Side Channel Attacks on IoT Applications

By

Yan Yan

Department of Computer Science

University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

January 2019

Word count: 39931

Abstract

The Internet of Things (IoT) has become a reality: small connected devices feature in everyday objects including childrens' toys, TVs, fridges, heating control units, etc. Supply chains feature sensors throughout, and significant investments go into researching next-generation healthcare, where sensors monitor wellbeing. A future in which sensors and other (small) devices interact to create sophisticated applications seems just around the corner. All of these applications have a fundamental need for security and privacy and thus cryptography is deployed as part of an attempt to secure them.

This thesis explores a particular type of security threat against IoT devices, namely side channel attacks (SCA), that has been proven only more powerful over the years. In brief, a side channel attack targets the implementation of security measures and recovers secret data by exploiting execution related information. For instance, secret keys can be recovered by statistically analysing the timing or power consumption of the execution of cryptographic algorithms, or sometimes results of faulty executions; data protected in encrypted packets can be revealed by the length of packets and timing of responses.

Three vulnerabilities in IoT applications have been identified in this work including a flawed Random Number Generator (RNG) design, an effective application of Differential Power Analysis (DPA) and the practicability of Traffic Analysis (TA). These vulnerabilities commonly exist in many IoT scenarios and thus should be taken into account when designing new applications.

Acknowledgements

The research could never be done without help of many others. Firstly I would like to thank my primary supervisor Elisabeth Oswald for her continued guidance and support through the research, especially her offer to me to take part in the LADA project. I would thank my second supervisor Theo Tryfonas for sharing his valuable visions from the aspect of cyber security which greatly contributed to the theme of my research. The support of my family to my decision of pursuing the PhD was also among the most important factor in completing this research.

Great contributions came from my colleagues and co-authors Srivinas Vivek Vanketah and Arnab Roy in my research on the ARX ciphers and generic distingsuiers. I sincerely appreciate the comments from Dan Page and Dritan Kaleshi during my internal PhD progress reviews.

I would thank George Oikonomou for introducing me to Contiki OS as well various IoT devices, and Carolyn Whitnall for her patience in answering my frequent questions related to statistics as well as helping me in writing this thesis. Si Gao helped me set up the experiments for power analysis, Jake Longo for taught me the basis of signal processing and David McCann offered great help with ELMO. Joey Green, Marco Martinoli and James Howe also provided precious advises during my writing of this thesis.

And finally, my many thanks to the cryptography research group of University of Brisol, especially to the people of the side channel research group.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE:.....

Contents

List of Figures

List of Tables

Nomenclature

ADC Analogue to Digital Converter

ARX Addition Rotation and XOR

CDF Cumulative Distribution Function

CPA Correlation Power Analysis

DPA Differential Power Analysis

DTLS Datagram TLS

ECC Elliptic Curve Cryptography

ECDF Empirical Cumulative Distribution Function

GE Gate Equivalent

HD Hamming Distance

HMAC Hash Message Authentication Code

HW Hamming Weight

IoT Internet of Things

ISO International Organization for Standardization

LCG Linear Congruential Generator

LFSR Linear Feedback Shift Register

LSB Least Significant Bit

LWC Light-Weight Cryptography

LWE Learning With Error

MCU Micro-Controller Unit

MSB Most Significant Bit

MTU Maximum Transmission Unit

NIST National Institute of Standards and Technology

OSI Open Systems Interconnection

OS Operating System

PDF Probability Density Function

PPT Probabilistic Polynomial Time

PRI Ping Response Interval

PRNG Pseudo Random Number Generator

RFID Radio-Frequency IDentification

RF Radio Frequency

RNG Random Number Generator

SCA Side Channel Attacks

SNR Signal to Noise Ratio

SoC System on Chip

SPA Simple Power Analysis

SSL Secure Sockets Layer

TA Traffic Analysis

TI Texas Instruments

TLS Transport Layer Security

TRNG True-RNG

VANET Vehicular Ad-hoc Network

WSN Wireless Sensor Network

XOR Exclusive-or

Chapter 1

Introduction

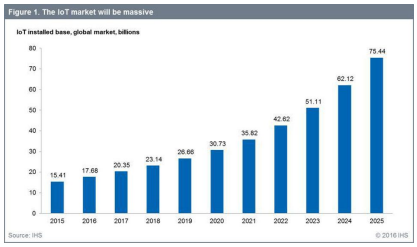
The recent revolution of silicon technologies has dramatically changed the landscape of microcontrollers. The concept of Internet of Things (IoT) was derived in the latest procedure of integrating wireless communication, electronic systems, and Internet technologies. In brief, IoT is a collection of technologies that empowers daily objects, such as vehicles, furniture and wearables(Figure 1.1a), with embedded microcontrollers. These objects are then connected to the Internet so that they can be managed and monitored remotely through interfaces such as cell phones or web browsers, providing a great flexibility for future applications. Furthermore, the computational power that comes with the embedded microcontrollers also enables the IoT objects to automatically interact with variate environments without requiring human intervention. It is estimated that:

> “ *... the IoT market will grow from an installed base of 15.4 billion devices in 2015 to 30.7 billion devices in 2020 and 75.4 billion in 2025.* ” ([1])

A concept image from the Libeilium project [3] (Figure 1.2) is a good demonstration of the capability of IoT applications.

Security is becoming an ever greater challenge for IoT applications, as

(a) Concept of IoT (Source: [2])



(b) Estimated IoT Market to 2025 (Source: [1])

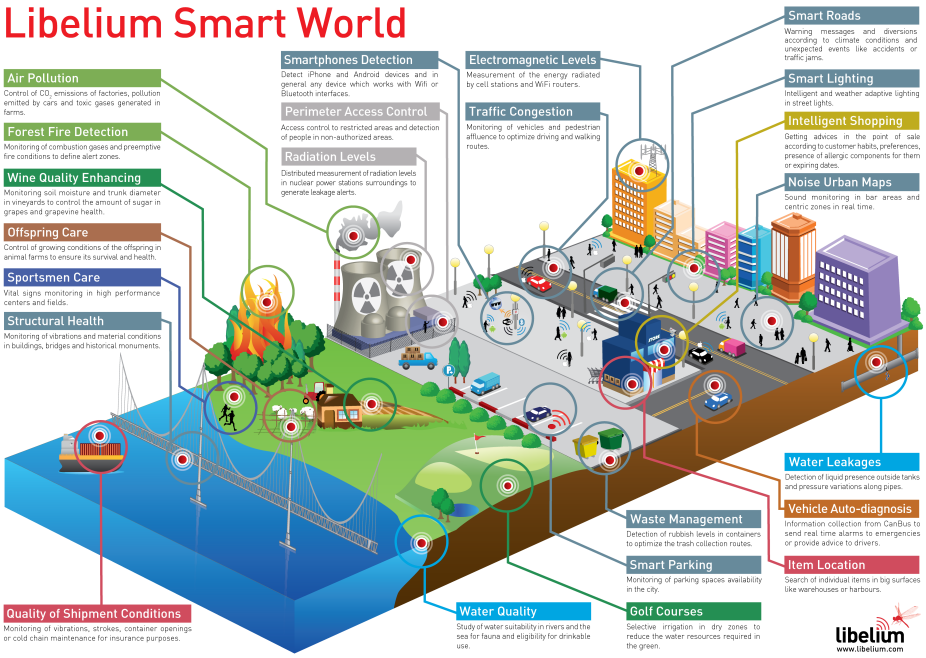


Figure 1.2: Concept Image of Libelium Smart World Project (Source: [3])

more and more information is being digitalised by these devices. It is estimated that:

> “ *The total volume of data generated by IoT will reach 600 ZB per year by 2020, 275 times higher than projected traffic going from data centers to end users/devices (2.2 ZB); 39 times higher than total projected data center traffic (15.3 ZB).* ” ([4])

Not only is the amount of data explosively increasing, the damage that could be caused by compromised IoT devices is also escalating. For example, private data in health care applications is highly confidential, and unauthorised data could lead to lethal consequences in an autonomous car application.

1.1 Research Motivation

Different IoT applications have differ desires. For example, Wireless Sensor Network (WSN) applications have a strong demand in the energy efficiency of the devices as well as their size. Vehicular Ad-hoc Network (VANET), on the other hand, has more freedom in terms of energy and size but has a higher demand on the response time and thus computational power. Designers of IoT application always found themself struggling in the dilemma of demands. For example, reducing the size often results in a reduced volume of battery; higher speed normally comes with more energy consumption as well as more complicated circuits and thus more space. When it comes to applications with large deployment, such as WSN, the cost also became a major concern.

This trade off has had a great impact in the security aspect of IoT applications. In pursuing the compatibility to existing Internet protocols, the IoT community has made many efforts in porting Internet security measurements such as Secure Sockets Layer (SSL)[5]/Transport Layer Security (TLS)[6] into the IoT scenarios over the years, yet we have not seen many substantial

success in this approach. Many solutions soon lose their practicability mostly due to their unaffordable overhead to the constrained environments of IoT.

As a result, in recent years, people have started investing in new security standards that are designed specifically IoT scenarios, such as the NIST Light-Weight Cryptography (LWC) competition[7] kick-started in 2015. This research is thus driven by the demand of new security solutions for IoT. We put our focus on addressing the threats in IoT applications, especially in terms of side channel attacks.

1.2 Research Contributions

The contribution of this thesis is our study towards various types of side channel attacks that could breach the security of IoT application in practical scenarios. Notably, we highlight the security issues we present in Chapter 3 and Chapter 6 as they are identified on devices that are popular among the research community as well as the market. This thesis also features a variety of different side channel information being exploited, from physical leakage such as radio frequency (Chapter 3) and power consumption (Chapter 4 and Chapter 5), to non-physical leakage such as packet metadata (Chapter 6).

1.3 Thesis Outline

A wide range of security concerns in IoT application has been involved in this thesis. We begin the thesis by introducing the preliminaries in Chapter 2, followed by a case study on a classic IoT device, TI CC2538, where we found several flaws in its Random Number Generator (RNG) design which is unsuitable for any cryptographic implementation to rely upon. We then move on to the aspect of a major security threat against IoT devices, i.e. power analysis, which any cryptographic implementation should withstand for embedded systems. In Chapter 4 we provide a thorough

report on applying typical Correlation Power Analysis (CPA) against a specific type of lightweight cipher for IoT, named ARX, and showed that its so claimed "inherent resilience against side channel attacks" is unreliable when it is implemented naively. A novel chosen message Differential Power Analysis (DPA) strategy is presented in Chapter 4. Chapter 5 proposes a novel type of power analytical distinguisher that exploits the ordering of leakage to recover the secret keys. Ordinal distinguishers fall into the category of generic distinguisher which is non-profiling DPA methods that is robust when the leakage behaviour of target device is unknown; thus effective against the emerging IoT devices. Chapter 6 shows our study towards the practicability of traffic analysis, a non-physical type of side channel attack, in IoT scenarios. Unlike power analysis, traffic analysis exploits the packet metadata that are unprotected by the cryptographic schemes and directly reveals information inside the encrypted data; thus they must be taken into account to IoT application developers as the security may still be breached even if the underlining cryptography are utilised properly. We finally conclude the thesis in Chapter 7.

1.4 Publications

Part of the work presented in this thesis has been published in the proceedings of various conferences. Chapter 3 and Chapter 6 are based on our published results which are:

1. **Cryptographic Randomness on a CC2538: A Case Study**[8]
   Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Cryptographic randomness on a CC2538: A case study'. In: *IEEE International Workshop on Information Forensics and Security, WIFS 2016, Abu Dhabi, United Arab Emirates, December 4-7, 2016*. IEEE, 2016, pp. 1–6. ISBN: 978-1-5090-1138-4. DOI: 10.1109/WIFS.2016.7823912. URL:

https://doi.org/10.1109/WIFS.2016.7823912

Author Contribution This work was co-authored with my supervisors Elisabeth Oswald and Theo Tryfonas. As the main author, I was responsible for all major works. This includes investigating the Pseudo Random Number Generator (PRNG) driver and reverse engineering the True-RNG (TRNG) design of CC2538, as well as proposing the attack vector against its Radio Frequency (RF) seed.

2. **Exploring Potential 6LoWPAN Traffic Side Channels**[9]
   Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Exploring Potential 6LoWPAN Traffic Side Channels'. In: *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. EWSN ’18. Madrid, Spain: Junction Publishing, 2018, pp. 270–275. ISBN: 978-0-9949886-2-1. URL: http://dl.acm.org/citation.cfm?id=3234847.3234911

   Author Contribution This work was co-authored with my supervisors Elisabeth Oswald and Theo Tryfonas. As the main author, I was responsible for conducting all experiments presented in the paper as well as analysing the data. I am also responsible for proposing the novel application fingerprint attack exploiting in [9].

Contents in Chapter 4 and Chapter 5 are not published as of writing this thesis. My contribution in these works are:

1. **Chapter 4**: This work was co-authored with my supervisor Elisabeth Oswald and college Srinivas Venkatesh. As the main author I was responsible for conducting all related experiments and proposing the chosen message DPA strategy. Our college Srinivas Venkatesh has contributed in part of the mathematical formalisation in this work.

2. **Chapter 5** This work was done with my supervisor Elisabeth Oswald and colleges Arnab Roy and Srinivas Venkatesh. In this work, I was responsible of proposing our new distinguishers and their implementations as well as conducting all the experiments.

Chapter 2

Preliminaries

This thesis contains various topic related to security of IoT. In order to make this thesis reasonably self-contained, preliminary work required to understand the overall background related to the topics is provided here. Topic specific preliminaries are provided at the beginning of each chapter.

2.1 Basic Statistics and Information Theory

Statistical techniques and concepts from information theory are frequently used in this thesis. This section reviews some fundamental concepts which are covered by any statistical textbook such as [10] and [11].

2.1.1 Cumulative Distribution Function

Cumulative Distribution Function (CDF) is one of the most common way when describing a distribution of a random variable. For a real value random variable X, its CDF $D_X(x)$ is defined as:

$$D_X(x) = Pr(X \leq x)$$

Hence by definition, it holds that:

$$Pr(a < X \leq b) = D_X(b) - D_X(a)$$

An estimation of CDF based on sampled data is referred as an Empirical Cumulative Distribution Function (ECDF).

2.1.2 Probability Density Function

For a continuous random variable X, the Probability Density Function (PDF) is defined as the derivative of its cumulated distribution function $D_X(x)$:

$$PDF_X(x) = \frac{d}{dx} D_X(x)$$

And equivalently:

$$D_X(x) = \int_{-\inf}^{x} PDF_X(t)dt$$

2.1.3 Pearson Correlation Coefficient

Pearson correlation coefficient, also referred as Pearson product-moment correlation coefficient, is a widely used measurement to evaluate the linear correlation between two variables X and Y. It is defined as:

$$p_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where σ_X and σ_Y are the variances of X and Y, and $cov(X,Y)$ the covariance of X and Y. When given two sample sets $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_n\}$, Pearson correlation coefficient can be computed by:

$$p_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of X and Y.

Pearson's correlation is a value in range $[-1, 1]$ where 0 implies X and Y are uncorrelated. A correlation of 1 indicates a perfect positive linear relation whereas -1 indicates negative one.

2.1.4 Shannon's Entropy

Shannon's entropy is a common method to evaluate the uncertainty of random variables. For a discrete random variable x sampled from a distribution X, the Shannon's entropy is defined as (using binary logarithm):

$$H(X) = -\sum_{x}^{x \in X} Pr(x) \log_2 Pr(x)$$

Entropy is non-negative and intuitively a higher $H(X)$ implies more uncertainty of X. The term entropy in this thesis always refers to Shannon's entropy, unless otherwise stated.

2.1.5 Mutual Information

Mutual Information is a quantity that measures the dependency between two variables. For two variables $x \in X$ and $y \in Y$, their mutual information is defined as:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X,Y) - H(X) - H(Y) \end{aligned}$$

Intuitively, a higher mutual information implies more dependency between the variables.

2.1.6 Kolmogorov–Smirnov Test

Kolmogorov–Smirnov test (KS test) is a non-parametric statistical that compares a sample with a reference distribution (one-sample KS test), or compare two samples (two-sample KS test). Most application of KS test in

this thesis are two-tailed two-sample KS test, i.e., we are mostly interested in testing whether two samples are sampled from the same distribution. The statistics of two-sample KS test, also referred KS distance, is defined as:

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$$

where $F_{1,n}(x)$ and $F_{2,m}$ are the ECDFs of the samples being tested and sup the supremum function. Intuitively, higher $D_{n,m}$ implies less likely the samples are equal and vice versa.

2.1.7 Signal-to-Noise Ratio

Signal to Noise Ratio (SNR) is the measurement to evaluate a desired signal compared to back ground noise. It is defined as:

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}$$

where σ^2_{signal} and σ^2_{noise} are the variances of desired signal and noise respectively.

Specifically, the following estimation proposed in [12] is also widely used in power analysis which we explain more details later in Section 2.6:

$$SNR = \frac{Var(E(T|X))}{E(Var(T|X))}$$

where T is the set of leakage values on power traces and X the set of target intermediates.

2.2 Contiki OS

Contiki OS[13] is an open source Operating System (OS) dedicated to IoT devices that is highly optimised towards energy efficiency. The OS also features fully support for IP-based networks (IPv4 and IPv6). It has been

widely used among the WSN academic research community. There are several similar open source embedded OS such as Tiny OS, OpenWSN and RTOS, etc.

Contiki OS has a wide support for various market available System on Chip (SoC)s for IoT. In my work, Contiki OS is mainly used in conjunction with two specific devices which are CC2538[14] and TelosB[15].

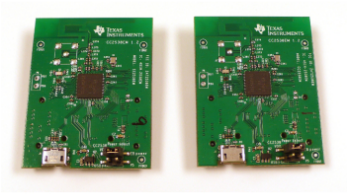
2.3 CC2538 and TelosB

CC2538[14] is a SoC for IoT launched by Texas Instruments (TI) in 2013. It features high end computational performance powered by an ARM Cortex-M3 Micro-Controller Unit (MCU), low power consumption and IEEE 802.15.4[16] compliant RF transceiver. In addition, CC2538 has hardware support for cryptographic operations including AES[17], SHA2[18], RSA[19] and Elliptic Curve Cryptography (ECC)[20] accelerators, which suggests its use in secure IoT applications. As a result, the chip features in the suggested list for Zigbee[21] and 6LoWPAN[22] solutions on TI's website[23], and projects such as Contiki[13] and OpenWSN[24] began to support the CC2538 with enthusiasm.

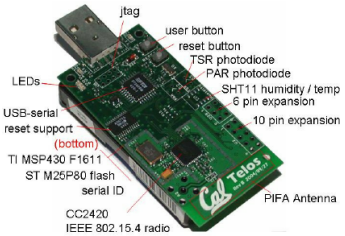
On the other hand, TelosB[15] (also known as Sky mote) is a low cost open source SoC published by UC Berkeley with 802.15.4 support. Compared to CC2538, TelosB is more of a low end device with a less powerful MSP430 MCU and less RAM. The IoT related experiments in this thesis are focused on these devices as they represent a wide range of similar platforms. A full list of supported platforms of Contiki OS can be found in [13].

2.4 SCALE Board

The SCALE board [25] (Figure 2.2) is an open source test board dedicated to side channel and fault attack analysis. The board contains an on board



(a) CC2538 Evaluation Module



(b) TelosB

Figure 2.1: Devices used with Contiki-OS

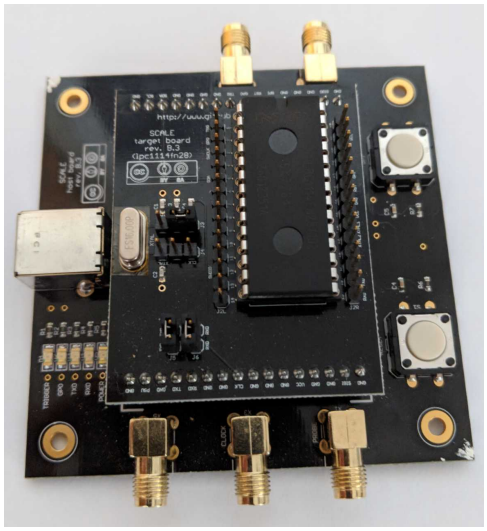


Figure 2.2: A NXP LPC1114fn28 (ARM Cortex M0 architecture) housed on a SCALE board (Photo taken by Joey Green)

oscillator (clocked at a rate of 16MHz), and supplies the chip with power either via a dedicated power supply or via USB. To aid power analysis the board directly taps into the core's supply voltage and amplifies it, thereby facilitating a convenient measurement point suitable for a standard probe.

