device captured under multiple environments, which relaxes the contraints on the test collections. However, this also leads to multi-modal or non-Gaussian variables, which can be properly utilized by non-parametric RndF and MCA ensemble classifiers. RndF and MCA improved classification by 9%-19% over MDA/ML and 3%-20% over GRLVQI across the range of SNR considered. Furthermore, the better trained ensemble classifiers

significantly enhanced device ID verification performance when unknown rogue devices were presented to the network. For operation at a *fixed* $\%C$=90% correct classification accuracy, RndF correctly detected 31 of 36 rogue scenarios at SNR=12.0 dB. This exceeds Generalized Relevance Learning Vector Quantization-Improved (GRLVQI) rogue detection performance which detected 28 of 36 rogue scenarios at SNR=18.0 dB and MDA/ML which detected 25 of 36 rogue scenarios at SNR=30.0 dB. Considering the SNR differences for *fixed* $\%C$=90% operation, RndF emerged as the superior alternative and provided an SNR "gain" G_S=18.0 dB and G_S= 6.0 dB relative to MDA/ML and GRLVQI processes, respectively. Alternately, for operation under *fixed* SNR conditions, RndF ID verification processing detected 11 more rogue scenarios than MDA/ML and 6 more rogue scenarios than GRLVQI. In real-world applications where RF emissions crowd the 2.4 GHz spectral region, the SNR is relatively low and ensemble methods are envisioned as providing a more robust alternative for authenticating ZigBee device IDs and enhancing network security.

For the XRx ZigBee dataset, a comparison is provided of RF-DNA fingerprinting performance using classification variables generated from high-cost PXIe and low-cost USRP receiver collections. By testing all combinations of ZigBee devices in a given population, biased results from favorable device selection are avoided, and results can be generalized for the entire population. RndF Variable Importance (VI) analysis show that the process favors RF-DNA phase variables over amplitude or frequency variables. PXIe-collected RF-DNA phase variables are given higher importance relative to corresponding USRP-collected phase variables. Device authentication results show an average $\%C$=0.76% higher classification when using PXIe collected RF-DNA phase variables over USRP for the 3-class case over all SNRs tested, and $\%C$=2.22% higher for the 6-class case. Device verification results show an average of 10% more Rog:Auth scenarios being identified with the PXIe collected data versus the USRP-collected data.

Thus, as expected, performance is degraded when using the lower cost receiver. However, the performance decrease is not statistically significant at the 95% confidence level. In addition, the magnitude of performance drop is small and may be acceptable given the much lower cost of the USRP receiver. These experiments consider the most-challenging case when all ZigBee devices are from the same manufacturer and model number.

7.2 Suggestions for Future Work

7.2.1 Linear Regression Attack

SCA collections typically suffer from large amounts of noise, either from the environment (heat, cold, etc.) or other processes executing concurrently in the microcontroller. The highest R_a^2 for the time samples examined in this work using the 8-bit linear regression model was only R_a^2=0.36. Thus there is a great deal of noise unaccounted for in the collection. The 8-bit model was used to attack the SubBytes output of the first round of AES as the PIC microcontroller accesses the SubBytes tables one byte at at time. However, other operations are performed 16-bits at a time such as the AddRoundKey operation. When a 16-bit linear regression model was based off of AddRoundKey, SSE was kept low. Further investigation into 16-bit models is needed to determine if they further reduce the variance that not accounted for in the model and result in a higher R_a^2 value. In addition, other terms can be added to the model to allow for a better fit if they are determined to be effective.

The PIC microcontroller used here followed the Hamming Weight (HW) power model well and as such the differences between those of a CEMA attack and LRA were minimal. A Field Programmable Gate Array (FPGA) often routes adjacent memory or logic elements to different areas on the chip to meet timing and space constraints. If an AES core were synthesized on an FPGA such that adjacent bits were spatially separated, the associated side-channel leakage for those bytes may be different, leading to divergence from the power model. This should be investigated to see if LRA performs significantly better than CEMA.

7.2.2 RndF Profiling Attacks

The *40PIC* dataset had training sets of 5000 observations per device. With a 256 class byte-wise profiling attack, there are approximately 19.53 traces per class. It is expected that with a larger training set, the usefulness of non-Gaussian variables to RndF classification should improve. In addition, for Template Attacks with a non-pooled covariance matrix, a larger training set will allow a better estimation of the class covariance matrix and Template Attack classification performance should improve. Future work should use larger training sets to compare classifier performance.

Data collections with the PIC microcontroller produced non-Gaussian variables and RndF classification performed better than Template Attack. XY-scans of the PIC microcontroller revealed that regions near the wires connecting the CPU die to the external pins with high CEMA values have larger numbers of non-Gaussian variables. However, the number of non-Gaussian variables as a percentage of the total variables collected was not nearly as high as in the original *40PIC* dataset. The cause for non-Gaussian variables needs to be investigated further. In addition, side channel collections using a similar setup on more advanced ARM and FPGA devices have not revealed any non-Gaussian variables. For those devices Template Attack outperforms RndF profiling attack. The cause for this should be investigated further in our future work.