All real power traces used in Chapter 4 and Chapter 5 are collected on a NXP LPC1114fn28 (ARM Cortex M0 architecture) housed on the SCALE board.

2.5 Side Channel Attacks

Securely implementing cryptographic primitives is a difficult task and vulnerabilities are constantly found on even the latest digital products. Not only because the algorithms are being badly implemented [26][27][28], confidentiality of cryptographic secrets could also be easily compromised when an adversary is given additional execution information such as the execution time [29] and traces of power consumption [30]. In a cryptography context, these attacks are referred to as Side Channel Attacks (SCA) and they have been proven to be a great threat to embedded systems, which are the major components in any IoT applications. The information being exploited by these attacks is usually referred to as side channel leakage (or simply leakage) in side channel literatures.

The concept of side channel attacks can be further extended to wider scenarios where the adversary aims not only to recover the cryptographic keys but also to directly extract information regardless of data being encrypted, such as the attacks described in [31] and [32]. The work presented in this thesis mainly involves two specific types of side channel attacks, DPA and Traffic Analysis (TA), which we provider more details later in Section 2.6 and Section 2.7.

2.6 Power Analysis Attacks

Power analysis attacks are based on the fact that processor consumes different amount of energy depending on the data being processed. The landmark work done by Kocher et al. published in 1999 has for the first time demonstrated the potential of recovering the secret key through analysing the traces of power consumptions during the execution of a cryptographic algorithm (see Figure 2.3 for an example).

Power analysis attacks often require the adversary to have physical access

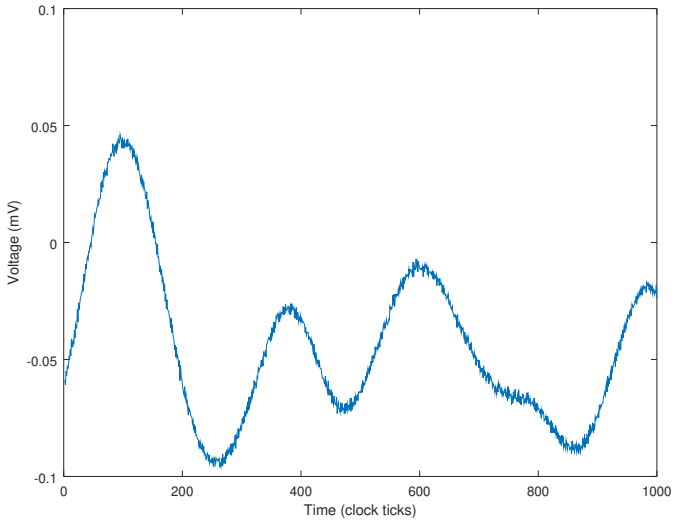


Figure 2.3: An example: power trace of a round of SPARX cipher on SCALE board

to the device and it is generally assumed that the adversary collects the power traces together with known plaintext and/or ciphertext. Power analysis is particularly critical to the security of embedded devices, such as smart cards , microcontrollers or Radio-Frequency IDentification (RFID) widely used in IoT applications, as in many cases they are deployed into environments which lack physical protection.

A significant amount of work has followed the idea of [33] over the years and power analysis has become one of the most explored subjects among different side channel attacks. Power analysis is generally categorised into two classes:

Simple Power Analysis (SPA) SPA tries to derive the key directly from only few or even just one trace. SPA is normally challenging in practice as the adversary needs to know all the details of the cryptographic implementation, and extracting the signals in the trace is also a complex task [30].

Differential Power Analysis (DPA) In contrast to SPA where the adversary aims to derive the key using only a few traces, DPA utilises many traces and performs statistical tests to recover the key. DPA is generally considered more robust than SPA as it requires less detailed knowledge of the implementation and can cope with extremely noisy traces [30].

The work presented in this thesis is mostly concerned about the impact of DPA attacks against IoT devices. We address more details in this section.

2.6.1 Univariate vs Multivariate

In a DPA attack, a power trace $\mathbf{L}$ recorded by the adversary contains many leakage points of real values:

$$\mathbf{L} = (L(v_1), L(v_2), \dots, L(v_i), \dots)$$

whereby each point corresponds to an intermediate value (or simply briefed as intermediate) v_i as it is defined by the implementation of a cipher. L is the leakage function, and a generally accepted form of defining L is a linear combination of a deterministic and a non-deterministic component:

$$L(v) = M_D(v) + \epsilon$$

where M_D is a device leakage model (or just leakage model) which is characterises the target device, and ϵ is an independent random noise sampled from some Gaussian distribution $\mathcal{N}(0, \sigma^2)$. In practice, M_D could be a very complex function and no a priori knowledge is available to the adversary.

It is generally assumed that the power traces are aligned, i.e., the same index in the traces implies the same operation in the cryptographic algorithm. A standard DPA style attack is based on this assumption and is typically univariate, i.e. a single leakage point from each leakage trace together with a

subkey dependent hypothesis on an intermediate value (the so-called target function) form the input to a mathematical function, which is called the side channel distinguisher (or simply distinguisher) [34][1]. This single point is typically unknown, and thus the distinguisher is independently applied to all points in a leakage trace. A successful attack will both recover a portion of an unknown key as well as the trace point(s) at which the targeted key-dependent value has been computed. Some typical distinguishers are explained later in this section.

Clearly an adversary could utilise the leakage of all leakage points simultaneously as in [35], but such attacks, called multivariate attacks, require knowledge of the precise locations of all targeted intermediates and their respective leakage models [36]. Consequently, these attacks place considerable demands on what real-life leakage adversaries can obtain and thus are seen as complementary to standard DPA style attacks.

In this thesis, a univariate attack is always assumed, unless stated otherwise. A power trace is hence denoted as a pair (x, t) where x is the known input/output and t the leakage value of intended target intermediate v. We denote:

$$t = L(v) = L(F_{k^*}(x))$$

where F_{k^*} is the target function F embedded with the secret key k^*.

2.6.2 Side Channel Distinguishers

Side channel distinguishers (or simply distinguishers) are mathematical functions (usually based on statistical tests) in DPA attacks that takes a key hypothesis, a.k.a. key guess, and leakages value as inputs, and outputs a distinguishing score corresponding to the hypothesised key; thus giving a ranking on the key hypothesis. Without loss of generality, in this thesis, distinguishing scores are always defined such that a higher distinguishing

[1] It is accepted to use the term DPA attack irrespective of the distinguisher.

score is always considered more likely the hypothesised key equals to the targeted secret key and vice versa.

There are many distinguishers defined in DPA literatures. Here we introduce the classification proposed by [37] with instances including the most well known distinguishers described in [30].

Partition-based Distinguisher

A partition-based distinguisher defines a partition $\mathcal{P}_k$ of the leakage values according to hypothesised intermediates computed from the hypothesised key k and the known input x of the traces. Then, the partitions for each key hypothesis are checked with statistical tests. The key associated with the the most meaningful partition with respect to the real physical leakage is finally returned as the best key guess.

Example 1. (**Single-bit DPA**) One of the most well known distinguishers of this type is the Single-bit-Difference-of-Means DPA attack [30], referred as Single-bit DPA in this thesis. For this distinguisher, the adversary selects a bit of the target intermediate and partitions the leakage values based on the value of the target bit. For example, denote by $B(v)$ the bit selection function that returns a specific bit of b, the adversary defines the partitions for a key hypothesis k as:

$$P_0 = \{t : B(F_k(x)) = 0\}$$
$$P_1 = \{t : B(F_k(x)) = 1\}$$

And the distinguishing score for key guess k is defined as:

$$D_k = |\bar{P}_0 - \bar{P}_1|$$

where $\bar{P}_0$ and $\bar{P}_1$ are the sample means of P_0 and P_1.

The intuition of this distinguisher is simple: it exploits the difference in

the leakage of the selected bit and the fact that the correct partition will maximise such difference.

Another type of well known distinguisher in this class is generic distinguisher, which we explain in details in the related Chapter 5.

Comparing-based Distinguisher

For comparing-based distinguishers, the adversary selects a prediction model M and predicts the leakage based on the intermediates computed with the key guess. Then, the predicted leakage for each key guess is compared to the leakage values of collected traces using statistical tests. The best key guess is finally returned as the key guess that is most similar leakage prediction to the actual leakage.

The selection of the prediction model M strongly relies on the adversary's knowledge to the target device. However, for an attack, only relative difference between predicted leakages are important [30]. Without a priori knowledge of the target device, the most commonly used power models are the Hamming Weight (HW) and Hamming Distance (HD) models, and their implications of leakage are justified in [30]. Note that in order to use the HD model, the adversary is additionally required to predict the preceding or succeeding intermediate; therefore the HW model is preferable in many cases.

Example 2. (**Pearson's Correlation**) The most commonly seen distinguisher is (the absolute value of) Pearson's correlation using HW prediction. Given a set of traces $\{(x_i, t_i)\}$, for a key guess k, the adversary predicts:

$$t'_i = HW(F_k(x_i))$$

And the distinguishing score for k is defined as:

$$D_k = |p_{T,T'}| = |\frac{\sum_{i=1}^{n}(t_i - \bar{t})(t'_i - \bar{t'})}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2}\sqrt{\sum_{i=1}^{n}(t'_i - \bar{t'})^2}}|$$

where $T = \{t_i\}$ and $T' = \{t'_i\}$, and $\bar{t}$ and $\bar{t'}$ are the means of T and T'.

Due to the popularity of Pearson's correlation, it is sometimes ambiguously referred as CPA in some literatures.

Note that these classifications are not exclusive. For example, Pearson's correlation can be utilised in a "generic emulating" manner as a single-bit distinguisher (i.e. only one bit of v is taken as model M). However, the fact that it then only operates in a single bit manner implies a loss in efficiency. On the other hand, single-bit DPA can also partition the traces based on a sophisticated leakage prediction model rather than simply selecting a bit in the target intermediate.

The performance, or efficiency, of distinguishers are generally evaluated by the number of traces required to recover the secret key. Note that single-bit DPA and Pearson's correlation normally only assumes the adversary being able to passively observe the traces. In case of a more powerful adversary that has full control over a device identical to the target, template attacks [38][39] are another option that efficiently recover the secret key in general.

Template Attack The most distinctive character of template attack is the profiling stage that builds a template of target device before the attack. The template is constructed through traces collected on a device identical to the target with known secrets and thus known intermediates. During the attack, the adversary matches traces collected on the target devices to the template built beforehand and derives the key.

There are various methods for profiling as well as matching the template, e.g. one may use the intermediate values for profiling and then matches

the template using least-square strategy [30]. It is worth mentioning that due to the natural similarities, there is an emerging trend of utilising machine learning technologies in DPA attacks [40] [41]. However, no significant improvement was reported utilising machine learning techniques to my knowledge.

Since the capability of profiling is such a strong assumption to the adversary's power, template attacks (and their variations) are also referred to as profiling attacks, in contrast to the non-profiling attacks such as single-bit DPA (Example 1) and Pearson's correlation (Example 2). In practice, profiling attacks also suffers from the misalignment of acquisition setups between the profiling phase and attack phase. Therefore, this thesis mainly focuses on non-profiling attacks.

2.7 Traffic Analysis

Traffic analysis is well studied in the context of encrypted Internet traffic, especially for web applications based on HTTPs and TCP/IP. In general, rather than attacking the cryptographic primitives employed by the security protocols, traffic analysis exploits side channel information that are not protected by the encryption, such as packet length, responding time, unencrypted packet headers, etc. These attacks typically require the adversary to have some priori knowledge of the application being attacked, for instance, the set of potential web sites the victim may visit, or the available options on a specific web site[32]. The adversary recovers information related to the encrypted data through correlating the unprotected side channel information to the known options those may be adopted by the victim.

The landmark study by Chen et al. [32] discussed different side channel attacks against web applications and [42] studied the practicability of an attack specifically targeted Google and Bing search boxes. Later work by Mather and Oswald [43] proposed the use of Mutual Information to pinpoint

the potential leakage points in web traffic. For non-HTTPs applications, the papers [44], [45] and [46] described attacks against encrypted text, voice and video traffic respectively.

Machine learning is widely used to analyse the traffic, and behaviours of different classifiers are studied by [47] and [48]. Based on all these published works we can conclude that two features, the packet length and response time, are the most exploited among all attacks. Different countermeasures are proposed accordingly, such as Traffic Morphing [49], HTTPOS [50] and Format-transformation Encryption [51].

2.8 Lightweight Cryptography

Lightweight cryptography refers to cryptographic schemes that are suitable for extremely resource constrained environment and its development is mostly driven by different demands of IoT application. As of the variety of IoT applications, the so called "lightweight" is defined differently according to different context. Without loss of generality, lightweight ciphers are normally optimised towards the following aspects:

- **Throughput** evaluates the amount of plaintext that can be processed per time unit.
- **Latency** which evaluates how quick the invoker of the cryptographic algorithm can get an output.
- **Energy consumption** evaluates the amount of energy consumed by the cipher.

Additional properties are evaluated specifically for hardware or software oriented designs. For hardware, **Gate Equivalent (GE)** reflects the size of area needed to implement the cipher. For software, memory consumption and code size are evaluated.

There is hardly a uniformly best cipher in IoT, as different properties are desired in different use cases. For instance, energy efficiency is essential in WSN applications so ciphers consuming the least energy is desired. VANET have strict requirements to the response time of the devices and thus latency is prioritised. In case of RFID, small sized hardware implementation is preferred to bring down the cost, as well as the capability to operate with minimum energy when the device is only powered passively.

The National Institute of Standards and Technology (NIST) lightweight cryptography project started in 2015 [7] and eventually became one of the most remarkable event in the development of lightweight cryptography by driving a significant attention into this field. For asymmetric cryptography, lattice-based schemes are one of most popular paradigms in recent years. [52] proposed an efficient encryption scheme that is even capable to run on an 8-bit AVR processor, and the one proposed by [53] highly optimises the circuit size for a hardware implementation. Other cryptographic primitives are also proposed in related literatures such as the signature scheme of [54]. [55] provides a nice survey to the practicability of lattice-based cryptography in IoT scenarios and [56] provides a dedicated report on the energy consumption of different lattice-based schemes.

Despite the boost of performance in recent years, many asymmetric schemes are still far from practical in the extremely resource constrained scenarios, such as wireless sensors and RFID. For example, the minimum signature size achieved in [54] is 5600 bits, i.e., 700 bytes, whereas the Maximum Transmission Unit (MTU)[2] specified for 6LoWPAN network is 129 bytes [22], where the payload consumed by headers required to transmit the key are not even counted. Such discrepancy between proposed schemes and existing standards remains an open issue to date for asymmetric cryptography; therefore in this thesis we focus on the relatively concrete side of symmetric

[2]The maximum amount of data that can be transmitted in one transmission.

cryptography. A thorough survey for this topic is done by [57].

The question people may ask when arguing the need of lightweight symmetric cipher is why AES[17] is not enough? It is indeed that AES excels in many aspects and it could be used whenever applicable, for instance, it is adopted by the 802.15.4 standard[16] as the only cryptographic primitive for link layer encryption as well as being shipped as the built-in cryptographic coprocessor for TI CC2538 and several other products in that series. However, as argued in [57], for extremely resource constrained applications, improvements should be made whenever possible. [57] has listed several drawbacks of AES that makes people rethink whether we should relying on it, where the major problems are related to its 8-bit S-Box structure. For software implementations, either storing the 8-bit S-Box or code that generates it on the fly is memory consuming and consequently energy consuming. Also due to its large block size and look-up based S-Box designs, hardware implementations can hardly achieve a GE less than such as 2000 with reasonable countermeasures, or efficiently masked in software in terms of code size and/or memory. All these issues therefore left space to be improved by lightweight ciphers.

As of many existing ciphers, lightweight symmetric ciphers also consist of linear components and non-linear components, where the latter are generally the more interesting target for side channel analysis as pointed out by [58]. The most popular design paradigms are listed in [57], such as the S-Box-based constructions including look-up table and bit-slice. This thesis focuses on the other non-S-Box option, ARX, where more details are given in Section 2.8.1. A study towards the side channel aspects of the S-Box-based constructions is done by [59].

2.8.1 ARX Ciphers

ARX cipher refers to the family of ciphers that base their round function on the simple combination of modular addition, rotation, and Exclusive-or (XOR). The idea of combining addition modulo 2^n, XOR, and rotation as a round function, has been suggested as early as 1987 in the block cipher FEAL[60]. The appeal of this construction is primarily in the fact that when choosing n equal to the word size of a processor, software implementations gain considerable speed-ups.

Round functions in ARX ciphers have efficient and simple expressions via functions that are typically available as instructions on small embedded devices. This enables excellent performance both with respect to execution time and energy consumption, which makes ARX ciphers very appealing to the IoT scenario. Specifically, for software implementations on microcontrollers, modular addition is supported by most, if not all, processors and can be performed very efficiently without utilising additional registers [57]. Some recent examples of ARX ciphers include Chacha20 [61] and Salsa20 [62] family stream ciphers, SHA-3 finalists BLAKE [63] and SKEIN [64], as well as other block ciphers such as SPECK [65] and SPARX [66], etc. Here we specifically address SPARX as it is the first instance of this type of cipher that has a proved bound against differential and linear cryptanalysis [66], whereas justifying the security of ARX ciphers is generally more difficult due to its lack of S-Box and only uses modular addition for non-linear layer [57].

Side Channel Attacks on ARX

The absence of look-up tables has left ARX with an impression of having certain inherent resilience against side channel attacks. For example, the authors of SPARX [66] stated:

> “ *... The choice of using the ARX paradigm was based on three observations. First, getting rid of the table look-ups, asso-*

ciated with S-Box based designs, increases the resilience against side-channel attacks. Second, this design strategy minimizes the total number of operations performed during an encryption, allowing particularly fast software implementations. Finally, the computer code describing such algorithms is very small, making this approach especially appealing for lightweight block ciphers where the memory requirements are the harshest. " (Section 1, [66])

An earlier [67] research also stated:

> " *Some algorithms or SHA-3 candidates (i.e. BLAKE or CubeHash) do not use such substitution table, while they rely exclusively on modular addition $\boxplus$, rotation $<<$ and XOR $\oplus$ operations (so-called ARX constructions). In this case, side-channel analysis is still possible but the XOR or modular addition selection functions are less efficient than for the Sbox case. Moreover, it has been theoretically proven that the XOR selection function is less efficient that the modular addition operations. Indeed, the propagation of the carry in the modular addition leads to some non-linearity whereas the XOR operation if completely linear.* " (Section 2.2, [67])

Furthermore, because the non-linear component is given by the addition modulo 2^n, it does not need to be encoded as a table lookup and it is arguable that the ARX instructions take almost constant time on most platforms. When considering cache timing attacks, the absence of tables is a distinctive advantage, as stated in [68] and [69]. Remarkably, [70] has quantified the difficulty of attacking different instructions utilised in an ARX ciphers in terms of Pearson's correlation (see Example 2) and concluded that even the most effective target, which is the only non-linear operation, i.e., modular addition, does not seem effective enough to mount a practical attack.

As of writing the thesis, there have been minimal successful side channel attacks on ARX ciphers. The most significant result to our knowledge is q[71] which demonstrated that it is possible to improve on straightforward DPA style attacks when targeting the modular addition in SKEIN [64]. In [71], the authors observed that the symmetrical structure of modular addition eventually results into a pair of correlation peaks; therefore the performance of an attack can be improved by testing pairs of correlations rather than a single correlation. However, attacks like [71] still face the practical problem that the number of key hypotheses that are tested via the target function increases exponentially with the operand size (the detailed reasoning is given in Chapter 4). For example, in the case of a 32-bit modern processor such as an ARM-M0, performing a DPA style attack requires the adversary to enumerate both 32-bits adders which has a space complexity of 2^{64}. To solve this issue, [71] assumes a stronger adversary that is capable of choosing the plaintext to be encrypted, comparing to the classic settings in Example 1 and Example 2 where the DPA adversary only passively collect traces.

These results were partly the motivation of the work in Chapter 4 and Chapter 5 in this thesis, where a thorough analysis is given on this topic.

Side Channel Countermeasures of ARX

Countermeasures such as masking (secret sharing) are well understood and costly [72]. Specifically, protecting ARX ciphers has been considered even more costly by many literatures, for example:

> “ *When variables are added or subtracted, they must be available in arithmetic masking form and so will be the result; when they are XORed, rotated or shifted, they must be available in Boolean masking form and so will be the result. The problem is now when variables undergo operations of different types; namely when we have a variable in Boolean masking form and it must be*

> *added to another variable, or vice versa, when we have a variable in arithmetic masking form and it must be rotated or XORed to another variable. The only solution to that problem identified up to now is converting a variable in Boolean masking form to arithmetic masking form or vice versa when needed. The number of such conversions required depends on the way the different operations are alternated in the algorithm. This can be quite often as most ARX algorithms get their non-linearity exactly from this alternation.* " (Section 3, [73])

and

> " *If the algorithm uses an ARX structure, then it is possible to use the arithmetic structure of modular addition to mask this operation directly. Rather than masking each carry propagation bit separately, we can instead generate secret shares and combine them using modular addition directly. The problem is then in the interaction between the masking of the modular addition and the masking of the linear part because they are done in different groups.* " (Section 5.1.4, [57])

It is indeed true that for ARX constructions, one has to cope with the fact that there are Boolean operations (requiring Boolean masking or secret sharing) and arithmetic operations (requiring arithmetic masking or secret sharing). Several efforts have been made attempting to mitigate this issue including [74] and [75]. The latest result of this topic comes from [76] where the authors combined several optimisations based on the results of [75] and [77]. [76] achieved a 36% speed improvement on ChaCha20 comparing to [77] by reducing some of the redundant instructions as well as the randomness required in [77]. The authors eventually reported an overhead of 4.5 fold to code size and 35 to 42 times in clock cycles for their optimised ChaCha20

implementation [76].

Chapter 3

Flawed Designs: Cryptographic Randomness on CC2538

3.1 Introduction

RNGs are one of the most critical component for all cryptographic schemes. A typical usage of RNG is the secret key generation which is at the heart of any security. RNGs are also widely needed in other scenarios. For example, in generating the nonces for ElGamal[78] and Schnorr[79] signatures, or sampling the errors for Learning With Error (LWE)[80] based schemes([81], [82], [83], etc). As such, designs of RNG must be addressed with care, as its failure would immediately compromise any cryptographic scheme relying on the randomness it generates.

On the other hand, our study finds that some classic IoT devices advertising for security usage failed to deliver a qualified RNG. In this chapter, we present a case study on TI's SoC CC2538[14], a popular device featuring cryptographic support that has been widely used in WSN applications. We

show its vulnerabilities in various aspects that could have devastating consequences. The combination of Contiki OS on CC2538 is a frequently seen platform in many IoT applications among academic and industry.

Note that similar vulnerabilities have been previously reported by [84] in 2010 on some predecessors in the series of CC2538. Essentially, [84] explained that how the vulnerable PRNG can be exploited to break the ECDSA signature in a smart meter application, as well as proposed the potential of biasing the radio based TRNG. However, [84] did not provide an implementation of the radio biasing attack of which blank we have filled in this work.

This work was done in conjunction with E. Oswald and T. Tryfonas. It has been published in:

Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Cryptographic randomness on a CC2538: A case study'. In: *IEEE International Workshop on Information Forensics and Security, WIFS 2016, Abu Dhabi, United Arab Emirates, December 4-7, 2016*. IEEE, 2016, pp. 1–6. ISBN: 978-1-5090-1138-4. DOI: 10.1109/WIFS.2016.7823912. URL: https://doi.org/10.1109/WIFS.2016.7823912[8]

As the main author I was responsible for all main aspects of the work. This includes investigating the software aspect of Contiki[13] drivers for CC2538, reverse engineering the RNG-related circuits on the device, and proposing as well as implementing the non-invasive attack that biases the RNG.

3.2 Preliminaries

3.2.1 Random Number Generators

RNGs are widely used in cryptography. The quality of the random numbers generated are often critical to the security of cryptographic schemes. For

example, a key generated by a badly designed RNG would greatly endanger the encryption scheme relying on its secrecy.

In practice, RNGs are usually implemented as a PRNG seeded by a TRNG, which we briefly introduce in this section.

TRNG

According to [85], TRNGs are procedures of producing totally unpredictable bits by extracting randomness from physical processes that behave in a fundamentally nondeterministic way. Due to their physical nature, sometimes they are also refereed as physical or hardware RNGs.

TRNGs can be implemented in various ways utilising different physical features, which are referred to as sources of randomness or entropy in some literatures. Some instances of TRNG implementation include:

- Noisy electrical circuits[86][87][88]. This approach has been commonly seen on security critical smart card applications.
- Noises sampled from various drivers. This approach has been adopted by the Linux kernel[89] for desktops.
- Radio noises sampled in the air[90][91]. This approach is commonly seen in radio equipped devices such as wireless sensors.

PRNG

Although random numbers can be obtained by a TRNG on its own, their efficiency and randomness are often bounded by the properties of their physical entropy source. Therefore, in practice, TRNGs are mostly used only to generate seeds that are used to initialise the PRNGs, which then generate the randomness as requested by applications.

Cryptographically, [92] has provided the following formal definition of a PRF:

Definition 1. *Let l be a polynomial and let G be a deterministic polynomial-time algorithm such that for n and any input $s \in \{0,1\}^n$, the result $G(s)$ is a string of length $l(n)$. We say that G is a pseudorandom generator if the following conditions hold:*

1. *(Expansion:) For every n it holds that $l(n) > n$.*

2. *(Pseudo-randomness:) For any Probabilistic Polynomial Time (PPT) algorithm D, there is a negligible function $negl$ such that*

$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq negl(n)$$

, where the first probability is taken over uniform choice of of $s \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $r \in \{0,1\}^{l(n)}$ and the randomness of D.

We call l the expansion factor of G.

The research of constructing cryptographically secure PRNGs remains active nowadays and new constructions are constantly being proposed, such as [93] [94] and [95]. Notably, constructions based on keyed Hash Message Authentication Code (HMAC) or block ciphers (typically AES[17]) as recommended by the NIST document [96] are among the most widely adopted approaches.

3.2.2 Randomness Validation

There are many methods proposed to validating the randomness of PRNGs. The NIST random bit test suite[97] implements a collection of statistical randomness test and is widely used by researchers. A full list of the tests performed by the NIST test suite can be found in [97].

3.3 Related Work

Two siblings of CC2538, CC2430[98] and CC2530[99], have been previously reported of using a 16 bit Linear Feedback Shift Register (LFSR) as PRNG in [84] and [100]. These chips are the predecessors of CC2538 in the SimpleLink^{TM}series and the same RNG designs, including both TRNG and PRNG, were shared among them. The blogs reported the problems also warned that it could easily be exploited to compromise the Z-Stack library[101] and Smart Energy Profile ECC in many Smart Meter applications. Although the potential of biasing the TRNG by injecting a jamming signal was also contemplated, the blogs did not provide any design of the jamming signal as well as an implementation. To this end, we examine the technical feasibility and demonstrate a working attack in this chapter.

Although CC2538 has hardware support for various cryptographic operations (see Section 2.3), there is no dedicated RNG provided for cryptographic applications. Instead, the user guide[102] suggests the use of an onboard 16 bit LFSR as a PRNG which is seeded by the RF module through sampling radio noise. Whilst the user guide at no point suggested that this method should be used in conjunction with cryptographic algorithms, developers have little choice in the absence of alternatives.

3.4 RF Noise as TRNG

In this section, we present the work of reverse engineering the TRNG design of CC2538. Unlike many higher end devices, such as security ICs, CC2538 does not come with a dedicated TRNG. The only option provided in the user guide is to use the RF module to sample radio noise as random bits:

> “ *For the CC2538, when a random value is required, writing the SOC_ADC_RNDL register with random bits from the IF_ADC in the RF receive path seeds the LFSR. ... (Section 23.12) Single*

> *random bits from either the I or Q channel can be read from the RFRND register.* ” (Section 16.2.2, [102])

The user guide[102] also reports on the good quality of the randomness:

> “ *Randomness tests show good results for this module. However, a slight DC component exists. In a simple test where the RFRND.IRND register was read a number of times and the data was grouped into bytes, about 20 million bytes were read. When interpreted as unsigned integers between 0 and 255, the mean value was 127.6518, which indicates that there is a DC component. ... For the first 20 million individual bits, the probability of a 1 is* $P(1) = 0.500602$ *and* $P(0) = 1 - P(1) = 0.499398$. ” (Section 23.12, [102])

and their test results are shown in Figure 3.1.

To verify the claims in the manual, we applied the NIST Statistical Test Suite[97] on 13263600[1] bits sampled by this seeding method in a common office environment with multiple wireless devices (smart phones and laptops, etc) activated. Since the Contiki driver only uses the bits generated in I channel and one bit is returned upon each read to the RNG register; therefore we concatenated all bits into one bit stream. The bits passed all tests in the NIST test suite, with $P(0) = 0.49995001$ and $P(1) = 0.50004999$, which shows that the RF noise (when not tampered with) is indeed a good source for random numbers. However, it remains unclear whether such source can practically be influenced by crafted RF signals.

3.4.1 Reverse Engineering the TRNG Design

The documents supplied by TI do not explain further details of how IF_ADC in the receive I/Q channels are translated to random bits. We have neither

[1]There is no standardised guile line for number of bits to be tested. However, we consider this value as a sufficiently large sample size.

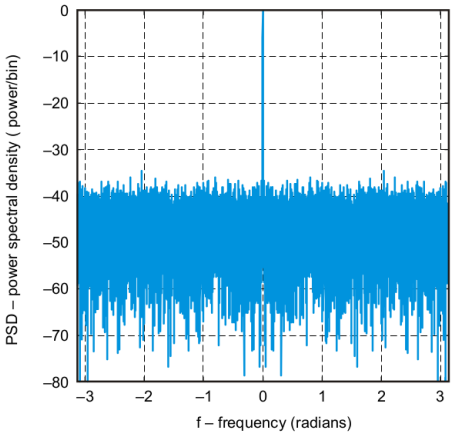


Figure 23-19. FFT of the Random Bytes

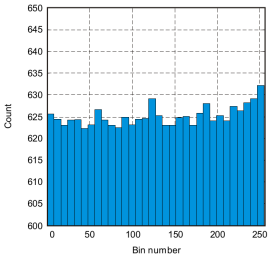


Figure 23-20. Histogram of 20 Million Bytes Generated With the RANDOM Instruction

Figure 3.1: RF core seeding result (Source: [102])

been able to find any public document describing the RF design of CC2538. However, we noticed that the same design has been applied to several products in TI's SimpleLink^{TM}series, some of which provided a better explanation of their RF core and RNG designs.

In the CC2430 user manual[98], we found a description of its RF core as in Figure 3.2 which explains that the input analogue signal to IF_ADC goes through the following components:

- Low Noise Amplifier (LNA) which amplifies the signal.
- Mixer which down converts the signal frequency. The Frequency Synthesiser is used as the local oscillator.
- Band pass filter which removes the out of band signals.

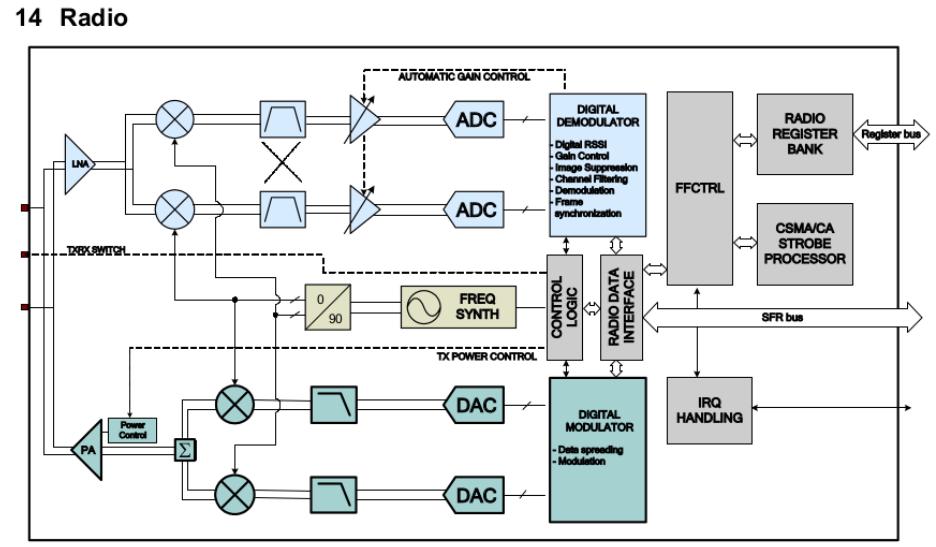


Figure 32: CC2430 Radio Module

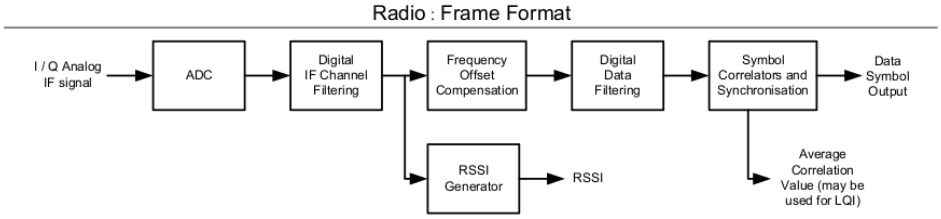


Figure 38: Demodulator Simplified Block Diagram

Figure 3.2: CC2430 RF Design (Source: [98]

- The Automatic Gain Control (AGC) circuit further adjusts the signal strength to the input level of ADC.

The CC2520 Data Sheet[103] explains the random bit is actually the LSB from ADC output:

> " *Single random bits from either the I or Q channel (configurable) can be output on GPIO pins at a rate of 8MHz. One can also select to xor the I and Q bits before they are output on a GPIO pin. These bits are taken from the least significant bit in the I and/or Q channel after the decimation filter in the demodulator.* " (Section 24, [103])

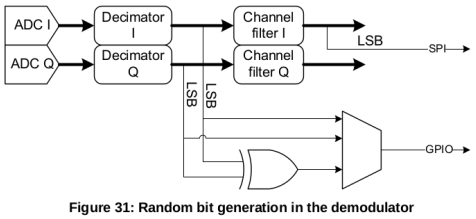


Figure 3.3: CC2520 RNG design (Source: [103])

A block diagram is also provided, as shown in Figure 3.3.

Interestingly, we noticed that CC2538, CC2520, CC253X and CC2540/41 reported exactly the identical randomness test result in their user manuals ([102] [103] [99]). We suspect the above evidence showed that CC2538 is very likely to have adopted the exactly same designs.

The information provided in all these documents explained the randomness of the seeding method. Denote the analogue RF signal by V_s and noise by N, the analogue input to the ADC, denote by V_{in}, can be represented as:

$$V_{in} = V_s + N \tag{3.1}$$

The noise N can be induced by multiple sources in practice, including noise produced by the signal source, environmental noise, and noise induced by the components in the device itself, etc.

The random bit b can therefore be represented as:

$$b = LSB(V_{in}) = LSB(V_s + N) \tag{3.2}$$

where $LSB() \in \{0, 1\}$ represents the operation of taking the LSB of A/D conversion output.

From Equation (3.2) we observe that any difference in V_{in} that is larger than the scale of ADC, i.e. the voltage represented by the LSB of ADC, could flip b. According to the CC2538 Datasheet[104], the receiver can be sensitive to signals down to $-97dBm$ (typical value with $T_A = 25°C$, $V_{DD} = 3V$ and

$f_C = 2440MHz$). On the other hand, the typical environmental noise in our test environment is about $-92dBm$ which is significantly higher than the receiver sensitivity. We consider this reading as a generic office use case and the result of randomness test as evidence to the sampling method.

3.4.2 Biasing the RF Signal in Practice

Equation (3.2) indicates that the random bit b is jointly determined by the signal V_s and noise N. Although an adversary can generate arbitrary signals, i.e. V_s is fully controlled by the adversary, it is clear that controlling N is difficult in practice. For instance, noises accumulated by different amplification stages are physically inevitable and intrinsic to the physical device. Hence it is not straightforward to fully control $V_{in} = V_s + N$ in practice.

An alternative attempt is to provide the RF with an 'illegal' input V_{in}. We considered two methods in our experiments: saturation and decimation. Saturation attempts to provide the RF with a strong signal that is above its acceptance level, whereas decimation attempts to suppress any RF signal to beneath the receiver sensitivity.

Ideally we expect these illegal inputs will trigger the ADC into a fault state which could potentially result into a predictable ADC output and thus biased b. But in practice, decimation does not seem practical for the same reason that noise induced by the circuit itself is physically inevitable. This made saturation the only viable option for us. We further note that the undisclosed circuit design of the device also posed a difficulty in our experiments. Without knowledge of the exact circuit design, we had to perform black box experiments.

The device we used is an OpenMote[105] powered by CC2538. The receiver has been configured to the default $2475MHz$ (channel 25 in 802.15.4) for Contiki CC2538 driver. We extended the length of each seed from RF to

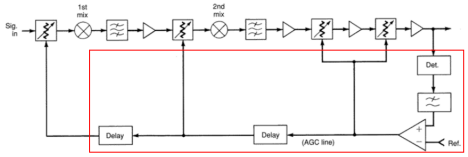


Figure 3.4: Example of receiver AGC (Source: [107])

128 bits in our experiments in coping to a potential PRNG design based on AES-128 as explained in Section 3.2, although we still consider the bits are generated bitwise when applying the NIST test suite.

Constant Strength Sine Signal

The first signal we attempted was a constant strength sine wave signal on the working frequency. According to CC2538 data sheet[104], the saturation signal strength for the RF receiver is $10dBm$. We have attempted to increase the input signal strength up to $13dBm$, which is roughly double of the saturation voltage, but no bias was observed. The seed sampled under this signal has passed all tests in the NIST test suite.

The result implies that the AGC circuit could have tuned down the signal which might have consequently prevented the seed from being biased. Although the exact AGC design for CC2538 is unclear, Figure 3.4 demonstrates an example of AGC design using 4 Voltage Controlled Amplifiers (VGAs). The output signal is parallelly connected to a detector to estimate the signal strength. The output of detector is compared to a reference voltage and their difference is provided as a feedback to adjust the control voltage of VGAs. To prevent signal distortion caused by abrupt voltage change, such as during a lightning storm, AGC designs normally adopt an attack time (or called settle time) before it adjusts the gain. [106] provides a detailed description of different AGC designs.

Variable Strength Sine Signal

Our second attempt was the variable strength signal which we intended to exploit the delay in AGC adjustments. To be more specific, the signal is a sine wave at the working frequency (so to pass the filters) which abruptly increases its strength to create a saturation, and then gradually decimates to tune down AGC.

The signal can be achieved by multiplying a strong sine wave signal at the carrier frequency to a controlling sawtooth signal. Denote the carrier signal V_C by:

$$V_C(t) = A\sin(\omega_C t) \tag{3.3}$$

where $\omega_C = 2475MHz$ is the carrier frequency in our case. The signal needs to be strong enough to transiently saturate the ADC when the RF is detecting environmental noise. Assuming an 8 bit ADC and environmental noise at $-92dBm$, V_C requires to be theoretically at least $-68dBm$.

We control the amplitude by a sawtooth signal V_S denote by:

$$V_S(t) = -(\omega_S t \mod 1) + 1 \tag{3.4}$$

where ω_S is the frequency of bursts in the signal which is much lower than ω_C and should be slightly lower than the frequency of AGC adjustments.

The desired signal $V(t)$ is thus their product:

$$V(t) = V_C(t)V_S(t) \tag{3.5}$$

The CC2538 user guide[102] describes a programmable register (namely RFCORE_XREG_AGCCTRL3) which allows the user to select the AGC settle timing between 15, 20, 25 and 30 periods with default 20. In the same document, the description of register RFCORE_XREG_RFC_OBS_CTRL0 stated that the random bit at both I and Q channels are updated at $8MHz$

which suggests that the receiver may have a sample rate of $16MHz$. Since the controlling signal should be slightly lower than the AGC adjustment frequency, this gives us an estimation of sawtooth signal near $0.8MHz$. Ideally, we expect $V(t)$ has the following properties:

1. Capable of passing the band pass filter.
2. Generate bursts that saturates the ADC; therefore bits sampled during those period results into predicted bits.

We implemented such signal using Gnu Radio Companion[108] with a HackRF One[109] directly connected to an OpenMote[105]. However, no significant bias was reported by the NIST test suite. The exact cause of failure is unknown due to lack of design documentation but one potential reason might be that the period of transient is too short to significantly affect the seed.

Strong constant signal

We then attempted a strong constant signal. The idea is to treat the whole circuit as a deterministic compression function that maps any V_{in} to $\{0, 1\}$. Under this assumption, the same V_{in} should always generate the same b, either 0 or 1. In order to achieve constant V_{in} in Equation (3.1), V_s needs to be significantly greater than N to suppress its impact in Equation (3.2), as any ADC would have only a limited resolution.