A variant of RndF called Rotation Forest [64] grows trees on a randomly selected subspace of the full dimensional set, using Principal Component Analysis (PCA) to rotate the subspace variables. Rotation Forest may be beneficial in cases with correlated variables by performing PCA on a smaller subset of variables for each tree. Initial rotation forest experiments have shown a 10% improvement in correct classification rate using a ARM Cortex M3 processor. Future research could compare the performance with Template Attack, and possibly perform a limited cross-device attack on another ARM processor.

7.2.3 RF-DNA Fingerprinting on ZigBee with RndF

Classifiers in this work were trained to learn the PHY layer attributes of authorized devices, and subsequently used to detect network intrusion by rogue devices. Training in this way is based on an underlying assumption that variables from authorized devices comprise the complete variable space. Test metrics such as posterior class probabilities reflect how much an unknown observation "looks like" one of the authorized devices and can be used to establish separation boundaries between known classes. However, these models are not trained to define boundaries between known and unknown devices operating in the variable space. Future work in RF-DNA fingerprinting with RndF should investigate verification-driven learning which allows for this possibility. A probability decision tree estimates a normal distribution around each region partitioned by a tree. Decision forests have been used to combine density estimates from multiple trees to define distributions around training observations [50]. Regions with no training observations are given lower likelihood and could be relabeled as rogue regions for network intrusion scenarios. This type of verification-driven learning is expected to enhance rogue identification and give better warning of network intrusion attacks.

The observation waveform Region of Interest (ROI) used to generate parametric summary statistics of variance, skewness and kurtosis is not typically Gaussian, but often multi-modal. Hence non-parametric summary statistics may summarize these ROIs better. Non-parametric feature generation methods should be investigated to assess classification improvement relative to the parametric summary statistics as used here. Other ensemble learning algorithms may be examined such as those specified in Section 2.3.3. Although only 15% of the pair-wise correlation between variables in datasets here were greater than 0.5, classification performance may be improved with ensembles of multivariate decision trees such as those used in [64, 73]. Finally, for the XRx dataset, only RndF was used for device classification and rogue rejection experiments. An analysis of variables and

subsequent comparison of RndF to other classifiers would provide insight into which classifiers are best to use with this type of collection data.

Appendix A: Derivation of Linear Least Squares Estimator

This appendix provides the derivation of the Least Squares estimator as described in [65].

Suppose for an observation y dependent on a series of independent variables $\{x_1, x_2, \cdots, x_u\}$, we have the linear model

$$y = r + \sum_{i=1}^{u} \beta_i x_i \,, \tag{A.1}$$

with r as the residual term. For N observations, these can be represented in the forms

$$\mathbf{Y} = [y_1, y_2, \cdots, y_N]' \tag{A.2}$$

$$\mathbf{R} = [r_1, r_2, \cdots, r_N]'$$

$$\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_u]'$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1u} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nu} \end{pmatrix},$$

where $'$ represents the transpose. The linear model can be represented in matrix form by

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{R} \,. \tag{A.3}$$

The linear Least Squares estimator estimates the dependent term with model parameters estimates given by

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}, \tag{A.4}$$

leading to the error term

$$\hat{\mathbf{R}} = \mathbf{Y} - \hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}} \tag{A.5}$$

The goal of Least Squares is to minimize the Sum of Squares Error (SSE) $\hat{\mathbf{R}}'\hat{\mathbf{R}}$, that is

$$\begin{aligned}\min_{\text{w.r.t. } \hat{\beta}} \mathbf{R}'\mathbf{R} &= (\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}})'(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ &= (\mathbf{Y}' - \mathbf{X}'\hat{\boldsymbol{\beta}}')(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ &= \mathbf{Y}'\mathbf{Y} - 2\hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{Y} + \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}}\end{aligned} \tag{A.6}$$

To find the minimum, the partial derivative with respect to $\hat{\boldsymbol{\beta}}$ is taken in

$$\begin{aligned}\frac{\partial \hat{\mathbf{R}}'\hat{\mathbf{R}}}{\partial \boldsymbol{\beta}} &= \frac{\partial \mathbf{Y}'\mathbf{Y} - 2\hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{Y} + \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}}}{\partial \boldsymbol{\beta}} = 0 \\ -2\mathbf{X}'\mathbf{Y} + 2\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} &= 0 \\ \mathbf{X}'\mathbf{Y} &= \mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}\end{aligned} \tag{A.7}$$

This estimate for $\hat{\boldsymbol{\beta}}$ is the least squares estimator.

Appendix B: ZigBee Stat RF-DNA Fingerprint Features

This appendix shows the histograms of all 25 top ranked *Stat* RF-DNA variables at $SNR = 30.0$ dB identified by RndF VI metric. Each variable shows a distinct non-Gaussian shape. These results were presented in [93].