For experimental purpose, we have configured three programmable LNAs in the AGC to their maximum gain ($6 + 21 + 9 = 36(dB)$) and have disabled the attenuator in Anti Aliasing Filter (AAF, up to $9dB$). We consider these modifications can be compensated by a strong signal amplifier in practice. The signal source is implemented by Gnu Radio Companion (GRC)[108] with HackRF One[109], connected to the target OpenMote through a SMA cable for the best signal strength. Table 3.1 lists the configuration which

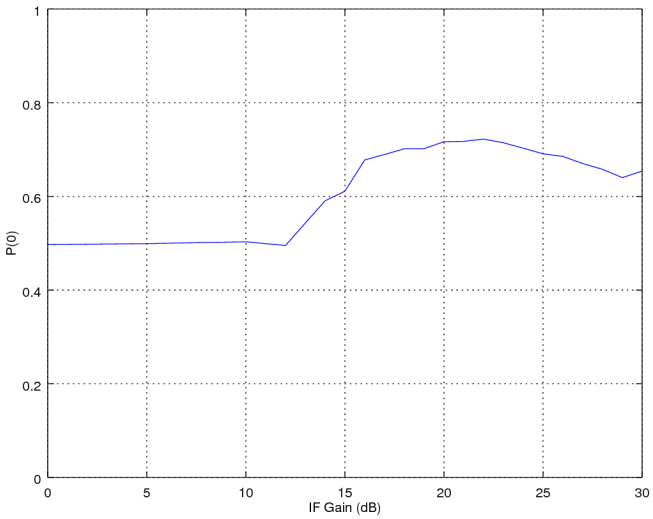
Table 3.1: GRC Signal Source Configuration

Sample Rate	8 MHz
Output Type	Complex
Waveform	Constant
Amplitude	0
Offset	1
IF Gain	[0, 30] dB
Output Voltage Amplitude	[0,176.0] mV

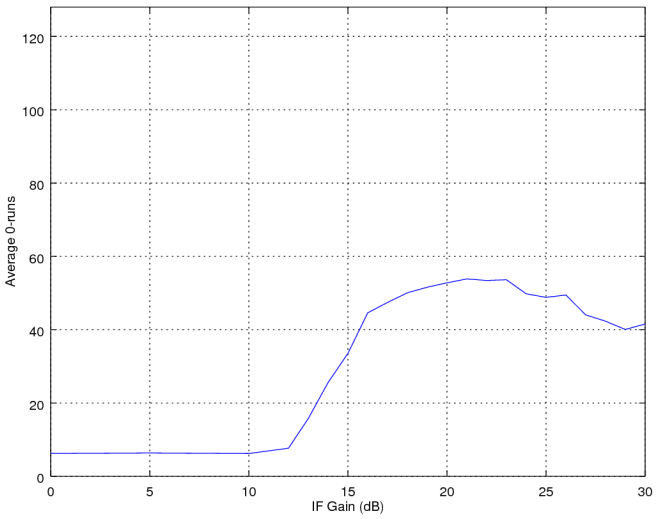
effectively generates a carrier wave on desired frequency.

Applying the signal, we observed abnormal 0-runs, i.e. consecutive 0 bits, appeared in the seeds as we increase IF gain to values above $10dB$. Figure 3.5a shows how $P(0)$ is biased and Figure 3.5b shows the average number of bits of longest 0-runs in each seed. We can see that the bias has reached its peak at IF_Gain = $22dB$ in both figures. At such gain 27.709% of the 128 bit seeds have longest 0-runs over 64 bits. It is not a surprise to see the sampled seeds have failed nearly all tests in the NIST test suite, indicating they have been strongly biased by our signal. We cannot determine the exact cause of bias decrease for IF Gain over 22dB due to lack of circuit design, but one potential cause might be the distortion under strong signal strength.

We also re-applied the signal to the same OpenMote we previously used in the strong sine wave signal experiments. A even stronger bias is observed as shown in Figure 3.6, with 17.820% of the seeds ended in 128 consecutive 0 bits. This may be caused by the strong sine wave signal in the previous experiments which permanently biased the device. We therefore restored its AGC configuration to default and re-ran the NIST test suite but the sampled seed passed all tests as before. We also tested using example applications provided by Contiki and found no malfunctioning on the device. The permanent bias does not seem to affect the device under normal operational status and can only be triggered by the constant signal.



(a) $P(0)$ to IF Gain



(b) Average longest 0-runs in each seed to IF Gain

Figure 3.5: Biased Seed on OpenMote. Signal source amplitude from 19.5mV (10dB), 76.0mV(22dB) to 176.0mV (30dB).

This leads to a very dangerous Trojan-like attack where devices could be primed in such a way that they remain functional under normal operating conditions, and eventually 'activated' via supplying the activation signal upon which they are unable to produce random numbers.

Again due to lack of design documentation we can not explain how exactly the bias is triggered by the signal. Nevertheless our experiments have demonstrated that sampling seed using RF noise could potentially be biased by jamming signal and hence potentially breach any cryptographic protocol relying on the randomness.

3.5 LFSR as PRNG

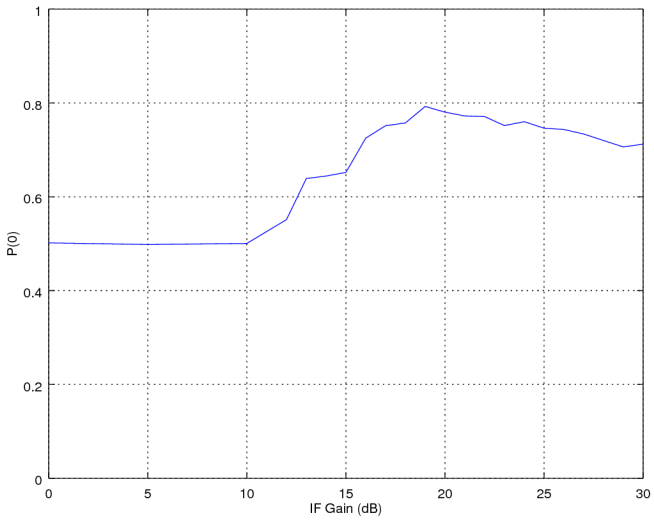
The instructions for the inbuilt PRNG is given as follows in CC2538 user guide:

> The random-number generator is a 16-bit linear-feedback shift register (LFSR) with polynomial
>
> $$x^{16} + x^{15} + x^2 + 1$$
>
> (that is, CRC16[110]). It uses different levels of unrolling depending on the operation it performs. The basic version (no unrolling) is shown in Figure 16-1[2].

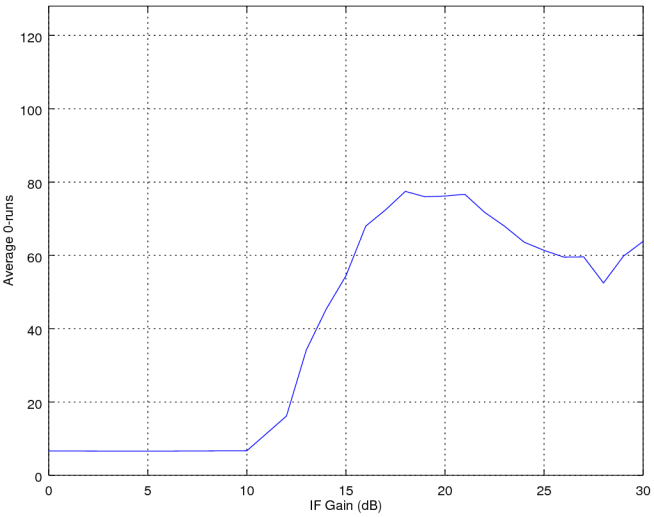
When used as a PRNG, the in_bit in Figure 3.7 is constantly 0. The Contiki driver calls the PRNG strictly following CC2538 user guide:

> " *Another way to update the LFSR is to set the RCTRL bits in the SOC_ADC_ADCCON1 register to 01. This clocks the LFSR once (13x unrolling), and the RCTRL bits in the*

[2]Provided in Figure 3.7



(a) $P(0)$ to IF Gain



(b) Average longest 0-runs in each seed to IF Gain

Figure 3.6: Biased Seed on OpenMote used in previous experiments. Signal source amplitude from 19.5mV (10dB), 76.0mV(22dB) to 176.0mV (30dB).

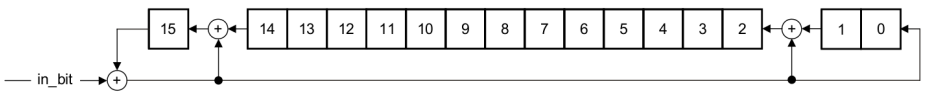


Figure 16-1. Basic Structure of the RNG

Figure 3.7: CRC16 LFSR (Source: [102])

SOC_ADC_ADCCON1 register automatically clear when the operation completes. ” (Section 16.2.1, [102])

In other words, the LFSR is updated by performing 13 CRC16 operations in Figure 3.7 upon each RNG call. Since the CRC16 is deterministic and the register has only 16 bits, the PRNG can be modelled as a Deterministic Finite Automaton (DFA) which made its output easily predictable.

Formally, because there are only 16 bits in the LFSR, we can denote the universal set of its possible values (or called states) $\mathbb{S}$ as:

$$\mathbb{S} = \{S_i | S_i \in \{0,1\}^{16}\} \tag{3.6}$$

Equation (3.6) implies that the LFSR can have no more than $|\mathbb{S}| = 2^{16} = 65536$ states.

We denote the LFSR update operation, as:

$$F : \mathbb{S} \to \mathbb{S} \tag{3.7}$$

where F is 13 times of CRC16 operation on the current state according to the manual.

Denote the 16 bits random seed as S^*. The PRNG output can be formalised as:

$$\begin{aligned} S_0 &= S^* \\ S_{i+1} &= F(S_i) \end{aligned} \tag{3.8}$$

Since $\mathbb{S}$ is finite and F is deterministic, the random number stream is cyclic. The longest non-repetitive PRNG output sequence under seed S^* can be represented as:

$$R_{S^*} = (F^0(S^*), F^1(S^*), ..., F^{n-2}(S^*), F^{n-1}(S^*)) \tag{3.9}$$

where $S^* = F^0(S^*) = F^n(S^*)$. Each call to the PRNG effectively returns

the first element in the sequence and updates it by one cyclic left rotation. Since the elements within R_{S^*} are non-repetitive, we have $n \leq |\mathbb{S}|$ for any R_{S^*}, i.e. the cycle of PRNG output is at most 65536 calls.

For a re-sampled seed $S^{*'}$ inside R_{S^*}, i.e. $S^{*'} = F^k(S^*)$ where $k \in \mathbb{Z}_n$, the corresponding sequence $R_{S^{*'}}$ is:

$$\begin{aligned} R_{S^{*'}} =& (F^k(S^*), F^{k+1}(S^*), ..., F^{n-1}(S^*), F^0(S^*), F^1(S^*), ..., \\ & F^{k-2}(S^*), F^{k-1}(S^*)) \end{aligned} \tag{3.10}$$

Observing Equation (3.9) and Equation (3.10), we can see R_{S^*} is indeed $R_{S^{*'}}$ left rotated by $(n-k)$ times. This is equivalent to say that S^* generates identical output as $S^{*'}$ with $(n-k)$ preceding calls. As a result, assume consecutive PRNG calls on R_{S^*} returns a sequence of:

$$(S_i, S_{i+1}, ..., S_j)$$

then the same sequence will eventually be replicated by calls on $R_{S^{*'}}$.

This property also indicates that any seed not in R_{S^*} generates a completely different sequence. By enumerating $\mathbb{S}$, we found there exists only four non-overlapping sequence for this CRC16 constructed PRNG, which are:

- R_{0x0001} with $n = 32767$.
- R_{0x0003} with $n = 32767$.
- R_{0x0000} with $n = 1$. $(F(0x0000) = 0x0000)$
- R_{0x8003} with $n = 1$. $(F(0x8003) = 0x8003)$

Notice that R_{0x0000} and R_{0x8003} are excluded in the driver due to their monadic output according to the manual[102]. The enumeration can be done on a CC2538 in less than a minute for such a small space of 65536.

3.6 Contiki RNG Driver Issues for CC2538

Several bugs in Contiki CC2538 RNG driver were also found and reported in our study. That includes:

1. Reading out of LFSR without ready check.
2. Lack of validity check when reading random seed bits from the RF module.
3. A bug that drops the Most Significant Bit (MSB) and leaves the Least Significant Bit (LSB) to be constantly zero in seeding the LFSR.
4. The user guide suggests only to use the lower byte (8 bits) as the random number output but the driver actually used all 16 bits in the LFSR.

Note that these bugs may negatively impact the quality of RNG. For example, reading LFSR without a ready check returns a repetitive output before update, and reading RF without validation results into a constant erroneous bit. We have patched the above mentioned bugs before conducting our experiments in this chapter.

3.7 Security Impact: Breaking DTLS

Similar to the ECC key breach of Smart Energy Profile as pointed out by [84], the above mentioned PRNG on CC2538 can be exploited to break several protocols in Datagram TLS (DTLS) which is a widely supported standard in many IoT applications.

Contiki supports DTLS via an implementation called tinydtls[111]. Two cipher suites, namely Pre-Shared Key[112] (PSK) and ECDHE_ECDSA[113] are implemented by the latest available version (0.8.2) and the only supported curve is secp256r1[114]. In this chapter we only discuss ECDHE_ECDSA.

1: **procedure** EccKeyGen(Domain Parameter $T = (p, a, b, G, n, h)$ as specified by [114])
2: Randomly select $d \in [1, n-1]$.
3: Compute $Q = dG$.
4: **return** (Q, d), where Q is the public key and d the secret key.
5: **end procedure**

Figure 3.8: ECC Key Generation

Unfortunately, tinydtls does not implement its own RNG; instead it loops the Contiki API (random_rand()) which is then implemented by the CC2538 built-in PRNG (see tinydlts/dtls_prng.h) as we described in Section 3.5. As a result, the generated random numbers are from a very restricted set that is too small for any cryptographic use. This renders already any key generation vulnerable.

Figure 3.8 describes the ECC Key Generation. The RNG is involved in the selection of d. Since T is public, an adversary can pre-compute all possible public keys by enumerating all secret keys beginning in each position of R_{0x0001} and R_{0x0003}. Upon observing a public key, the adversary can immediately look up its corresponding secret key in the pre-computed look up table. Since the look up table has only 65534 entries, the pre-computation took less than 5 minutes on a laptop powered by i7-2620M. Besides rendering key generation trivially insecure, one can further apply two trivial attacks during a DTLS handshake. As before, these attacks work easily because of the poor randomness and the fact that popular EC schemes use public, standardised base points.

ECDSA ECDSA[115] is an authentication scheme that allows a party to authenticate a message. In DTLS, it is used to sign the server parameters (details in [116]) during the handshake to provide server side authenticity. As described in Figure 3.9, ECDSA requires a secret random number k to generate a point on the curve R via scalar multiplication of a base point.

1: **procedure** EcdsaSign(Domain Parameter $T = (p, a, b, G, n, h)$, server key pair (Q, d) and a message) to be signed m.
2: Randomly select $k \in [1, n-1]]$.
3: Compute $kG = (x_1, y_1)$ and let $r = x_1 \mod n$.
4: Compute $e =$ SHA-1(m).
5: Compute $s = k^{-1}(e + dr) \mod n$.
6: **return** (m, r, s) as the message-signature pair.
7: **end procedure**

Figure 3.9: ECDSA Signing

The x-coordinate r of this point becomes part of the signature. Hence it can be observed by the adversary, who can recover k by searching r in the look up table of pre-computed points. He can then recover the secret signing key d by computing:

$$\begin{aligned} e &= SHA-1(m) \\ d &= r^{-1}(sk - e) \mod n \end{aligned} \tag{3.11}$$

ECDHE ECDHE[113] is a key exchange protocol that allows two party to derive a shared secret. In DTLS, ECDHE is performed at the end of DTLS handshake to derive a shared secret, which is then used to derive the symmetric session key for application data encryption.

Figure 3.10 provides a brief description of ECDHE. The adversary can recover r_A and r_B by observing Q_A and Q_B that is being sent in the packets; hence computes K to derive the symmetric key.

Because G is public, it is again possible to derive r_A and r_B by looking up Q_A and Q_B in the pre-computed table. Once these quantities are known to the adversaries, they can also compute Q_{AB} and hence the session key.

We have tested the above attacks by sniffing two CC2538 nodes performing handshake using the example code provided by tinydtls. The secret keys have been successfully recovered using the look up table we generated.

1: **procedure** EcdheKeyExchange(Domain Parameter $T = (p, a, b, G, n, h)$. Party A's key pair (Q_A, d_A) and party B's key pair (Q_B, d_B))
2: A randomly picks $r_A \in [0, n-1]$.
3: B randomly picks $r_B \in [0, n-1]$.
4: A computes $Q_A = r_A G$ and sends Q_A to B.
5: B computes $Q_B = r_B G$ and sends Q_B to A.
6: Both A and B computes $Q_{AB} = r_A r_B G = r_A Q_B = r_B Q_A$.
7: **return** Both A and B returns $K = Hash(Q_{AB})$ as the shared secret.
8: **end procedure**

Figure 3.10: ECDHE Key Exchange

```
static unsigned long seed = 1;

int
rand (void)
{
  return do_rand (&rand);
}
```

Figure 3.11: rand() implementations in stdlib

3.8 Other RNG implementations in Contiki

Investigating (P)RNG implementations in other platforms supported by Contiki, we realised that most of them do not have dedicated PRNG implementations and by default wrap rand() in stdlib as their PRNG. We traced some of the open sourced stdlib implementations. For the majority of the libraries, i.e. stdlib for ARM[117], AVR[118] and MSP430[119], the rand() implementation can be abstracted as Figure 3.11. The type of variable *seed* may vary on different platforms. The *do_rand()* function outputs a congruent of linear transformation of *seed* and updates *seed* by the output.

It is clear that such design would also yield into a predictable random number stream with cycle no longer than the range of *seed*, as the same *seed* returns the same output. On the above platforms, the cycles are no longer than 2^{32}, 2^{16} and 2^{16} calls respectively.

3.9 Patching the PRNG

The PRNG issues can be efficiently patched by using more sophisticated PRNG implementation for cryptographic applications, such as switch to constructions based on approved hash functions and block ciphers as recommended by [96]. Specifically for CC2538, SHA-256 and AES have hardware coprocessor support and therefore can be considered candidates for implementing cryptographically secure PRNG according to [96].

Nevertheless, any secure PRNG still needs to be seeded by a secure TRNG. It is unfortunate that the approach taken for the CC2538, which is based on reusing an existing radio module with no protection against active adversaries, fails to meet even basic requirements as we demonstrated inSection 3.4.2. Therefore despite all the cryptographic co-processors provided, we would not recommend the device, i.e. CC2538, to be used for critical security applications.

3.10 Summary

In this chapter we reviewed the provision for cryptographic random numbers on the CC2538 and related devices. First, we discussed the poor choice of using a 16 bit LFSR as PRNG and demonstrated how this design flaw can be exploited to break DTLS running on these devices. We also found that the provision for randomness within the popular Contiki software and DTLS implementation tinydtls is inadequate. Any open source efforts, or indeed also any products, that built on them should review their instantiation of random numbers carefully.

We also investigated how to tamper with the RF source and showed in practice how to configure signals to that end. We reverse engineered the design of the path that produces random bits from the RF module, and developed some attacks that can bias the random bits in practice. This

shows that even if the poor PRNG was replaced with a sound one, the source for the seed of any PRNG on the CC2538 is vulnerable to practical attacks. However, the signal strength required for the attack might not be achievable unless the adversary have direct physical access to the device.

We believe that the same design choices have also been adopted by many other products in the CC series including CC2420[120], CC2430[98], CC2520[103] and CC253X, CC2540/41 series[99]; thus all these products suffer from the same problems. Only the latest CC26XX/CC13XX[121] series has abandoned this design and implemented a dedicated RNG suitable for cryptographic purposes. We recommend to update the legacy devices for security sensitive application.

Chapter 4

DPA on ARX Ciphers

4.1 Introduction

The IoT boom in the recent years has drawn a great interest into the research of lightweight cryptography. Among different approaches being adopted to design lightweight ciphers, the ARX paradigm is particularly interesting to the side channel research community, partly due to its seemingly "inherent resilience" against side channel attacks. Remarkably, the work done by Biryukov et al. [70] studied various instructions used to construct lightweight ciphers and concluded:

> " *The software implementations of the three ARX designs we considered are characterized by a certain level of "intrinsic" resilience against CPA. ... These features make ARX constructions excellent candidates for the implementation of lightweight block ciphers for the IoT.* " (Conclusion, [70])

Such "intrinsic" resilience is greatly appealing to IoT designers as many resource constrained devices cannot afford the overheads brought by countermeasures against side channel attacks.