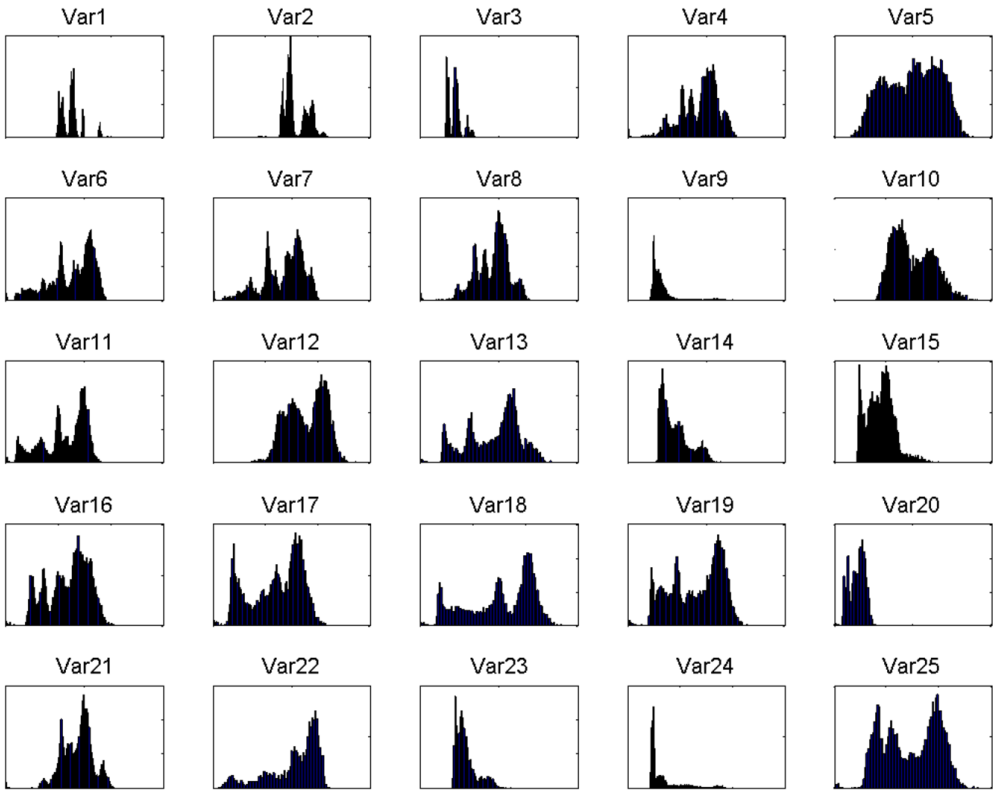


Figure B.1: Histogram of the top-ranked 25 *Stat* variables at $SNR = 30.0$ dB identified by RndF VI metric. Each variable shows distinct non-Gaussian shape.

Bibliography

[1] Agilent Technologies Inc. *Agilent E3238 Signal Intercept and Collection Solutions: Family Overview*. Technical Report Pub 5989-1274EN, Agilent, 2004.

[2] Agrawal, Dakshi, Bruce Archambeault, Josyula Rao, and Pankaj Rohatgi. "The EM SideChannel(s)". *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, 29–45. Springer Berlin / Heidelberg, 2003.

[3] Agrawal, Dakshi, Josyula R Rao, Pankaj Rohatgi, and Kai Schramm. "Templates as Master Keys". *Cryptographic Hardware and Embedded Systems–CHES 2005*, 15–29. Springer, 2005.

[4] Akkar, Mehdi-Laurent, Régis Bevan, and Louis Goubin. "Two Power Analysis Attacks Against One-Mask Methods". *Fast Software Encryption*, 332–347. Springer, 2004.

[5] Amaratunga, Dhammika, Javier Cabrera, and Yung-Seop Lee. "Enriched Random Forests". *Bioinformatics*, 24(18):2010–2014, 2008.

[6] Amit, Yali, Gilles Blanchard, and Kenneth Wilder. *Multiple Randomized Classifiers: MRCL*. Technical report, 2000.

[7] Archambeau, C., E. Peeters, F.-X. Standaert, and J.-J. Quisquater. "Template Attacks in Principal Subspaces". *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, 1–14. Springer Berlin / Heidelberg, 2006.

[8] Atmel Corporation, San Jose, CA. *AVR Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*, Rev5131E edition, February 2009.

[9] Baker, Nick. "ZigBee and Bluetooth Strengths and Weaknesses For Industrial Applications". *Computing and Control Engineering Journal*, 16(2):20–25, 2005.

[10] Baronti, Paolo, Prashant Pillai, Vince W. C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards". *Computer Communications*, 30(7):1655–1695, 5/26 2007.

[11] Batina, Lejla, Jip Hogenboom, and Jasper GJ van Woudenberg. "Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis". *Topics in Cryptology–CT-RSA 2012*, 383–397. Springer, 2012.

[12] Biau, Gérard. "Analysis of a Random Forests Model". *The Journal of Machine Learning Research*, 13:1063–1095, 2012.

[13] Biau, Gérard, Luc Devroye, and Gábor Lugosi. "Consistency of Random Forests and Other Averaging Classifiers". *J. Mach. Learn. Res.*, 9:2015–2033, June 2008.

[14] Bishop, Christopher M. *Pattern Recognition and Machine Learning*, volume 1. springer New York, 2006.

[15] Breiman, Leo. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Chapman & Hall, 1984.

[16] Breiman, Leo. "Bagging Predictors". *Machine learning*, 24(2):123–140, 1996.

[17] Breiman, Leo. "Random Forests". *Machine Learning*, 45:5–32, 2001.

[18] Breiman, Leo. "Statistical Modeling: The Two Cultures". *Statiscal Science*, 199–231, 2001.

[19] Brier, Eric, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, 135–152. Springer Berlin / Heidelberg, 2004.

[20] Buhlmann, Peter and Torsten Hothorn. "Boosting Algorithms: Regularization, Prediction and Model Fitting". *Statistical Science*, 22:477–505, 2007.

[21] Bushnell, Michael L. and Vishwani Agrawal. *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Springer, New York, NY, November 2001.

[22] Caruana, Rich, Nikos Karampatziakis, and Ainur Yessenalina. "An Empirical Evaluation of Supervised Learning in High Dimensions". *Proceedings of the 25th international conference on Machine learning*, ICML '08, 96–103. ACM, New York, NY, USA, 2008.

[23] Caruana, Rich, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. "Ensemble Selection From Libraries of Models". *Proceedings of the twenty-first international conference on Machine learning*, 18–25. ACM, 2004.

[24] Chari, Suresh, Josyula Rao, and Pankaj Rohatgi. "Template Attacks". *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, 51–62. Springer Berlin / Heidelberg, 2003.

[25] Cobb, W. E., E. D. Laspe, R. O. Baldwin, Michael A. Temple, and Y. C. Kim. "Intrinsic Physical-Layer Authentication of Integrated Circuits". *Information Forensics and Security, IEEE Transactions on*, 7(1):14–24, 2012.

[26] Cobb, William. *Exploitation of the Unintentional Information Leakage of Integrated Circuits*. Ph.D. thesis, Air Force Institute of Technology, June 2011.

[27] Croarkin, Carroll and Paul Tobias. *NIST/SEMATECH E-Handbook of Statistical Methods*. Technical report, NIST, 2006.

[28] Daemen, Joan and Vincent Rijmen. *The Design of Rijndael: AES-the Advanced Encryption Standard*. Springer, 2002.

[29] Danev, Boris and Srdjan Capkun. "Transient-Based Identification of Wireless Sensor Nodes". *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, 25–36. IEEE Computer Society, 2009.

[30] Danev, Boris, Heinrich Luecken, Srdjan Capkun, and Karim El Defrawy. "Attacks on Physical-Layer Identification". *Proceedings of the third ACM conference on Wireless network security*, 89–98. ACM, 2010.

[31] Díaz-Uriarte, Ramón and Sara Alvarez De Andres. "Gene Selection and Classification of Microarray Data Using Random Forest". *BMC bioinformatics*, 7(1):3, 2006.

[32] Díaz-Uriarte, Ramón and Sara Alvarez De Andres. "Gene Selection and Classification of Microarray Data Using Random Forest". *BMC bioinformatics*, 7(1):3, 2006.

[33] Dini, Gianluca and Marco Tiloca. "Considerations on Security in ZigBee Networks". *IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 58–65. 2010.

[34] Doget, Julien, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. "Univariate Side Channel Attacks and Leakage Modeling". *Journal of Cryptographic Engineering*, 1(2):123–144, 2011.

[35] Dubendorfer, C. K., B. W. Ramsey, and M. A. Temple. "An RF-DNA Verification Process for ZigBee Networks". *Military Communications Conference*, 1–6. 2012.

[36] Dubendorfer, C. K., Benjamin W. Ramsey, and Temple Michael A. *ZigBee Device Verification for Securing Industrial Control and Building Automation Systems*. Critical Infrastructure Protection. Springer, 7th edition, 2013.

[37] Duda, R. O and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1st edition, 1989.

[38] Egan, David. "The Emergence of ZigBee in Building Automation and Industrial Controls". *Computing and Control Engineering*, 16(2):14–19, 2005.

[39] Ferri, César, Peter Flach, and José Hernández-Orallo. "Learning Decision Trees Using the Area Under the ROC Curve". *International Conference on Machine Learning*, volume 2, 139–146. 2002.

[40] Fisher, Ronald A. "The Use of Multiple Measurements in Taxonomic Problems". *Annals of Eugenics*, 7(2):179–188, 1936.

[41] Gebotys, Catherine H, Simon Ho, and Chin Chi Tiu. "EM Analysis of Rijndael and ECC on a Wireless Java-based PDA". *Cryptographic Hardware and Embedded Systems–CHES 2005*, 250–264. Springer, 2005.

[42] Gierlichs, Benedikt, Kerstin Lemke-Rust, and Christof Paar. "Templates vs. Stochastic Methods". Louis Goubin and Mitsuru Matsui (editors), *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, 15–29. Springer Berlin / Heidelberg, 2006.

[43] Guilley, S., L. Sauvage, P. Hoogvorst, R. Pacalet, G.M. Bertoni, and S. Chaudhuri. "Security Evaluation of WDDL and SecLib Countermeasures against Power Attacks". *IEEE Transactions on Computers*, 57(11):1482–1497, 2008.

[44] Hammer, Barbara and Thomas Villmann. "Generalized Relevance Learning Vector Quantization". *Neural Networks*, 15(8):1059–1068, 2002.