This chapter extends the result of [70] into a practical scenario where

we thoroughly studied the above claimed side channel resilience of ARX ciphers against various DPA attacks. It is based on my unpublished work co-authored with Elisabeth Oswald and Srinivas Venkatesh. As the main author I was responsible for conducting all experiments in this chapter as well as proposing the chosen message DPA strategy described in Section 4.5. My college Srinivas Venkatesh has contributed in part to the mathematical formalisations in this work.

This chapter begins by presenting some practical concerns when targeting the only non-linear operation in ARX-Boxes, i.e., modular addition, in Section 4.3, followed by experiments in Section 4.4 and a novel chosen message DPA strategy in Section 4.5. We then demonstrate the practical results when targeting the linear operations in ARX-Boxes, i.e., XOR and rotation, in Section 4.6 and conclude the chapter in Section 4.7.

4.2 Preliminaries

4.2.1 Notations

In Chapter 4 we frequently require the notation of a specific bit in the binary presentation of a variable x. The notation $[x]$ indicates the binary representation of an integer x: $x = [x]_{n-1}[x]_{n-2}...[x]_1[x]_0 = \sum_{i=0}^{n-1} 2^i[x]_i$, hence $[x]_i$ denotes the i-th bit of $[x]$. The notation $[x][y]$ implies the concatenation of two bit strings $[x]$ and $[y]$. The notation $[x]^k$ denotes k times repetition of $[x]$. Specifically, $[*]^k$ denotes an arbitrary k-bit string.

We are mostly concerned with modular addition, logical rotation (abbreviate as rotation) and XOR operations over the field $\mathbb{Z}_{2^n}$. We denote them as:

- $x \boxplus y$: $(x + y) \mod 2^n$
- $x >> y$: Right rotate x by y bits.

- $x << y$: Left rotate x by y bits which equates to $x >> (n - y)$.
- $x \oplus y$: Exclusive-or of x and y.

We often require to be able to change the value of a bit to its (one's) complement (i.e. we flip a bit, but leave all other bits unchanged). For this purpose we define the flip function $\mathcal{F}_i(x)$ which returns x with the i-th bit flipped:

$$\mathcal{F}_i(x) = x \oplus 2^i$$

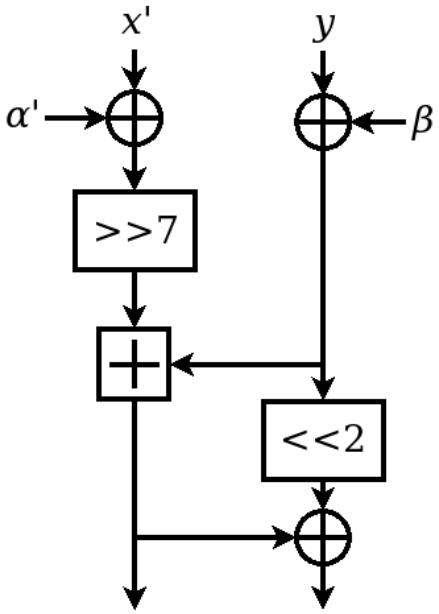
4.2.2 SPARX Round Function and Generalised ARX-Box

SPARX [66] is a recently published ARX cipher that bases its round function on SPECKEY [66]. The SPARX round function takes a 32-bit input and divides this in two equal halves. The two halves are firstly XOR-ed with some key material, and then a modular addition $\boxplus$ takes place (one of the inputs to this addition is also rotated), which is then the left half, as depicted in Figure 4.1a.

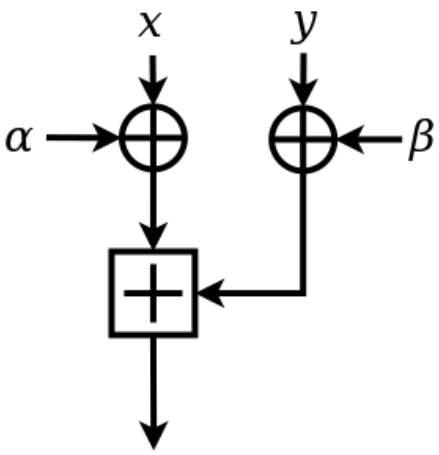
Whilst the rotations are important with respect to the cipher's resilience against classical cryptanalysis, these rotations, which are based on known constants, do not add to the cipher's resilience against side channel attacks. Typical DPA-style attacks target these operations in the first (or last) cipher round, and thus because the rotations are based on known values, they can be easily incorporated into any side-channel distinguisher at no extra cost. Thus in our study, we consider a simplified ARX structure that omits the rotation in the SPARX round function which can be represented as:

$$s = S(x, y) = (x \oplus \alpha) \boxplus (y \oplus \beta) \tag{4.1}$$

where (x, y) are the inputs, (α, β) the secret keys, and all variables are in the field $\mathbb{Z}_{2^n}$. Figure 4.1b presents the circuit that is equivalent to Equation (4.1).



(a) SPARX round function



(b) Generalised ARX-Box $S(x, y)$

Note that most ARX-Boxes can be reduced to the form of Equation (4.1) by substituting the inputs. For example, the upper half of SPARX ARX-Box (up to the modular addition) is equivalent to setting:

$$\begin{cases} x = x' << 7 \\ \alpha = \alpha' << 7 \end{cases} \tag{4.2}$$

The lower half of the right branch (from $<< 2$) in Figure 4.1a can also be merged into the second round in a similar manner.

Take SKEIN [64] (Figure 4.2) for another example where the plaintext is directly added to the key. The initial subkey addition can be generalised into Equation (4.1) as:

$$s = S(x, 0) = (x \oplus 0) \boxplus (0 \oplus \beta)$$

by setting the plaintext to x and Subkey 0 to β, and letting $\alpha = y = 0$.

Our experiments are focused on the SPARX cipher for two reasons. First

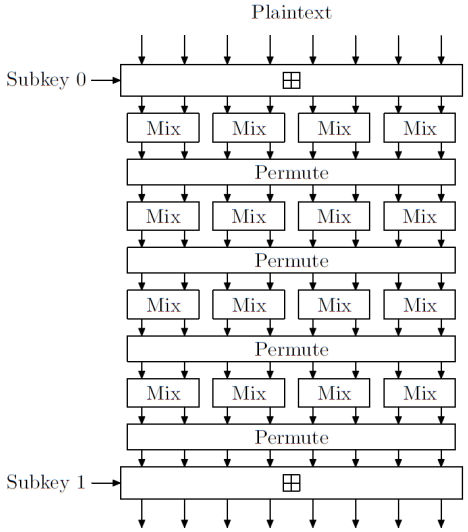


Figure 4.2: Skein hash function (Source: [122])

of all, it is the first ARX construction with a provable crypto-analytical bound [66]. Secondly, it has a public available reference implementation in C which can be easily ported to various platforms. However, the analytical aspect of our work is generally applicable to arbitrary ARX ciphers through reductions as explained above.

To our knowledge, only minimum work has been done on the side channel aspects of ARX-ciphers with practical results. The closest was published in [71] where the authors proposed to improve the straightforward correlation attack on a ARM M3 processor against the ARX-Box of SKEIN through testing pairs of correlation peaks. However, this approach requires one of the adders to be known to the adversary and thus not compatible with our generalised ARX-Box. [123] reported another relevant result where the authors successfully recovered the key of the SPECK cipher [65] in a straightforward correlation attack targeting the writing back of the output of key XOR. We consider the result of [123] as a supporting evidence of [70] where the later reported that memory instructions, such as read and

write, produce leakages in a much higher magnitude comparing to arithmetic instructions. Nevertheless, as will be shown later in Section 4.6, the same target function could still be vulnerable against DPA attacks in the naive reference implementation of the SPARX ARX-Box, even though the operands are manipulated only in the registers.

4.3 Observations on Modular Addition as a DPA Target

The effectiveness of a DPA style attack is commonly evaluated by the number of traces required to recover the key. It is generally perceived that completely linear targets such as the XOR and rotation operations are difficult to attack with DPA: attacks on such targets require many more traces than attacks on highly non-linear target functions, and even with very large numbers of leakages there remains some keys that cannot be distinguished from each other [124]. This statement is further supported by [70] where the authors reported the difficulties of independent correlation attacks using HW predictions against the XOR and rotation instructions.

Considering the fact that modular addition is the only non-linear operation in an ARX-Box, it is naturally the primary target for DPA attacks as explained by [70]. However, we realised that straightforward DPA methods might not be as effective as they generally are against S-Boxes; for instance, the case of AES described in [30]. In this section, we address some issues that should be concerned when targeting modular addition in a DPA attack.

4.3.1 Equivalent Keys

Our study found that one cannot achieve a first order success exploiting only the leakage of modular addition due to the existence of two types of equivalent keys, i.e., keys cannot be distinguished from each other.

Proposition 1. *It holds that:*

$$(x \oplus \alpha) \boxplus (y \oplus \beta) = (x \oplus \mathcal{F}_{n-1}(\alpha)) \boxplus (y \oplus \mathcal{F}_{n-1}(\beta)) \tag{4.3}$$

for arbitrary x and y.

Proof. Let

$$\begin{cases} s = (x \oplus \alpha) \boxplus (y \oplus \beta) \\ s' = (x \oplus \mathcal{F}_{n-1}(\alpha)) \boxplus (y \oplus \mathcal{F}_{n-1}(\beta)) \end{cases} \tag{4.4}$$

Specifically, the MSBs of s^* and s' can be represented as:

$$\begin{cases} [s]_{n-1} = ([x]_{n-1} \oplus [\alpha]_{n-1}) \oplus ([y]_{n-1} \oplus [\beta]_{n-1}) \oplus c_{n-1} \\ [s']_{n-1} = ([x]_{n-1} \oplus [\mathcal{F}_{n-1}(\alpha)]_{n-1}) \oplus ([y]_{n-1} \oplus [\mathcal{F}_{n-1}(\beta)]_{n-1}) \oplus c'_{n-1} \end{cases} \tag{4.5}$$

where c_{n-1} and c'_{n-1} are the carry bits of s and s' generated at their previous $n-1$ bits. Since (α, β) and $(\mathcal{F}_{n-1}(\alpha), \mathcal{F}_{n-1}(\beta'))$ only differ at their MSBs, s and s' have identical lower $n-1$ bits and $c_{n-1} = c'_{n-1}$. Note that

$$\begin{cases} [\mathcal{F}_{n-1}(\alpha)]_{n-1} = \widetilde{[\alpha]_{n-1}} \\ [\mathcal{F}_{n-1}(\beta)]_{n-1} = \widetilde{[\beta]_{n-1}} \end{cases} \tag{4.6}$$

Therefore by XOR Equation (4.4), we have

$$\begin{aligned} s \oplus s' &= [s]_{n-1} \oplus [s']_{n-1} \\ &= \widetilde{[\alpha]_{n-1}} \oplus [\alpha]_{n-1} \oplus \widetilde{[\beta]_{n-1}} \oplus [\beta]_{n-1} = 0 \end{aligned} \tag{4.7}$$

Hence

$$s = s' \tag{4.8}$$

□

Proposition 1 implies that $(\mathcal{F}_{n-1}(\alpha), \mathcal{F}_{n-1}(\beta))$ will always result in the same modular sum and thus the same as leakage as the correct key (α, β);

therefore they cannot be distinguished from each other.

"Generic" distinguishers, which we explain in more detail in Chapter 5, cannot distinguish keys when their corresponding hypothetical intermediates are permutations of each other [125]. Proposition 2 shows that additional equivalent keys should be taken into account for this class of distinguishers.

Proposition 2. *Let* (x_i, y_i) *for* $1 \leq i \leq t$ *be* t *pairs of input to an ARX-Box. For the following equation:*

$$(x_1 \oplus \alpha) \boxplus (y_1 \oplus \beta) = (x_2 \oplus \alpha) \boxplus (y_2 \oplus \beta) = \ldots = (x_t \oplus \alpha) \boxplus (y_t \oplus \beta) \quad (4.9)$$

there exist at least 8 pairs of (α, β) *that satisfies.*

To prove Proposition 2, we first prove Proposition 3 and Proposition 4.

Proposition 3. *Let* (α, β) *be a solution to Equation* (4.9)*, then there exists at least* 4 *solutions to Equation* (4.9) *given as*

$$\{(\alpha, \beta), (\alpha', \beta), (\alpha, \beta'), (\alpha', \beta')\}$$

where

$$\begin{cases} \alpha' = 2^{n-1} \oplus \alpha \\ \beta' = 2^{n-1} \oplus \beta \end{cases}$$

Proof. Flipping the MSB of α is arithmetically equivalent to adding $\pm 2^{n-1}$. Therefore denote:

$$\alpha \oplus x = \alpha' \oplus x \oplus 2^{n-1} = \alpha' \oplus x + \Delta$$

where $\Delta \in \pm 2^{n-1}$. It follows that:

$$\begin{aligned} S_{\alpha,\beta}(x, y) &= (x \oplus \alpha) \boxplus (y \oplus \beta) \\ &= (x \oplus \alpha' + \Delta) \boxplus (y \oplus \beta) \\ &= S_{\alpha',\beta}(x, y) + \Delta = S_{\alpha',\beta}(x, y) \oplus 2^{n-1} \end{aligned} \quad (4.10)$$

for arbitrary (x, y).

Since (α, β) is a solution to Equation (4.9), for any $i \neq j$, substitute Equation (4.10):

$$\begin{aligned} S_{\alpha,\beta}(x_i, y_i) &= S_{\alpha,\beta}(x_j, y_j) \\ S_{\alpha',\beta}(x_i, y_i) \oplus 2^{n-1} &= S_{\alpha',\beta}(x_j, y_j) \oplus 2^{n-1} \\ S_{\alpha',\beta}(x_i, y_i) &= S_{\alpha',\beta}(x_j, y_j) \end{aligned}$$

Thus (α', β) is also a solution to Equation (4.9).

Due to the symmetry of α and β, it can be easily proved that:

$$S_{\alpha,\beta'}(x_i, y_i) = S_{\alpha,\beta'}(x_j, y_j) \tag{4.11}$$

for any $i \neq j$.

Applying Equation (4.10) twice to both (α, β), we have:

$$S_{\alpha',\beta'}(x_i, y_i) = S_{\alpha',\beta'}(x_j, y_j) \tag{4.12}$$

for any $i \neq j$.

Therefore if (α, β) is a solution to Equation (4.9), then (α', β), (α, β') and (α', β') are also solutions to Equation (4.9). □

Note that Proposition 1 is indeed a special case of Proposition 3 under the condition of a specific value of $([\alpha]_{n-1} \oplus [\beta]_{n-1})$.

Proposition 4. *If (α, β) is a solution to Equation (4.9), then $(\tilde{\alpha}, \tilde{\beta})$ is also a solution to Equation (4.9), where*

$$\begin{cases} \tilde{\alpha} = (2^n - 1) \oplus \alpha \\ \tilde{\beta} = (2^n - 1) \oplus \beta \end{cases} \tag{4.13}$$

Proof. Since $(\tilde{\alpha}, \tilde{\beta})$ are complements of (α, β); therefore

$$(\tilde{\alpha} \oplus x, \tilde{\beta} \oplus y)$$

also complements

$$(\alpha \oplus x, \beta \oplus y)$$

for arbitrary (x, y).

Hence we have:

$$\begin{cases} x \oplus \alpha = 2^n - 1 - (x \oplus \tilde{\alpha}) \\ y \oplus \beta = 2^n - 1 - (y \oplus \tilde{\beta}) \end{cases} \tag{4.14}$$

Therefore

$$\begin{aligned} S_{\alpha,\beta}(x, y) &= (x \oplus \alpha) \boxplus (y \oplus \beta) \\ &= (2^n - 1 - (x \oplus \tilde{\alpha})) \boxplus (2^n - 1 - (y \oplus \tilde{\beta})) \\ &= 2^{n+1} - 2 - S_{\tilde{\alpha},\tilde{\beta}}(x, y) \end{aligned} \tag{4.15}$$

for arbitrary (x, y).

Since (α, β) is a solution to Equation (4.9), for any $i \neq j$, we have:

$$\begin{aligned} S_{\alpha,\beta}(x_i, y_i) &= S_{\alpha,\beta}(x_j, y_j) \\ 2^{n+1} - 2 - S_{\tilde{\alpha},\tilde{\beta}}(x_i, y_i) &= 2^{n+1} - 2 - S_{\tilde{\alpha},\tilde{\beta}}(x_j, y_j) \\ S_{\tilde{\alpha},\tilde{\beta}}(x_i, y_i) &= S_{\tilde{\alpha},\tilde{\beta}}(x_j, y_j) \end{aligned} \tag{4.16}$$

which implies $(\tilde{\alpha}, \tilde{\beta})$ is also a solution to Equation (4.9). □

Given the correct key (α, β), Proposition 3 derives 2 pairs of equivalent keys. Each of these equivalent keys derives another 4 pairs of equivalent keys and thus Proposition 2 is proven. Note that, for generic distinguishers, equivalent keys in Proposition 2 result in partitions that are permutations of each other and as such are indistinguishable.

4.3.2 Ineffective Single Bit DPA

Single bit DPA is ineffective against modular addition. Observe that the i-bit of the modular sum $[s]_i$ can be represented as:

$$[s]_i = ([x]_i \oplus [\alpha]_i) \boxplus ([y]_i \oplus [\beta]_i) + c_i = [x]_i \oplus [\alpha]_i \oplus [y]_i \oplus [\beta]_i \oplus c_i \quad (4.17)$$

where c_i denotes the carry bit from adding the previous bits and specifically $c_0 = 0$. Equation (4.17) implies that single bit key guesses $([\alpha]_i, [\beta]_i)$ and $(\widetilde{[\alpha]_i}, \widetilde{[\beta]_i})$ are equivalent and thus cannot be distinguished from each other in a DPA attack. Consequently, applying single bit DPA on each bit of s only recovers $\alpha \oplus \beta$ and reduces the key space from 2^{2n} to 2^n, which might still be costly in practice.

4.3.3 The Enumeration Space and Divide-and-conquer

When single bit DPA is not viable as we explained in Section 4.3.2, the adversary will have to perform multi bit DPA attacks. Note that determining s requires the adversary to simultaneously enumerate both α and β. For n-bit operands, this implies the key enumeration space is 2^{2n} which quickly becomes impractical as n increases. Take SPARX [66] for example: its 16 bit operands implies 2^{32} keys need to be enumerated which could be costly in practice.

Alternatively, a general solution to reduce the key enumeration space is the divide-and-conquer strategy that recovers (α, β) chunk-wise. Denote by s_c, α_c, β_c, x_c, y_c the l-bits chunks starting from the i-th bit of s, α, β, x, y respectively and their corresponding previous bits as s_p, α_p, β_p, x_p and y_p. Observe that:

$$s_c = (x_c \oplus \alpha_c) \boxplus (y_c \oplus \beta_c) \boxplus c_i \quad (4.18)$$

where c_i, the carry bit from adding the previous bits, can be expressed as:

$$c_i = \begin{cases} 0 \text{ if } (x_p \oplus \alpha_p) + (y_p \oplus \beta_p) < 2^i \\ 1 \text{ otherwise} \end{cases} \tag{4.19}$$

and specifically $c_0 = 0$.

Equation (4.18) indeed suggests a naive approach of divide-and-conquer by recursively recovering the key bits from LSB to MSB. However, it can be easily proved that Proposition 1 and Proposition 3 also apply to each chunk. Therefore, in such a straightforward divide-and-conquer approach, all equivalent keys recovered in each chunk will have to be carried out into the next chunk, resulting in a key space which exponentially explodes with the number of chunks.

Reviewing Proposition 1 and Proposition 3, we notice that the equivalent keys only differ at their MSBs; thus, attacking each chunk indeed recovers the unique lower $l - 1$ bits of (α_c, β_c). Exploiting this feature, by dropping the MSBs of (α_c, β_c) and overlapping them with the LSBs of the next chunk, one can avoid equivalent keys and uniquely recover the lower order of $l - i$ bits (α_c, β_c), except for the last chunk where there is no next chunk to be overlapped. Applying this overlapping method, we managed to reduce the resulting key space in a divide-and-conquer attack to 2 in general and 4 for generic distinguishers.

4.3.4 Weak Non-linearity

Despite being the only non-linear component in ARX ciphers, modular addition is known to be only weakly non-linear, the impact of which has been previously reported by Lemke et al. [126].