[45] Handschuh, Helena and Bart Preneel. "Blind differential cryptanalysis for enhanced power attacks". *Selected Areas in Cryptography*, 163–173. Springer, 2007.

[46] Harmer, Paul, Michael Temple, Mark Buckner, and Ethan Farquhar. "4G Security Using Physical Layer RF-DNA with DE-Optimized LFS Classification". *Journal of Communications*, 6(9):671–681, 2011.

[47] Harmer, Paul K., Donald R. Reising, and Temple Michael A. "Classifier Selection for Physical Layer Security Augmentation in Cognitive Radio Networks". *IEEE Int Conf on Communications ICC2013*. 2013.

[48] Harmer, Paul K. and Michael A. Temple. "An Improved LFS Engine for Physical Layer Security Augmentation in Cognitive Networks". *Int. Conf. on Computing, Networking and Communications (ICNC)*, 719–723. IEEE, 2013.

[49] Heuser, A., M. Kasper, W. Schindler, and M. Stöttinger. "A New Difference Method for Side-Channel Analysis with High-Dimensional Leakage Models". *Topics in Cryptology–CT-RSA 2012*, 365–382, 2012.

[50] Heuser, Annelie, Michael Kasper, Werner Schindler, and Marc Stottinger. "How a Symmetry Metric Assists Side-Channel Evaluation-A Novel Model Verification Method for Power Analysis". *Digital System Design (DSD), 2011 14th Euromicro Conference on*, 674–681. IEEE, 2011.

[51] Heuser, Annelie and Michael Zohner. "Intelligent Machine Homicide". Werner Schindler and SorinA. Huss (editors), *Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, 249–264. Springer Berlin Heidelberg, 2012.

[52] Heyszl, Johann, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. "Localized Electromagnetic Analysis of Cryptographic Implementations". *Topics in Cryptology–CT-RSA 2012*, 231–244. Springer, 2012.

[53] Ho, Tin Kam. "The Random Subspace Method for Constructing Decision Forests". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[54] Homma, Naofumi, Sei Nagashima, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. "High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching". *Cryptographic Hardware and Embedded Systems-CHES 2006*, 187–200. Springer, 2006.

[55] Hospodar, Gabriel, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. "Machine Learning in Side-Channel Analysis: A First Study". *Journal of Cryptographic Engineering*, 1:293–302, 2011.

[56] IEEE. *Wireless MAC and PHY Specifications for Low-Rate WPANS*. Technical report, IEEE Standard 802.15.4-2006, 2006.

[57] Jaffe, Josh. "A First-Order DPA Attack Against AES in Counter Mode With Unknown Initial Counter". *Cryptographic Hardware and Embedded Systems-CHES 2007*, 1–13. Springer, 2007.

[58] Jung, J. Y. and J. W. Lee. "ZigBee Device Design and Implementation for Context-Aware U-Healthcare System". *2nd Int. Conf. on Systems and Networks Communications*, 22–22. IEEE, 2007.

[59] Kay, Steven M. *Fundamentals of Statistical Signal Processing, Volume2: Detection Theory*. Prentice Hall PTR, New Jersey, 1st edition, 1998.

[60] Kinney, Patrick. "Zigbee Technology: Wireless Control that Simply Works". *Communications design conference*, volume 2. 2003.

[61] Kocher, Paul, Joshua Jaffe, and Benjamin Jun. *Differential Power Analysis*, volume 1666 of *Advances in Cryptology CRYPTO 99*, 789–789. Springer Berlin / Heidelberg, 1999. ISBN 978-3-540-66347-8.

[62] Kocher, Paul C. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". *Advances in CryptologyCRYPTO96*, 104–113. Springer, 1996.

[63] Kohonen, Teuvo. *Self-Organizing Maps*, volume Vol 30. Springer, New York, 3rd edition, 2001.

[64] Kuncheva, Ludmila I. and Juan J. Rodríguez. "An Experimental Study on Rotation Forest Ensembles". *Proceedings of the 7th international conference on Multiple classifier systems*, MCS'07, 459–468. Springer-Verlag, Berlin, Heidelberg, 2007.

[65] Kutner, Michael, Christopher Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw-Hill/Irwin, 1974. ISBN 9780073108742.

[66] Lerman, Liran, Gianluca Bontempi, and Olivier Markowitch. "Side Channel Attack: An Approach Based on Machine Learning". *COSADE - Second International Workshop on Constructive Side-Channel Analysis and Secure Design*, Lecture Notes in Computer Science. Germany, 2011.

[67] Li, Tao, Shenghuo Zhu, and Mitsunori Ogihara. "Using Discriminant Analysis for Multi-Class Classification: An Experimental Investigation". *Knowledge and information systems*, 10(4):453–472, 2006.

[68] Liang, Nia-Chiang, Ping-Chieh Chen, Tony Sun, Guang Yang, Ling-Jyh Chen, and Mario Gerla. "Impact of Node Heterogeneity in ZigBee Mesh Network Routing". *IEEE Int. Conf. on Systems, Man and Cybernetics, SMC'06*, volume 1, 187–191. IEEE, 2006.

[69] Lilliefors, Hubert W. "On the Kolmogorov-Smirnov Test For Normality With Mean and Variance Unknown". *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[70] Lin, Yi and Yongho Jeon. "Random Forests and Adaptive Nearest Neighbors". *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[71] Lu, Y., K. H. Boey, M. O'Neill, and J. V. McCanny. "Practical Comparison of Differential Power Analysis Techniques on an ASIC Implementation of the AES Algorithm". *Signals and Systems Conference (ISSC 2009), IET Irish*, 1–6. 2009. ID: 1.

[72] Mangard, Stefan, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 0387308571.

[73] Menze, Bjoern, B. Kelm, Daniel Splitthoff, Ullrich Koethe, and Fred Hamprecht. "On Oblique Random Forests". *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, 453–469. Springer Berlin / Heidelberg, 2011.

[74] Menze, Bjoern H., B. Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A. Hamprecht. "A Comparison of Random Forest and its Gini Importance with Standard Chemometric Methods for the Feature Selection and Classification of Spectral Data". *BMC Bioinformatics*, 10(1):213, 2009.

[75] Messerges, Thomas S, Ezzy A Dabbish, and Robert H Sloan. "Investigations of Power Analysis Attacks on Smartcards". *USENIX workshop on Smartcard Technology*, volume 1999. 1999.

[76] de Meulenaer, Giacomo and François-Xavier Standaert. "Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks". *Mobile Lightweight Wireless Systems*, 229–242. Springer, 2010.

[77] Meyer, David, Friedrich Leisch, and Kurt Hornik. "The Support Vector Machine Under Test". *Neurocomputing*, 55:169 – 186, 2003.

[78] Microchip Inc., Chandler, AZ. *Explorer 16 Development Board Users Guide*, October 2005.

[79] Microchip Inc., Chandler, AZ. *PIC24FJ64GA004 Family 28/44-Pin General Purpose, 16-Bit Flash Microcontroller Datasheet*, November 2011.

[80] Milton, Susan and Jesse Arnold. *Introduction to Probability and Statistics*. McGraw-Hill, New York, 4th edition, 2003.

[81] Mitchell, R. and I. Chen. "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems". *Reliability, IEEE Transactions on*, 62(1):199–210, 2013.

[82] Miyamoto, A., N. Homma, T. Aoki, and A. Satoh. "Evaluation of Simple/Comparative Power Analysis against an RSA ASIC implementation". *ISCAS 2009, IEEE International Symposium on*, 2918 –2921. May 2009.

[83] Montminy, David. *Enhancing Electromagnetic Side-Channel Analysis in an Operational Environment*. Ph.D. thesis, U.S. Air Force Institute of Technology, WPAFB, OH, September 2013.

[84] Montminy, David P, Rusty O Baldwin, Michael A Temple, and Eric D Laspe. "Improving Cross-Device Attacks Using Zero-Mean Unit-Variance Normalization". *Journal of Cryptographic Engineering*, 1:1–12, 2012.

[85] Na, Xiaodong, Shuying Zang, and Jianhua Wang. "Evalution of Random Forest Ensemble Classification for Land Cover Mapping Using TM and Ancillary Geographical Data". *Fuzzy Systems and Knowledge Discovery*, 89–93. 2009.

[86] National Institute of Standards and Technology. *FIPS-197*. Technical report, NIST, November 2001.

[87] Nguyen, Nam Tuan, Guanbo Zheng, Zhu Han, and Rong Zheng. "Device Fingerprinting to Enhance Wireless Security Using Nonparametric Bayesian Method". *INFOCOM, 2011 Proceedings IEEE*, 1404–1412. IEEE, 2011.

[88] Niculescu-Mizil, Alexandru and Rich Caruana. "Predicting Good Probabilities with Supervised Learning". *Proceedings of the 22nd international conference on Machine learning*, 625–632. ACM, 2005.

[89] Nitze, I., U. Schulthess, and H. Asche. "Comparison of Machine Learning Algorithms Random Forest, Artificial Neural Network and Support Vector Machine to Maximum Likelihood For Supervised Crop Type Classification". *Proceedings of GEOBIA*, volume 4th. Brazil, May 2012.

[90] Oswald, Elisabeth and Stefan Mangard. "Template Attacks on MaskingResistance Is Futile". Masayuki Abe (editor), *Topics in Cryptology CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, 243–256. Springer Berlin Heidelberg, 2006.

[91] Patel, Hiren and Rusty Baldwin. "Random Forest Profiling Attack on Advanced Encryption Standard". *International Journal of Applied Cryptography*, 3(2), 2014.

[92] Patel, Hiren, Christine Shubert-Kabban, and Rusty Baldwin. "Statistical Analysis of Linear Regression Based Side Channel Attack". *International Journal of Information and Communication Technology*, To Appear 2013.

[93] Patel, Hiren, Michael Temple, and Rusty Baldwin. "Improving ZigBee Device Network Authentication Using Ensemble Decision Tree Classifiers with RF-DNA Fingerprinting". *Submitted to IEEE Transactions on Reliability*, 2014.

[94] Patel, Hiren, Michael Temple, Rusty Baldwin, and Benjamin Ramsey. "Application of Ensemble Decision Tree Classifiers to ZigBee Device Network Authentication Using RF-DNA Fingerprinting". *9th International Conference on Cyber Warfare and Security 2014*. West Lafayette, IN, 2014.