In general, consider a simplified modular addition

$$x \boxplus k = z$$

Flipping the higher bits of k leaves the lower order bits of the sum z unchanged. For instance, in an extreme case, flipping the MSB of k (or x) is a linear operation, because:

$$k' = k \oplus 2^{n-1} \implies (x \boxplus k) \oplus (x \boxplus k') = 2^{n-1}$$

that is, only the MSB of z is flipped in response.

In terms of DPA attacks, this implies that, given a set of plaintexts and a number of key guesses including k' as just described, the intermediate values related to the sum z are identical in most bits. Consequently, for typical linear leakage models, their resulting hypothetical leakages will be very close, making it hard to distinguish between the correct key guess and "similar" incorrect key guesses.

4.4 Correlation Attacks Against Modular Addition

With the observations of Section 4.3 in mind, we implemented a classic correlation attack using Pearson's correlation with HW predictions against modular addition. Remark that the optimisation proposed in [71] is not an option in our generalised ARX-Box as it requires the knowledge of one of the adders, while both are unknown in our case. Our implementation targets the lowest 4 bits for a practical key enumeration space (2^8); thus 2 pairs of equivalent keys are expected at the end of this attack. The 4 bit attack can be extended to all 16 bits of (α, β) as we explained in Section 4.3.3.

We first applied the attack on traces simulated as the HW of s with additive Gaussian noise at an SNR setting of 2^1. The attack successfully recovered the key using 500 traces as we show in Figure 4.3a. Our observations in Section 4.3 are also reflected in Figure 4.3a in the sense that:

- Figure 4.3a is symmetric, with each key having exactly one equivalent counterpart as expected in Section 4.3.1. The correct key $(0x0, 0x3)$ and

its equivalents $(0x8, 0xB)$ are together showing the highest (absolute value of) correlations of 0.406.

- Many of the incorrect key candidates are also showing significant correlations in Figure 4.3a, which confirms the prediction of Section 4.3.4.

Decreasing the SNR to 2^{-3} caused the attack fail with the same number of traces, as showed in Figure 4.3b. Even though the correct key(s) still showed a significant correlation of 0.179, an incorrect key candidate showed an even higher correlation, which could be explained by Section 4.3.4.

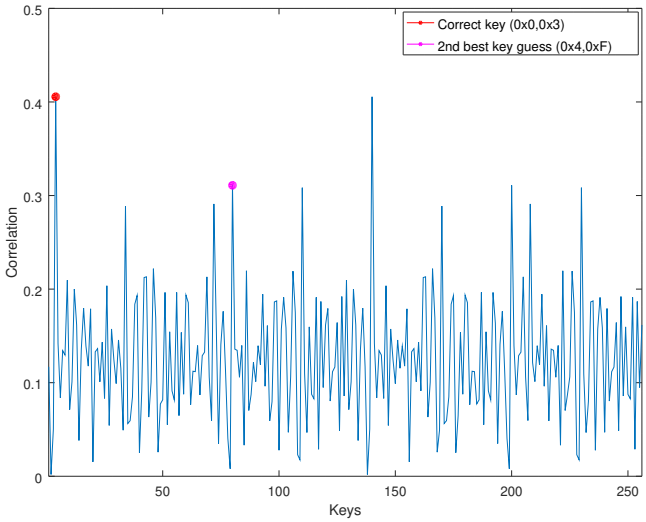
We carried the experiments over real traces collected on an ARM Cortex-M0 [127] housed on a SCALE board executing SPARX-64/128 with the reference C implementation [128]. The SNR of this device, estimated by the method proposed in [12], was 2.823 for all 16 bits of s and 0.043 for the 4 targeted bits. Although the correlations are basically stabilised for more than 1000 traces, the attack failed even with 2000 traces as we present in Figure 4.4. We additionally show the correlations of all 16 bits of s under the full key $(0x2200, 2233)$ in Figure 4.4a for reference.

In Figure 4.4a, the correlation of our 4 targeted bits reaches its peak 0.098 at time point 465 synchronously with the referenced full key correlation. The selected time point is unlikely to be false positive; however, the correct key failed to be distinguished at this time point as we show in Figure 4.4b. We suspect two major factors that might have contributed to this result:

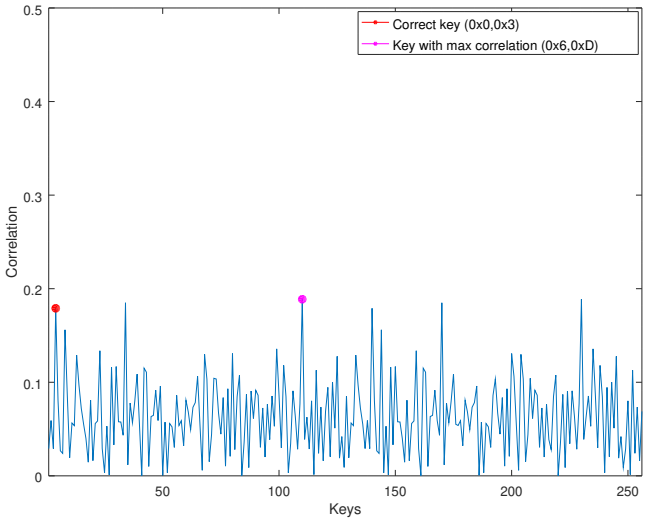
- The character of the leakage of the addition instruction on our targeted device was not well captured by the HW prediction.

- Ghost peaks due to the confusing incorrect key candidates as explained in Section 4.3.4.

In a non-profiling scenario, the adversary is unable to acquire an accurate prediction model for the correlation attack and has to rely on a classic model,

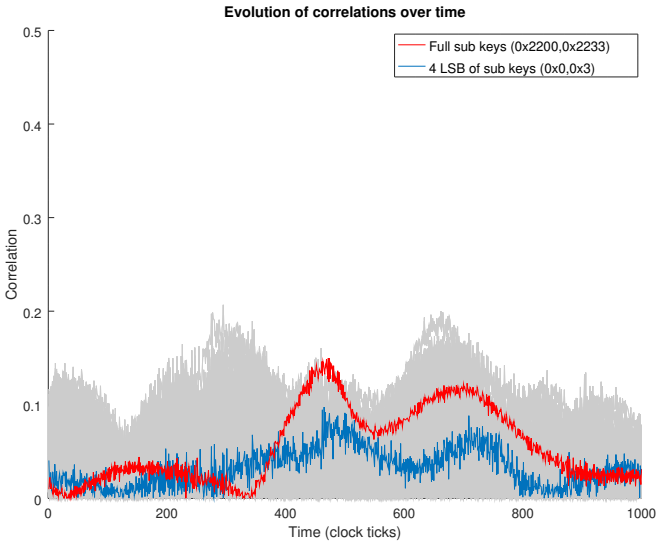


(a) SNR=2^1

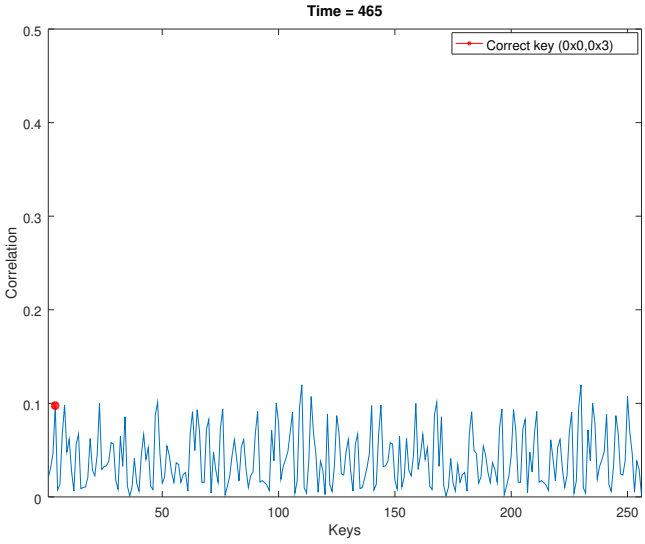


(b) SNR=2^{-3}

Figure 4.3: Correlations for SPARX modular addition using 500 simulated traces



(a) Evolution of the correlations



(b) Correlations of all key candidates at point 465

Figure 4.4: CPA on modular addition with 2000 traces

typically the HW, which we have shown to be ineffective in our experiments. Therefore we consider our results so far as evidence of the claimed "intrinsic resilience" of ARX ciphers stated in [70].

4.5 A Chosen Message DPA Against Modular Addition

In this section, we propose a novel chosen message DPA strategy against modular addition that was inspired by [129] and [130]. The new strategy features a computational complexity which scales linearly with the size of operands n, and can be used as an alternative to the divide-and-conquer approach. However, this attack imposes a relatively strong assumption on the leakage behaviour of target device, i.e. the measured leakage requires to be proportional to the Hamming Weight of the data being processed; therefore **the method may lack practical implication on certain architectures that do not hold the above assumption of Hamming Weight leakage.**

In a nutshell, our novel attack technique requires chosen inputs and recovers α, β using $2 \cdot c \cdot (n+1)$ leakages (c is a constant), and minimal, constant computational overhead. The attack requires the adversary to first obtain the leakage of an arbitrary pair of "base" input (x, y), and then the leakages of "alternative" inputs $\{\mathcal{F}_i(x), y\}$ for $i \in [0, n-1]$, that is, inputs with one bit of x flipped for each bit of x from the MSB to the LSB. By observing the differences induced by flipping each bit of x, the adversary first recovers the modular sum corresponding to the base input (x, y) and then derives (α, β) by solving a set of equations.

Our approach first tackles this in an ideal, non-noisy leakage scenario, thus $\mathcal{L}(x, y) = HW(s)$. Note that each of the chosen pairs of messages is identical to (x, y) except for one bit. We show that the change in Hamming

weights provides sufficient information to successively recover the bits of s starting from the most significant bit. With the same idea, we also recover $s(\widetilde{x}, y)$ (where $\widetilde{x}$ refers to the one's complement of x). Using $S(x, y)$ and $s(\widetilde{x}, y)$ we construct a set of two equations, which enable us to solve for (α, β).

A technical detail is that the presence of two exclusive-or operations as well as a modular addition operation slightly complicates the analysis of the effect of flipping a bit of the message x on $HW(S(x, y))$. We handle this by expressing the effect of flipping the ith bit of x as corresponding to a difference of $+2^i$ or -2^i for $S(x, y)$. Though we do not know the difference exactly, we show that using the change in Hamming weights we can predict the exact difference.

Finally, using some standard DPA techniques, we solve the noisy leakage case (i.e., the leakage is now $HW(S(x, y))$ + Gaussian noise) essentially by replacing comparisons between Hamming weights with statistical tests.

4.5.1 Formalisation of Attack

To describe our new algorithm, we first formalise attacking the ARX key exploiting the ideal and noisy HW leakage as the Hidden Adder Problem (HAP, Definition 2) and Noisy Hidden Adder Problem (NHAP, Definition 3), respectively.

Definition 2 (Hidden Adder Problem (HAP))**.** *Let (α, β) be randomly chosen from $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$. The adversary chooses pairs $x, y \in \mathbb{Z}_{2^n}$ and obtains leakage of the form $HW(S(x, y)) = HW((x \oplus \alpha) \boxplus (y \oplus \beta))$ for each of the pairs. The adversary must then recover (α, β).*

Definition 3 (Noisy Hidden Adder Problem (NHAP))**.** *Let (α, β) be randomly chosen from $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$. The adversary chooses pairs $x, y \in \mathbb{Z}_{2^n}$ and obtains leakage of the form $\mathcal{L}_{\alpha,\beta}(x, y) = HW(S(x, y)) + e = HW((x \oplus \alpha) \boxplus (y \oplus \beta)) + e$ where $e \sim \mathcal{N}(0, \sigma^2)$. The adversary must then recover (α, β).*

Notice that both the HAP (as well as NHAP) have exactly two solutions, as we have proved in Proposition 1. Indeed, through the execution of our algorithm it is provable that there exists two and only two solutions to HAP.

To explain our attack, we now define two sub-problems of HAP and NHAP called the Hidden Sum Problem (HSP) and the Noisy Hidden Sum Problem (NHSP), as described in Definition 4 and Definition 5.

Definition 4 (Hidden Sum Problem (HSP))**.** *Let* (α, β) *be randomly chosen from* $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$*. The adversary chooses pairs* $x, y \in \mathbb{Z}_{2^n}$ *and obtains leakage of the form* $HW(S(x,y)) = HW((x \oplus \alpha) \boxplus (y \oplus \beta))$ *for each of the pairs. The adversary must then recover* $S(x,y)$*.*

Definition 5 (Noisy Hidden Sum Problem (NHSP))**.** *Let* (α, β) *be randomly chosen from* $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$*. The adversary chooses pairs* $x, y \in \mathbb{Z}_{2^n}$ *and obtains leakage of the form* $\mathcal{L}_{\alpha,\beta}(x,y) = HW(S(x,y)) + e = HW((x \oplus \alpha) \boxplus (y \oplus \beta)) + e$ *where* $e \sim \mathcal{N}(0, \sigma^2)$*. The adversary must then recover* $S(x,y)$*.*

The adversaries in HSP and NHSP are given exactly the same form of leakage as in the adder problems (HAP and NHAP). Only their goals are changed to recover the sum $S(x,y)$ in the sum problems rather than the sub-keys in the adder problems. Later, in Section 4.5.4, we will show that HAP can be reduced to HSP.

Without loss of generality, we always consider the general case where $n \geq 2$. For the special case where $n = 1$, we have

$$HW(S(x,y)) = S(x,y) = x \oplus \alpha \oplus y \oplus \beta$$

which immediately gives (α, β) given $HW(S(x,y)), x$ and y.

4.5.2 Solving the Hidden Sum Problem

Our solution to the HSP recursively recovers $S(x,y)$ one bit at a time. Starting from the MSB down to the LSB, we flip each bit of x and observe

the resulting differences in the HW leakage. In the end we recover all bits of the hidden sum $S(x, y)$.

Since flipping $[x]_i$ also flips the bit $[x \oplus \alpha]_i$, this effectively changes the sum by $\pm 2^i$ due to commutativity of addition. In the following sections we explain how to exploit this property of $S(x, y)$ to solve HSP.

Define $\Delta s_y([x]_i)$ to be the difference in $S(x, y)$ induced by flipping $[x]_i$. We have:

$$\Delta s_y([x]_i) \equiv s(\mathcal{F}_i(x), y) - S(x, y) \equiv \pm 2^i \pmod{2^n}. \tag{4.20}$$

Equivalently,

$$s(\mathcal{F}_i(x), y) = S(x, y) \pm 2^i. \tag{4.21}$$

Recovering the MSB

In this section we give a solution that recovers the MSB of s, which is the base case for our algorithm.

Recall Equation (4.5) that flipping $[x]_{n-1}$ effectively flips $[s]_{n-1}$; thus the adversary can determine that $[s]_{n-1} = 0$ if the HW increases (from 0 to 1) and vice versa.

Lemma 1. *Given an arbitrary chosen input x, y, HW of the sums $S(x, y)$ and $s(\mathcal{F}_{n-1}(x), y)$, the MSB of the sum s is:*

$$[s]_{n-1} = \begin{cases} 0 & \text{if} \quad HW(s'_{n-1}) - HW(s) > 0, \\ 1 & \text{if} \quad HW(s'_{n-1}) - HW(s) < 0. \end{cases}$$

Proof. We can write $HW(s)$ as:

$$HW(s) = HW([s]_{n-1}) + HW([s]_{n-2}[s]_{n-3}...[s]_1[s]_0). \tag{4.22}$$

Also,

$$s'_{n-1} \equiv \Delta s_y([x]_{n-1}) + s \equiv \Delta s_y([x]_{n-1}) + [s]_{n-1}2^{n-1ni} + \sum_{i=0}^{n-2}[s]_i \cdot 2^i \pmod{2^n}. \tag{4.23}$$

Note that $[s]_{n-1} \in \{0,1\}$, $\Delta s_y([x]_{n-1}) \in \{+2^{n-1}, -2^{n-1}\}$ according to Equation (4.20), and $-2^{n-1} \mod 2^n = +2^{n-1}$. Equation (4.23) can, therefore, be categorised into four cases:

1. If $[s]_{n-1} = 0$, $\Delta s_y([x]_{n-1}) = +2^{n-1}$, then

$$s'_{n-1} = (+2^{n-1} + 0 \cdot 2^{n-1} + \sum_{i=0}^{n-2}[s]_i 2^i) \mod 2^n = [1][s]_{n-2}...[s]_1[s]_0.$$

2. If $[s]_{n-1} = 0$, $\Delta s_y([x]_{n-1}) = -2^{n-1}$, then

$$s'_{n-1} = (-2^{n-1} + 0 \cdot 2^{n-1} + \sum_{i=0}^{n-2}[s]_i 2^i) \mod 2^n = [1][s]_{n-2}...[s]_1[s]_0.$$

3. If $[s]_{n-1} = 1$, $\Delta s_y([x]_{n-1}) = +2^{n-1}$, then

$$s'_{n-1} = (+2^{n-1} + 1 \cdot 2^{n-1} + \sum_{i=0}^{n-2}[s]_i 2^i) \mod 2^n = [0][s]_{n-2}...[s]_1[s]_0.$$

4. If $[s]_{n-1} = 1$, $\Delta s_y([x]_{n-1}) = -2^{n-1}$, then

$$s'_{n-1} = (-2^{n-1} + 1 \cdot 2^{n-1} + \sum_{i=0}^{n-2}[s]_i 2^i) \mod 2^n = [0][s]_{n-2}...[s]_1[s]_0.$$

Observe that in Cases 1 and 2, where $[s]_{n-1} = 0$, we have

$$s'_{n-1} = [1][s]_{n-2}...[s]_1[s]_0.$$

Algorithm 1 Compute MSB of s

```
function [s]_{n-1} = GetMsb(x, y)
    ΔHW = HW(s'_{n-1}) − HW(s);
    if ΔHW > 0 then
        return 0;
    else
        return 1;
    end if
end function
```

Similarly, in Cases 3 and 4 where $[s]_{n-1} = 1$, we have

$$s'_{n-1} = [0][s]_{n-2}...[s]_1[s]_0.$$

Therefore, we obtain

$$HW(s'_{n-1}) = \begin{cases} HW(1) + HW([s]_{n-2}[s]_{n-3}...[s]_1[s]_0) & \text{if } [s]_{n-1} = 0, \\ HW(0) + HW([s]_{n-2}[s]_{n-3}...[s]_1[s]_0) & \text{if } [s]_{n-1} = 1. \end{cases} \tag{4.24}$$

Denote by ΔHW_{n-1} the (signed) difference in HW between s and s'_{n-1}. Subtracting Equation (4.24) by Equation (4.22), we have:

$$\Delta HW_{n-1} = HW(s'_{n-1}) - HW(s) = \begin{cases} HW(1) - HW(0) = +1 & \text{if } [s]_{n-1} = 0, \\ HW(0) - HW(1) = -1 & \text{if } [s]_{n-1} = 1. \end{cases} \tag{4.25}$$

Observing Equation (4.25), we can see that the sign of ΔHW_{n-1} solely depends on $[s]_{n-1}$. Since both $HW(s'_{n-1})$ and $HW(s)$ can be obtained as (ideal) leakage, we can thus recover $[s]_{n-1}$ by computing ΔHW_{n-1} and then applying Equation (4.25). □

Algorithm 1 provides the pseudo code for recovering the MSB.

Recovering the m-th bit

In Section 4.5.2 we explained how the MSB of $s = S(x, y)$ can be recovered from the noiseless HW leakage. We now show how to recover the remaining bits.

Lemma 2. *Suppose we flip the bit* $[x]_{n-m}$*. If:*

- $HW(s'_{n-m}) > HW(s)$*, then* $[s]_{n-m} = 0$*,*
- $HW(s'_{n-m}) = HW(s)$*, then* $[s]_{n-m} = \widetilde{[s]}_{n-(m-1)}$*,*
- $HW(s'_{n-m}) < HW(s)$*, then* $[s]_{n-m} = 1$*,*

for $2 \leq m \leq n$*.*

Proof. We assume that the higher-order $m-1$ bits of s:

$$s_{known} = [s]_{n-1}[s]_{n-2}\cdots[s]_{n-(m-1)}$$

have been determined. The goal is then to recover the next bit $[s]_{n-m}$. According to Equation (4.21), when $[x]_{n-m}$ is flipped, we obtain the flipped sum s'_{n-m}:

$$s'_{n-m} = s + \Delta s_y([x]_{n-m}) \tag{4.26}$$

where $\Delta s_y([x]_{n-m}) = \pm 2^{n-m}$.