[95] Patel, Hiren, Michael Temple, and Benjamin Ramsey. "Comparison of High-end and Low-end Receivers for RF-DNA Fingerprinting". *Military Communication Conference MILCOM 2014*. Baltimore, MD, October 2014.

[96] Polak, Adam C, Sepideh Dolatshahi, and Dennis L Goeckel. "Identifying Wireless Users Via Transmitter Imperfections". *Selected Areas in Communications, IEEE Journal on*, 29(7):1469–1479, 2011.

[97] Prouff, Emmanuel and Matthieu Rivain. "Theoretical and Practical Aspects of Mutual Information-Based Side Channel Analysis". *International Journal of Applied Cryptography*, 2(2):121–138, 2010.

[98] Quisquater, Jean-Jacques and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards". *Smart Card Programming and Security*, volume 2140 of *Lecture Notes in Computer Science*, 200–210. Springer Berlin / Heidelberg, 2001.

[99] Radmand, Pedram, Marc Domingo, Jaipal Singh, Joan Arnedo, Alex Talevski, Stig Petersen, and Simon Carlsen. "ZigBee/ZigBee PRO Security Assessment Based on Compromised Cryptographic Keys". *Int. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 465–470. IEEE, 2010.

[100] Raileanu, Laura and Kilian Stoffel. "Theoretical Comparison between the Gini Index and Information Gain Criteria". *Annals of Mathematics and Artificial Intelligence*, 41:77–93, 2004.

[101] Ramsey, Benjamin W., Barry E. Mullins, and Edward D. White. "Improved Tools for Indoor ZigBee Warwalking". *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, 921–924. IEEE, 2012.

[102] Ramsey, Benjamin W., Michael A. Temple, and Barry E. Mullins. "PHY Foundation for Multi-Factor ZigBee Node Authentication". *IEEE, Global Communications Conference (GLOBECOM),*, 795–800. IEEE, 2012.

[103] Rechberger, Christian and Elisabeth Oswald. "Practical Template Attacks". *Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, 440–456. Springer Berlin / Heidelberg, 2005.

[104] Rehman, Saeed Ur, Kevin Sowerby, and Colin Coghill. "Analysis of Receiver Front End on the Performance of RF Fingerprinting". *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, 2494–2499. IEEE, 2012.

[105] Rehman, Saeed Ur, Kevin W. Sowerby, and Colin Coghill. "Analysis of Impersonation Attacks on Systems Using RF Fingerprinting and Low-End Receivers". *Journal of Computer and System Sciences*, 80(3):591 – 601, 2014. URL http://www.sciencedirect.com/science/article/pii/S0022000013001220.

[106] Reising, Donald R. *Exploitation of RF-DNA for Device Classification and Verification Using GRLVQI Processing*. Ph.D. thesis, U.S. Air Force Institute of Technology, Dayton, OH, August 2012.

[107] Reising, Donald R., Michael A. Temple, and Mark E. Oxley. "Gabor-Based RF-DNA Fingerprinting For Classifying 802.16 e WiMAX Mobile Subscribers". *Int. Conf. Computing, Networking and Communications (ICNC)*, 7–13. IEEE, 2012.

[108] Ren, Yanting and Liji Wu. "Power Analysis Attacks on Wireless Sensor Nodes Using CPU Smart Card". *Wireless and Optical Communication Conference (WOCC), 2013 22nd*, 665–670. IEEE, 2013.

[109] Rich Caruana, Nikos Karampatziakis and Ainur Yessenalina. "An Empirical Evaluation of Supervised Learning in High Dimensions". *International Conference on Machine Learning*, volume 25. Helsinki, Finland, 2008.

[110] Riscure. *EM Probe Station Inspector Data Sheet*, November 2011. URL https://www.riscure.com/benzine/documents/EMProbeStation.pdf.

[111] Riscure. "Inspector - The Side Channel Test Platform", July 2011. URL http://www.riscure.com/inspector/product-description.html.

[112] Rivest, Ronald L, Adi Shamir, and Len Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Communications of the ACM*, 21(2):120–126, 1978.

[113] Rohde and Schwarz. *Probe Set HZ-15 Data Sheet*, 01.01 edition, May 2006.

[114] Sato, Atsushi and Keiji Yamada. "Generalized Learning Vector Quantization". *Advances in neural information processing systems*, 423–429, 1996.

[115] Schindler, Werner, Kerstin Lemke, and Christof Paar. "A Stochastic Model for Differential Side Channel Cryptanalysis". *Cryptographic Hardware and Embedded Systems CHES 2005*, 30–46. Springer Berlin / Heidelberg, Scotland, 2005.

[116] Shannon, Claude E. "Communication Theory of Secrecy Systems*". *Bell system technical journal*, 28(4):656–715, 1949.

[117] Sklar, Bernard. *Digital Communications: Fundamentals and Applications*. Pearson, 2001.

[118] Standaert, François-Xavier, François Koeune, and Werner Schindler. "How to Compare Profiled Side-Channel Attacks". *Applied Cryptography and Network Security*, 485–498. Springer, 2009.

[119] Standaert, François-Xavier, Tal G Malkin, and Moti Yung. "A Unified Framework For the Analysis of Side-Channel Key Recovery Attacks". *Advances in Cryptology-EUROCRYPT 2009*, 443–461. Springer, 2009.

[120] Standaert, Franois-Xavier and Cedric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". *Cryptographic Hardware and Embedded Systems CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, 411–425. Springer Berlin / Heidelberg, 2008.

[121] Stubbs, Tyler D. *A Comparison of RF-DNA Fingerprinting Using High/Low Value Receivers With ZigBee Devices*. Master's thesis, US Air Force Institute of Technology, WPAFB, OH, March 2014.

[122] Suh, G. Edward and Srinivas Devadas. "Physical Unclonable Functions For Device Authentication and Secret Key Generation". *Proceedings of the 44th annual Design Automation Conference*, 9–14. ACM, 2007.

[123] Sun, Quan and Bernhard Pfahringer. "Bagging Ensemble Selection". *AI 2011: Advances in Artificial Intelligence*, 251–260. Springer, 2011.

[124] Texas Instruments Incorporated, Dallas, Texas. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, RevC edition, 2014.

[125] Theodoridis, Sergios and Konstantinos Koutroumbus. *Pattern Recognition*. Elsevier, Burlington, MA, fourth edition, 2009.

[126] Tihon, I. and V. Croitoru. "ZigBee Sensor Networks Telesurveillance". *10th Int. Symp. on Signals, Circuits and Systems (ISSCS)*, 1–4. 2011. ID: 1.

[127] Vidgren, Niko, Keijo Haataja, Jos Luis Patino-Andres, Juan Jose Ramirez-Sanchis, and Pekka Toivanen. "Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned". *46th Hawaii Int. Conf. on System Sciences (HICSS)*, 5132–5138. IEEE, 2013.

[128] Viola, P. and M.J. Jones. "Robust Real-Time Face Detection". *International journal of computer vision*, 57(2):137–154, 2004.

[129] Williams, MD, Michael A. Temple, and Donald R. Reising. "Augmenting Bit-Level Network Security Using Physical Layer RF-DNA Fingerprinting". *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 1–6. IEEE, 2010.

[130] Wright, Joshua. "KillerBee: Practical ZigBee Exploitation Framework". *11th ToorCon conference, San Diego*. 2009.

[131] cheng Yin, Xu, Chang ping Liu, and Zhi Han. "Feature Combination Using Boosting". *Pattern Recognition Letters*, 26:2195–2205, 2005.

[132] Zhang, Yichi, Lingfeng Wang, Weiqing Sun, RC Green, and Mansoor Alam. "Distributed Intrusion Detection System in a Multi-Layer Network Architecture of Smart Grids". *Smart Grid, IEEE Transactions on*, 2(4):796–808, 2011.

[133] Zhu, Ji, Hui Zou, Saharon Rosset, and Trevor Hastie. "Multi-Class Adaboost". *Statistics and Its Inference*, 2:349–360, 2009.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
18–09–2014	Doctoral Dissertation	Sep 2011-Sep 2014

4. TITLE AND SUBTITLE

Advances in SCA and RF-DNA Fingerprinting Through Enhanced Linear Regression Attacks and Application of Random Forest Classifiers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Patel, Hiren J., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765 DSN: 785-3636

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-DS-14-S-03

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory, AFMC
Attn: Dr. Vasu Chakravarthy
2241 Avionics Circle, Bldg 620
Wright-Patterson AFB OH 45433-7734
Vasu.Chakravarthy@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Radio Frequency (RF) emissions from electronic devices expose security vulnerabilities that can be used by an attacker to extract otherwise unobtainable information. Two realms of study were investigated here, including the exploitation of 1) unintentional RF emissions in the field of Side Channel Analysis (SCA), and 2) intentional RF emissions from physical devices in the field of RF-Distinct Native Attribute (RF-DNA) fingerprinting. Statistical analysis on the linear model fit to measured SCA data in Linear Regression Attacks (LRA) improved performance, achieving 98% success rate for AES key-byte identification from unintentional emissions. However, the presence of non-Gaussian noise required the use of a non-parametric classifier to further improve key guessing attacks. RndF based profiling attacks were successful in very high dimensional data sets, correctly guessing all 16 bytes of the AES key with a 50,000 variable dataset. With variable reduction, Random Forest still outperformed Template Attack for this data set, requiring fewer traces and achieving higher success rates with lower misclassification rate. Finally, the use of a RndF classifier is examined for intentional RF emissions from ZigBee devices to enhance security using RF-DNA fingerprinting. RndF outperformed parametric MDA/ML and non-parametric GRLVQI classifiers, providing up to G_S=18.0 dB improvement (reduction in required SNR). Network penetration, measured using rogue ZigBee devices, show that the RndF method improved rogue rejection in noisier environments - gains of up to G_S=18.0 dB are realized over previous methods.

15. SUBJECT TERMS

Side Channel Analysis, SCA, Random Forest, Template Attack, RF-DNA, RF-Fingerprinting, ZigBee

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Michael A. Temple (ENG)
U	U	U	UU	221	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4279; email:michael.temple@afit.edu