In the RHS of Equation (4.26), bits "lower" than $[s]_{n-m}$ are unchanged after the addition operation and thus do not affect the HW. On the other hand, the addition to (or subtraction from) $[s]_{n-m}$ may potentially generate a carry bit that propagates through bits "higher" than $[s]_{n-m}$ and results in a change of HW.

Let ΔHW_{n-m} be the (signed) change of HW induced by flipping $[x]_{n-m}$:

$$\Delta HW_{n-m} = HW(s'_{n-m}) - HW(s). \tag{4.27}$$

We can categorise ΔHW_{n-m} by:

- Whether or not there exists a carry bit (either positive or negative),
- If there does exist a carry bit, then
  - Whether the carry triggers an overflow (and hence modular reduction).

We next analyse each of the above cases.

No carry bit. In the following conditions there is no carry bit:

1. If $[s]_{n-m} = 0$ and $\Delta s_y([x]_{n-m}) = +2^{n-m}$, then $\Delta HW_{n-m} = +1$.
2. If $[s]_{n-m} = 1$ and $\Delta s_y([x]_{n-m}) = -2^{n-m}$, then $\Delta HW_{n-m} = -1$.

Otherwise there must exist a carry bit.

Carry bit. The existence of a carry bit implies that either one of the following condition is satisfied:

- Case $C_1 : [s]_{n-m} = 1$ and $\Delta s_y([x]_{n-m}) = +2^{n-m}$.
- Case $C_2 : [s]_{n-m} = 0$ and $\Delta s_y([x]_{n-m}) = -2^{n-m}$.

is satisfied. This can be further categorised into:

Overflow. In this case, all the bits of s_{known} are flipped after the addition:

1. In Case C_1, it is required that $s_{known} = [1]^{m-1}$. The propagation results in s_{known} flipped to $[0]^{m-1}$, with $\Delta HW_{n-m} = -m$.
2. In Case C_2, it is required that $s_{known} = [0]^{m-1}$. The propagation results in s_{known} flipped to $[1]^{m-1}$, with $\Delta HW_{n-m} = +m$.

Conditions				ΔHW_{n-m}
$[s]_{n-m}$	$\Delta s_y([x]_{n-m})$	Overflow?	s_{known}	
[0]	$+2^{n-m}$	No	$[*]^{m-1}$	+1
	-2^{n-m}	Yes	$[0]^{m-1}$	+m
		No	$[*]^{m-(k+2)}[1][0]^k$	+k
[1]	$+2^{n-m}$	Yes	$[1]^{m-1}$	-m
		No	$[*]^{m-(k+2)}[0][1]^k$	-k
	-2^{n-m}	No	$[*]^{m-1}$	-1

Table 4.1: ΔHW under different conditions, where $2 \leq m \leq n$, $0 \leq k \leq m$.

No overflow. In this case, only a part of s_{known} is flipped after adding $\Delta s_y([x]_{n-m})$. Denote by $k \in [0, m-2]$ the number of bits flipped in s_{known} before the carry propagation terminates, then

1. In Case C_1, the carry propagation terminates at the least significant [0] of s_{known} which is required to have the form

$$s_{known} = [s]_{n-1}...[s]_{n-(m-(k+2))}[0][1]^k.$$

After the addition with $\Delta s_y([x]_{n-m}) = +2^{n-m}$, s_{known} changes to

$$[s]_{n-1}...[s]_{n-(m-(k+2))}[1][0]^k.$$

Therefore $\Delta HW_{n-m} = -k$.

2. Case C_2 is just the opposite of C_1 with $\Delta HW_{n-m} = +k$.

Table 4.1 summarises the above scenarios. It is shown in the table that positive ΔHW_{n-m} implies $[s]_{n-m} = [0]$ and negative ΔHW_{n-m} implies $[s]_{n-m} = [1]$ when both $m, k \geq 0$. The case $\Delta HW_{n-m} = 0$ is only possible when $k = 0$, which indicates that a carry bit exists without overflow. Referring to Table 4.1, in such a case s_{known} is required to be either:

- $s_{known} = [*]^{m-2}[1]$, for $[s]_{n-m} = 0$, or
- $s_{known} = [*]^{m-2}[0]$, for $[s]_{n-m} = 1$.

Algorithm 2 Compute m-th significant bit $[s]_{n-m}$ $(2 \leq m \leq n)$

function $[s]_{n-m} = GetNextBit(x, y, m, [s]_{n-1}[s]_{n-2}...[s]_{n-(m-1)})$
  $\Delta HW = HW(s'_{n-1}) - HW(s)$; ▷ Refer Table 4.1
  if $\Delta HW > 0$ **then**
    return 0;
  else if $\Delta HW < 0$ **then**
    return 1;
  else ▷ $\Delta HW == 0$
    if $[s]_{n-m+1} == 0$ **then**
      return 1;
    else
      return 0;
    end if
  end if
end function

In either case, $[s]_{n-m}$ can be determined by the LSB of s_{known}. To summarise, given ΔHW_{n-m}, we can uniquely determine $[s]_{n-m}$ from Table 4.1. □

Algorithm 2 provides the psuedo code to compute s_{n-m} for $2 \leq m \leq n$.

4.5.3 Complete Solution to HSP

Combining the methods described in Section 4.5.2 and Section 4.5.2, we now have a full solution to the HSP, as summarised in Algorithm 3. Notice that the same $HW(s)$ can indeed be reused in Algorithm 1 and Algorithm 2 ; hence Algorithm 3 only needs $n + 1$ traces to recover $S(x, y)$.

Algorithm 3 Compute s

function $s = GetSum(x, y)$
  ▷ We initialise the sum to its MSB
  $s = GetMsb(x, y)$;
  ▷ Recover one bit at a time from 2nd MSB to LSB
  for $(m = 2; m \leq n; m++)$ **do**
    $s = [s][GetNextBit(x, y, m, s)]$;
  end for
  return s;
end function

4.5.4 Solving HAP

In this section we show how HAP (cf. Definition 2) can be solved using a solution to HSP (cf. Section 4.5.2).

Lemma 3. *Let* $\Delta := ((S(x,y) - s(\widetilde{x},y) - 1) \pmod{2^n}) >> 1$. *The solutions to HAP are:*

$$\begin{cases} \alpha = \Delta \oplus x \\ \beta = y \oplus ((S(x,y) - \Delta) \pmod{2^n}) \end{cases}$$

or

$$\begin{cases} \alpha = (\Delta \boxplus 2^{n-1}) \oplus x \\ \beta = y \oplus ((S(x,y) - (\Delta + 2^{n-1})) \pmod{2^n}), \end{cases}$$

for arbitrary $x, y \in \mathbb{Z}_{2^n}$.

Proof. Observe that for any $x \in \mathbb{Z}_{2^n}$, we have

$$\widetilde{x} \oplus \alpha = \widetilde{x \oplus \alpha} = 2^n - 1 - (x \oplus \alpha).$$

Hence

$$\begin{aligned} S(x,y) - s(\widetilde{x},y) &= ((x \oplus \alpha) + (y \oplus \beta)) - ((\widetilde{x} \oplus \alpha) + (y \oplus \beta)) \pmod{2^n}, \\ &= (x \oplus \alpha) - (2^n - 1 - (x \oplus \alpha)) \pmod{2^n}, \\ &= 2(x \oplus \alpha) + 1 \pmod{2^n}. \end{aligned}$$

Note that we have already computed the values $S(x,y)$ and $s(\widetilde{x},y)$ in Section 4.5.2. Since 2 is not co-prime to the modulo 2^n, there are exactly two values of $x \oplus \alpha$ that satisfy the above equation: Δ and $\Delta \boxplus 2^{n-1}$. Hence the lemma follows. □

Algorithm 4 provides the pseudo code for solving HAP. The algorithm

has trace complexity $O(n)$ – requiring $2n + 2$ calls to the (ideal) leakage function.

Algorithm 4 Compute (α, β)

function $(\alpha, \beta) = GetAlphaBeta(\text{void})$
  Pick arbitrary (x, y);
  ▷ Compute $S(x, y)$ and $s(\widetilde{x}, y)$ by Algorithm 3
  $S0 = GetSum(x, y)$;
  $S1 = GetSum(\widetilde{x}, y)$;
  ▷ Recover (α, β) using Lemma 3
  $a1 = ((S0 - S1 - 1) \pmod{2^n}) >> 1$;
  $a2 = a1 \oplus 2^{n-1}$;
  $b1 = y \oplus ((S0 - a1) \mod 2^n)$;
  $b2 = y \oplus ((S0 - a2) \mod 2^n)$;
  return $\{(a1, b1), (a2, b2)\}$;
end function

4.5.5 Progressing to Noisy Leakage

In a real world attack setting an adversary is unlikely to have noise free leakages. We thus now consider how to translate the developed attack strategy into a more realistic setting.

In principle, the reduction explained in Section 4.5.4 also holds for NHAP to NHSP, as long as the adversary is able to recover $S(x, y)$ given the noisy leakage in NHSP. Further examining the HSP solution in Section 4.5.2, we see that it is indeed sufficient to solve HSP given only the signs of the difference $\Delta HW_i = HW(s'_i) - HW(s)$, $i \in [0, n-1]$. In the case of noisy leakages we can reveal this difference by sampling the leakage function (i.e. the device) multiple times on the same input. We thus get two sets of leakages:

$$S_1 = \{HW(s'_i) + e\},$$
$$S_2 = \{HW(s) + e\}.$$

Clearly by subtracting the averages of these sets (i.e. conducting a classical DPA-style attack), we can recover ΔHW_i also in the noisy case. Moreover,

Algorithm 5 Determine sign of ΔHW

```
function ΔHW = CompareHW(S_1, S_2)
    (t, p) = test(S_1, S_2)
    if p/2 ≥ SignificanceLevel then                ▷ Accept ΔHW = 0
        return 0
    else
        if t > 0 then                              ▷ +1 for positive ΔHW.
            return +1
        else                                       ▷ −1 for negative ΔHW.
            return −1
        end if
    end if
end function
```

because we are only interested in the sign of ΔHW_i, we can hope that in practice we don't require 'large' sets. To add a bit more rigour, we opted to implement a standard two-tailed t-test in our experiments. A two-tailed test can tell us if

- $HW(s'_i) = HW(s)$,
- $HW(s'_i) > HW(s)$, or
- $HW(s'_i) < HW(s)$.

Algorithm 5 outlines the pseudo code that determines the sign of ΔHW. It first conducts a two-tailed test:

$$H_0 : \overline{S_1} = \overline{S_2}$$
$$H_1 : \overline{S_1} \neq \overline{S_2}$$

using a set significance level, and interprets the result in terms of the sign of ΔHW_i.

4.5.6 Word Extension

For some software implementations the registers in the processor could be larger than the word length of the implemented cipher, e.g. it is easy to

imagine running a 16-bit SPARX on an ARM Cortex-M0 which has 32-bit registers. In such a case Algorithm 1 may not recover the correct MSB of $S(x, y)$ as the carry bit corresponding to the MSB of $S(x, y)$ will actually be stored in the higher part of the register until it is used in a later computation.

There is a simple fix to this issue. We can simply treat the over-flow carry bit as the MSB of the $n + 1$ bit sum, then enumerate over its possible values which are merely $\{0, 1\}$. Algorithm 3 needs to be updated accordingly to initialise two possible s with different MSBs, and then call Algorithm 2 on each one of them, although the same traces can be reused between two calls.

A side effect of the above fix is that an additional incorrect s will also be returned by Algorithm 3. Enumerating them in Algorithm 4 eventually results in 8 pairs of (α, β). To remove the invalid sub-keys, one can simply re-run the attack multiple times starting with different choices of (x, y), then take the intersection of (α, β) pairs returned by the attacks as the correct sub-keys.

4.5.7 Compatible Power Leakage Models

As explained in Section 4.5.5, the attack does not exploit any actual value of the leaked HW but only signs of the HW differences. Consequently, our method can be applied to any power leakage model that is "monotonic" w.r.t. the HW function. Additionally, for any other power leakage model that "essentially" leaks $S(x, y)$ and $s(\widetilde{x}, y)$ for arbitrary (x, y), one can trivially recover the sub-keys (α, β) (cf. Section 4.5.4). For instance, the above class includes those power leakage models that are monotonic w.r.t. the value of the variable itself.

4.5.8 Requirement of Chosen Inputs

Recall from Section 4.5.4 that to recover (α, β), the adversary needs to know the input (x, y) and the corresponding sums $S(x, y)$ and $s(\widetilde{x}, y)$. For this,

$n+1$ leakage function calls of a specific form are needed in order to recover $s(x, y)$. Similarly, to recover $s(\widetilde{x}, y)$ another $n+1$ specific leakage function calls are required. Although such a requirement can be easily satisfied in a chosen-message set up, in a setting where the adversary can only sample random values, there seems to be no obvious way to get such leakage cheaply. Therefore the attack in its current form would not be applicable given only uniform randomly chosen pairs (x, y). Thus finding ways to deal with this scenario makes an interesting topic for future work.

4.5.9 Experiments

The attack is mainly affected by three parameters in practice. These are the noise distribution (characterised by σ), the number of repeat queries to the leakage function, N, and the significance level of the two-tailed test. It is well understood that these three quantities jointly determine how well a test performs, and thus in turn, how often our attack succeeds. The success rate of an attack is simply given as the rate of correctly recovering (α, β):

$$\text{Success Rate} = \frac{\#(\text{Full sub-key recovered})}{\#(\text{Experiments})}. \tag{4.28}$$

Our set up is motivated by the SPARX [66] cipher which is designed for 16-bit architectures ($n = 16$). Algorithm 5 is implemented using a t-test. S_1 and S_2 are chosen to have the same sample size for simplicity. We simulated the attack using different configurations where $\sigma \in [0.1, 6]$ and $N \in [100, 1000]$. Recall that N is the number of samples used for each tests. A complete attack hence uses $2N(n+1)$ traces. It is well known that noisy traces require more samples, and thus the results follow an expected and natural trend.

However, the new strategy did not manage to recover the key on the real traces. We suspect it might be caused by a leakage model incompatible with

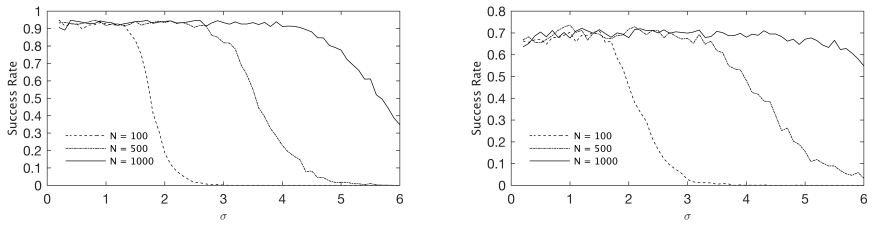


Figure 4.5: Simulation results for the attack. The left figure shows the success rate as a function of the number of leakages (per query) when choosing a significance of 0.01. The right figure is identical but for a significance of 0.05.

that described in Section 4.5.7.

4.6 CPA Against Rotation and XOR

We further extended our experiments to the more "difficult" targets in ARX-Boxes described in [70], which are the linear operations XOR and rotation. For a fair comparison with the experiments in Section 4.4 in terms of key enumeration space, we selected the target intermediate to be the 8 lower-order bits of $(x \oplus \alpha)$ of SPARX. Note that rotation has no effect in changing the HW of the operand; hence XOR and rotation share the same predicted of leakage in the HW model.

The correlation attack is again implemented with HW predictions and applied on real traces collected with the same setup of the experiments as in Section 4.4 (time axis shifted to XOR and rotate instructions). To our surprise, the result seems contradict to those reported by [70] which concluded that the addition is the most leaky instruction among those used in ARX ciphers. The key was successfully recovered using only 500 traces targeting the XOR as we show in Figure 4.6, which is much less than we used for our failed attack against modular addition. Figure 4.4a shows the (absolute values of) correlations over time. Three major local peaks are observed, with the last two seemingly overlapped. The correct key $0x11$ was identified on the last peak (time 520) with the highest correlation of 0.408.

The correct key has shows a clear distinguishing margin ahead of other key candidates at point 520 where we present more detail in Figure 4.6b. The symmetry in Figure 4.6b is due to the fact that:

$$HW(\tilde{x} \oplus \alpha) = HW(\widetilde{x \oplus \alpha}) = n - HW(x \oplus \alpha) \tag{4.29}$$

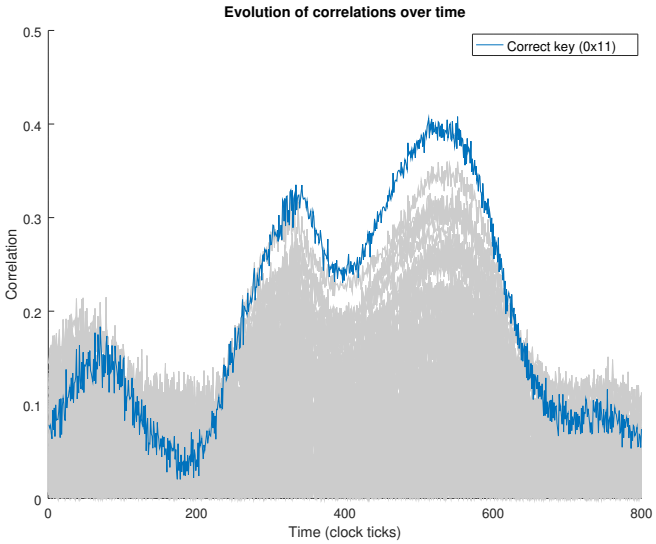
where $n = 8$ is the size of the target intermediate in bits. Equation (4.29) implies that complemented keys α and $\tilde{\alpha}$) result in HW leakage predictions complemented modulo n and thus the same absolute value of correlation.

We suspect the unexpected leakage in our results that is contrary to [70] arises from a combination of the SPARX software implementation we show in Figure 4.7, and a "signal amplification" effect deriving from the results of [131].

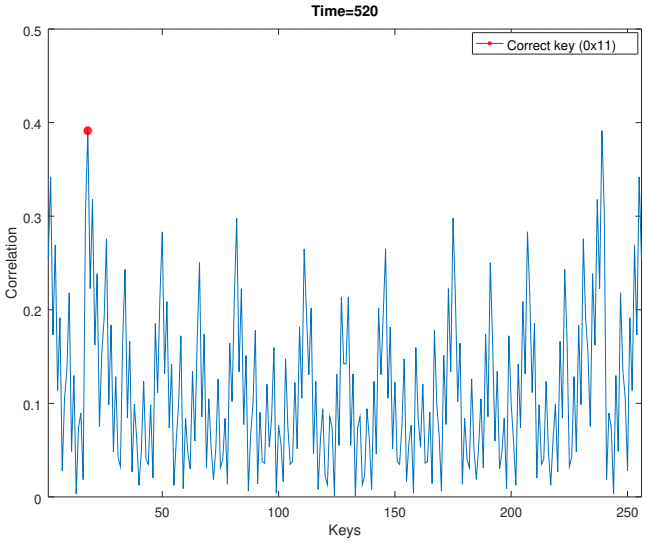
The C code of Figure 4.7a is taken from the SPARX reference implementation [128] and Figure 4.7b is the corresponding assembly compiled with ARM toolchain arm-none-eabi 6.3.1. The ARM Cortex M0 was reportedly known to predominantly leak the HW [131] and the assembly in Figure 4.7b shows that consecutive instructions have involved the operands of XOR and rotation, the HW of which are the same. These operands include:

1. r1 in L2,
2. Both r2 r1 in L2,
3. r1 as both input and output in L6,
4. Both r1 and r2 in L7, and
5. r1 as input in L10.

Recall [131] has reported on this processor that the leakage of each instruction can be well approximated by a linear combination of independent leakage of operands in consecutive instructions; therefore if the same leakage, in this



(a) Evolution of correlations



(b) Correlations of all Key candidates at point 520

Figure 4.6: Correlations of XOR and rotation using 500 real traces

```c
//Rotate left for 16 bit registers.
#define ROTL(x,n) (((x)<<n)|((x)>>(16-(n))))

static void A(uint16_t * l, uint16_t * r)
{
        (*l) = ROTL((*l), 9);
        (*l) += (*r);
        (*r) = ROTL((*r), 2);
        (*r) ^= (*l);
}

static void sparx_encrypt(uint16_t * x, uint16_t k[][2
    * ROUNDS_PER_STEPS])
{
        ...
        //Key XOR.
        x[2 * b] ^= k[N_BRANCHES * s + b][2 * r];
        x[2 * b + 1] ^= k[N_BRANCHES * s + b][2 * r +
            1];
        //Rotation and Addition.
        A(x + 2 * b, x + 2 * b + 1);
    ...
}
```

(a) SPARX ARX implemented in C

```
@Key XOR
eors    r1, r2

@Rotation
lsrs    r2, r1, #7
lsls    r1, r1, #9
orrs    r1, r2

@Modular addition
adds    r1, r0, r1
```

(b) SPARX ARX-Box implemented in ARM assembly

Figure 4.7: Implementation of 16 bit rotation on ARM-M0 (32 bit)

case $HW(x \oplus \alpha)$, has occurred multiple times in consecutive instructions, the leakage will be accumulated for each instruction, resulting in an amplified leakage, as suggested by our results in Figure 4.6. Therefore, even though XOR and rotation instructions might independently be considered difficult to attack according to [70], their leakage could still be easily exploited when executed sequentially with the same operands in a careless implementation.

4.7 Summary

In this chapter, we performed a case study on the DPA properties of an ARX-Box generalised from SPARX. We first showed that, although modular addition is the only non-linear operation in ARX-Boxes, typical correlation attacks would hardly succeed against this target by its nature in Section 4.4, and proposed a novel chosen message DPA strategy in Section 4.5. On the other hand, linear operations, i.e. XOR and rotation, despite being difficult targets for DPA when taken independently, could be vulnerable in combination when implemented inappropriately.

Although correlation attacks against modular addition have all failed on the real traces in this chapter, our further research found that it is not necessarily immune to DPA attacks. The non-injective property of modular addition implies it is a natural target for generic distinguishers which are free from errors introduced by inaccurate prediction model, as will be shown later in Chapter 5 with a successful key recovery on the same real traces.

Chapter 5

A Novel Type of Generic Distinguisher – Ordinal

5.1 Introduction

IoT devices are constantly evolving. New technologies are being applied to the latest processors to improve their performance, reduce their size and, most importantly for IoT, make them more energy efficient. These changes have had a great impact on the SCA properties of new microcontrollers. As pointed out by [132] and [131], other than the typical HW and HD models which are widely observed on earlier embedded devices, such as 8 bit smart cards described in [30], the latest processors are more frequently showing distinctive leakage characters.

When it comes to attacking a device without knowledge of its leakage behaviour, some DPA methods such as CPA are no longer appropriate as their distinguishability heavily relies on the adversary correctly predicting the leakage, which cannot be done without knowing the leakage model. Alternatively, the adversary may turn to methods that require less knowledge about the leakage model which typically include template attack [38] and generic distinguisher [125]. Template attacks require the adversary to have

full control over a device identical to the target to build a template for the target through a profiling stage. In comparison, generic distinguishers can be applied on arbitrary power models without relying on templates which makes them beneficial when profiling is not possible. However, generic distinguishers are also restricted in the sense that they can only be applied on non-injective target functions, as described in [125].

In this chapter we begin by introducing some preliminaries of generic distinguishers in Section 5.2. We then explain in Section 5.3 the characteristics of the ordering of leakage, and how they could be exploited to construct generic distinguishers in Section 5.4. In Section 5.5 and Section 5.6 we demonstrate the experimental results of these generic distinguishers applied on various target functions that can be commonly seen in lightweight ciphers for IoT applications under different leakage models.

This chapter is based on my original, unpublished research jointly authored with my colleague Arnab Roy and my supervisor Elisabeth Oswald. In this work, I am responsible for proposing our new distinguishers and their implementations as well as conducting all the experiments.

5.2 Preliminaries

5.2.1 Notations

In Chapter 5 we frequently uses vector variables which we denote in **bold** font. For a function $f : X \to Y$ and a vector $\mathbf{X} \in X^n$, $f(\mathbf{X})$ denotes applying f to each component of $\mathbf{X}$ locally.

For two sequences $\mathbf{A}$ and $\mathbf{B}$,

$$\begin{aligned} \mathbf{A} &= (a_1, a_2, ..., a_n) \\ \mathbf{B} &= (b_1, b_2, ..., b_m) \end{aligned} \tag{5.1}$$

$\mathbf{A} \subset \mathbf{B}$ implies $\mathbf{A}$ is a subsequence of $\mathbf{B}$, i.e. :

$$\exists i, j, m \in \mathbb{Z} : (a_i, a_{i+1}, ..., a_{i+m}) = (b_j, b_{j+1}, ..., b_{j+m}) \tag{5.2}$$

We denote by $v = \mathbb{F}_k(x)$ the DPA target intermediate where k is the embedded secret key in the target function F and x the known message to the adversary. We denote by $\mathcal{I}$ and $\mathcal{O}$ the input and output space of F and $\mathcal{K}$ the key space.

In an univariate DPA attack, we denote the traces collected by the adversary as a set of the form $\mathcal{T} = \{(x, t)\}$, where x is the input of the trace and t the leakage value corresponding to x.

5.2.2 Generic Distinguishers

Generic distinguishers, formally defined in [133], are DPA strategies that work without making any assumption about the device leakage. In brief, unlike CPA explained in Chapter 2, which requires the adversary to predict the hypothetical leakage with an assumed power model, generic distinguishers exploit the equivalence classes induced by the key-hypothesised target function F_k [133]; thus they work without the adversary predicting a power model.

Due to their "assumption free" nature, generic distinguishers are specifically of interest in non-profiling scenarios against targets with unknown leakage behaviour.Among the existing generic distinguishers, mutual information (MI), and Kolmogorov-Smirnov (KS) are the most popular choices suggested in the literatures.

$$D_{\mathrm{KS}}(k) = \mathop{\mathbb{E}}_{v \in \mathcal{V}} \left[\sup_{l} |F_L(l) - F_{L|V=v}(t)| \right]. \tag{5.3}$$

KS has been shown to be favourable over MI in certain cases because it it does not require the explicit estimation of densities, but only the calculation

of empirical cumulative distribution functions. Density estimation is a process for which there is no optimal choice in general [134].

All these distinguisher definitions are set up such that the largest value indicates the correct key guess.

5.2.3 The Bit Dropping Trick

One major restriction of generic distinguishers is that they require the target function F to be non-injective, otherwise each key hypothesis would produce partitions that are permutations of each other and cannot be distinguished, as explained in [125]. Alternatively, it was pointed out in [135] that by ignoring certain bits in the target intermediate v, one could effectively reduce an injective $F_{k^*}(x)$ into a non-injective function $drop_B(F_{k^*}(x))$ where B are the indexes of bits to be dropped. This technique is referred to "bit dropping".

Bit dropping has been proven practically effective in several cases and has been studied in several places in the literatures of [135][136][133][137]. It was shown that the selection of B greatly impacts the performance of the distinguishers to be applied later on and it is impossible to choose the optimal B without having a priori knowledge of the device leakage function M_D.

5.3 Ordering of Leakage

In this section we explain the observations of the ordering of leakage which our novel distinguishers will be based on in a noiseless setting. Like other generic distinguishers, our distinguishers too require the target function to be non-injective which implies $|\mathcal{I}| > |\mathcal{O}|$. We denote by m_i the device leakage corresponding to a target intermediate v_i:

$$m_i := M_D(v_i) \tag{5.4}$$

We define the ordered sequence $\mathbf{M}$ of m_i:

$$\mathbf{M} := (m_1, m_2, ..., m_{|\mathcal{O}|-1}, m_{|\mathcal{O}|}) \tag{5.5}$$

such that $p < q \implies m_p \leq m_q$, i.e. $m_1 \leq m_2 \leq ... \leq m_{|\mathcal{O}|-1} \leq m_{|\mathcal{O}|}$.

Since $|\mathcal{I}| > |\mathcal{O}|$, there exists at least a pair of collision inputs (x_1, x_2) such that:

$$\exists x_1, x_2 \in \mathcal{I}, x_1 \neq x_2 : F_{k^*}(x_1) = F_{k^*}(x_2) \tag{5.6}$$

Equation (5.6) in fact represents a collision of F_{k^*} at two different inputs (x_1, x_2). Extending it for all $v \in \mathcal{O}$, we define the sets of inputs that collide at a specific target intermediate v as collision set $\mathcal{C}_v$:

$$\mathcal{C}_v := \{x : F_{k^*}(x) = v\} \tag{5.7}$$

For simplicity, we denote y_i the device leakage corresponding to an input x_i:

$$y_i := M_D(F_{k^*}(x_i)) \tag{5.8}$$

Note that all inputs from the same collision set have the same device leakage $M_D(v)$:

$$M_D(v) = y_1 = y_2 = ... = y_{m-1} = y_m \tag{5.9}$$

for $\forall x_1, x_2, ..., x_{m-1}, x_m \in \mathcal{C}_v$.

Define $\mathbf{R}$ the ordered sequence of y_i for all inputs $x_i \in \mathcal{I}$:

$$\mathbf{R} = (y_1, y_2, ..., y_{|\mathcal{I}|-1}, y_{|\mathcal{I}|}) \tag{5.10}$$

such that $p < q \implies y_p \leq y_q$, i.e. $y_1 \leq y_2 \leq ..., \leq y_{|\mathcal{I}|-1} \leq y_{|\mathcal{I}|}$.

We define $\mathbf{X}$ the corresponding input sequence of $\mathbf{R}$:

$$\mathbf{X} := (x_1, x_2, ..., x_{|\mathcal{I}|-1}, x_{|\mathcal{I}|}) \tag{5.11}$$

such that $\mathbf{R} = M_D(F_{k^*}(\mathbf{X}))$. Specifically we have $y_i = M_D(F_{k^*}(x_i))$ for all $1 \leq i \leq |\mathcal{I}|$. Note that without consideration of noise, obtaining $\mathbf{R}$ and $\mathbf{X}$ does not require the knowledge of M_D. Since the traces are given in the form $\{(x, t)\}$, sorting them by t one immediately obtains $\mathbf{R}$ as the sequence of t and $\mathbf{X}$ the sequence of x.

Recall from Equation (5.9) that all inputs in a collision set $x_i \in \mathcal{C}_v$ have the same device leakage $y_i = M_D(v)$; thus their corresponding y_i would be in consecutive positions in $\mathbf{R}$. Consequently, $\mathbf{R}$ is indeed $\mathbf{M}$ with each component repeated $|\mathcal{C}_{v_i}|$ times:

$$\mathbf{R} = (\underbrace{y_1, \ldots, y_i}_{=\{m_1\}^{|\mathcal{C}_{v_1}|}}, \underbrace{y_{i+1}, \ldots, y_j}_{=\{m_2\}^{|\mathcal{C}_{v_2}|}}, \ldots, \underbrace{y_r, \ldots, y_{|\mathcal{I}|}}_{=\{m_{|\mathcal{O}|}\}^{|\mathcal{C}_{v_{|\mathcal{O}|}}|}}) \tag{5.12}$$

Note that for components of $\mathbf{X}$ in the same collision group $x_i \in \mathcal{C}_v$, it holds that $M_D^{-1}(y_i) = F_{k^*}(x_i) = v$. Thus define $\mathbf{V}$ the target intermediate sequence under the correct key k^*:

$$\mathbf{V} = M_D^{-1}(\mathbf{R}) = (\underbrace{v_1, \ldots, v_1}_{|\mathcal{C}_{v_1}|\text{times}}, \underbrace{v_2, \ldots, v_2}_{|\mathcal{C}_{v_2}|\text{times}}, \ldots, \underbrace{v_{|\mathcal{O}|}, \ldots, v_{|\mathcal{O}|}}_{|\mathcal{C}_{v_{|\mathcal{O}|}}|\text{times}}) = F_{k^*}(\mathbf{X}) \tag{5.13}$$

Although the values of $\{v_1, v_2, ..., v_{|O|}\}$ remain unknown without the knowledge of key k^* or the device leakage function M_D, $\mathbf{V}$ has a distinctive repetitive structure that could be exploited to recover the secret key k^*. Therefore, during an attack, the adversary proceeds as follows:

1. Construct $\mathbf{X}$ from the trace set $\mathcal{T}$.
2. For each key guess k, compute hypothetical intermediate sequence $\mathbf{V}_k = F_k(\mathbf{X})$.

In	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Out	11	15	3	2	10	12	9	1	6	7	8	0	14	5	13	4

Table 5.1: PRINCE S-Box

3. Test each $\mathbf{V}_k$ for the repetitive structure which must be held when $k = k^*$.

Example 3. Take a noiseless example where the target intermediate v is defined as the lowest 2 bits of the 16 bits PRINCE [138] S-Box output:

$$v = F_k(x) = SBox_{PRINCE}(x \oplus k) \mod 2^2 \tag{5.14}$$

where $k, x \in \mathbb{Z}_{2^4}$ and $SBox_{PRINCE}$ is given in Table 5.1.

Suppose the device leaks the HW of v and the correct key $k^* = 5$. Then, for instance, given an input $x = 1$, its corresponding leakage value t is:

$$t = HW(F_5((1))) = HW(2) = 1 \tag{5.15}$$

thus the adversary is given a trace $(x, t) = (1, 1)$.

During the attack, suppose the adversary has collected the following set of traces:

$$\begin{aligned}\mathcal{T} = \{(x,t)\} = \{&(0,0),(1,1),(2,1),(3,1),(4,2),(5,2),(6,1),(7,2),\\ &(8,1),(9,1),(10,0),(11,1),(12,2),(13,1),(14,0),(15,0)\}\end{aligned}$$

with the first components being plaintext known to the adversary and the second the corresponding leakage.

Sorting $\mathcal{T}$ by t gives:

$$\begin{aligned}\mathbf{T}_{sorted} = (&(0,0),(10,0),(14,0),(15,0),(1,1),(6,1),(9,1),(13,1),\\ &(2,1),(3,1),(8,1),(11,1),(4,2),(5,2),(7,2),(12,2))\end{aligned}$$

So the first and second components of $\mathbf{T}_{sorted}$ constitute $\mathbf{X}$ and $\mathbf{R}$ re-

spectively:

$$\mathbf{X} = (0, 10, 14, 15, 1, 6, 9, 13, 2, 3, 8, 11, 4, 5, 7, 12)$$
$$\mathbf{R} = (0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2)$$

For each key guess k, the adversary computes $\mathbf{V}_k = F_k(\mathbf{X})$. For example, if we substitute the correct key guess $k = 5$, we have:

$$\mathbf{V}_5 = F_5(\mathbf{X}) = (0, 0, 0, 0, 2, 2, 2, 2, 1, 1, 1, 1, 3, 3, 3, 3) \tag{5.16}$$

which holds the repetitive structure as demonstrated in Equation (5.13). In contrast, substituting an incorrect key, say $k = 0$, we have:

$$\mathbf{V}_0 = F_0(\mathbf{X}) = (3, 0, 1, 0, 3, 1, 3, 1, 3, 2, 2, 0, 2, 0, 1, 2) \tag{5.17}$$

which,unlike the case $k = 5$, does not hold the above mentioned repetitive structure as for the case $k = 5$.

Note that, due to the fact that the HW function is non-injective, there inevitably exist multiple target intermediates leaking exactly identical values which are $HW(v = 1) = HW(v = 2) = 1$ in this case. As a result, $\mathbf{T}_{sordted}$ would not be unique in this example and each one of them should be considered valid in later computations. Nevertheless, we argue that the practical impact of this issue is insignificant for the following reasons:

- We would be unlikely to observe multiple target intermediates having strictly identical leakage on any real device.
- Its impact would tend to be negligible compared to noise in practice.

Compared to existing generic distinguishers such as MI and KS which partition trace values based on predicted intermediate values, our observation of the ordering of leakage inspires an inverted approach that partitions predicted intermediate values based on the measured traces. For this idea

to work, we need to be able to define an ordering relationship on the traces, hence the name ordinal. We explain in detail how the above observation could be instantiated as distinguishers in Section 5.4.

5.4 Ordering-based Distinguishers

In Section 5.3 we explained the properties of the ordering of leakage which leads to the repetitive structure of $\mathbf{V}$. In this section, we develop two distinguishers, namely Ordinal-Entropy in Section 5.4.1 and Ordinal-Variance-of-Positions (Ordinal-VP) in Section 5.4.2, both inspired by this observation.

In principle, the major difference between the correct key and other incorrect keys lies in whether the distinctive repetitive structure holds for their corresponding hypothetical intermediate sequence $\mathbf{V}_k$, the computation of which is a common set up when implementing an ordinal distinguisher. Similar to the noiseless case explained in Section 5.3, the adversary constructs an approximation of $\mathbf{R}$, denote as $\hat{\mathbf{R}}$, by sorting the noisy traces $\hat{\mathcal{T}} = \{x, \hat{t}\}$ by their leakage values $\hat{t}$:

$$\hat{\mathbf{R}} = (\hat{t}_1, \hat{t}_2, ..., \hat{t}_{N-1}, \hat{t}_N) \tag{5.18}$$

where $p < q \implies \hat{t}_p \leq \hat{t}_q$, i.e. $\hat{t}_1 \leq \hat{t}_2 \leq ... \leq \hat{t}_{N-1} \leq \hat{t}_N$.

And the approximation of $\mathbf{X}$, denote as $\hat{\mathbf{X}}$, immediately follows by mapping $\hat{t}$ in $\hat{\mathbf{R}}$ to their associated x in $\hat{\mathcal{T}}$:

$$\hat{\mathbf{X}} = (x_1, x_2, ..., x_{N-1}, x_N) \tag{5.19}$$

where $(x_i, \hat{t}_i) \in \hat{\mathcal{T}}$ for $i \in \mathbb{Z}_N^+$.

To recover the key, the adversary makes a key guess k for every $k \in \mathcal{K}$ and computes the hypothetical intermediate sequence corresponding to key

guess k, denote as $\mathbf{V}_k$:

$$\mathbf{V}_k = F_k(\hat{\mathbf{X}}) \tag{5.20}$$

Under the correct key guess, i.e. $k = k^*$, the adversary would ideally expect $\mathbf{V}_k = \mathbf{V}$ and thus the repetitive structure can be observed in $\mathbf{V}_k$. However, such precise equality is unlikely to hold for noisy real traces; therefore in principle, the distinguishing scores of ordinal distinguishers can be viewed intuitively as the likelihood of the existence of such repetitive structures in $\mathbf{V}_k$ for a key guess k. The more likely $\mathbf{V}_k$ possess such repetitively structure, the more likely $k = k^*$ and vice versa. Following this principle, we propose two ordinal distinguishers in this section and experimentally prove their effectiveness against different target functions later in Section 5.5 and Section 5.6.

5.4.1 Ordinal-Entropy

Observe Equation (5.13) that takes an arbitrary subsequence $\mathbf{v} \subset \mathbf{V}$, the subsequence $\mathbf{v}$ preserves the repetitive structure. Denote $\mathcal{V}$ the multiset of $\mathbf{v}$:

$$\mathcal{V} = \{v : v \in \mathbf{v}\} \tag{5.21}$$

Since most components are repetitive in $\mathbf{v}$, $\mathcal{V}$ is thus expected to have a relatively low entropy. Based on this observation, for a key guess k, we take a subsequence $\mathbf{v}_k \subset \mathbf{V}_k$ and define its corresponding multiset $\mathcal{V}_k k$. We define the sub distinguishing score for $\mathbf{V}_k$ as the negative of the entropy of $\mathcal{V}_k$:

$$d_{\mathbf{v}_k} = -H(\mathcal{V}_k) = \sum_{i=1}^{|\mathcal{V}_k|} p_i \log_2 p_i \tag{5.22}$$

where p_i is the frequency of a target intermediate $v_i \in \mathcal{V}_k$. The negative sign is added so that higher scores indicate greater likelihood, consistent with other existing distinguishers.

Note that $d_{\mathbf{v}_k}$ can be computed for arbitrary $\mathbf{v}_k \subset \mathbf{V}_k$ and the selection of $\mathbf{v}_k$ is trivial. The following factors should be noticed in the selection of $\mathbf{v}_k$:

- Due to the mathematical property of Shannon's entropy, for arbitrary $\mathbf{v}_k$, $d_{\mathbf{v}_k}$ has a strict lower bound which is $-H(\mathcal{O})$. On the other hand, increasing the length of $\mathbf{v}_k$ implies more v to be added into $\mathcal{V}_k$ which effectively reduces the upper bound of $d_{\mathbf{v}_k}$. As a result, the range of $d_{\mathbf{v}_k}$ is reduced, resulting in a smaller distinguishing margin between the correct key and the others.

- Rearranging the elements in $\mathbf{v}_k$ has no impact on $\mathcal{V}_k$, nor thus on $d_{\mathbf{v}_k}$. This means errors within the selected $\mathbf{v}_k$ will be neglected when computing $d_{\mathbf{v}_k}$. Therefore a larger size of $\mathbf{v}_k$ implies more tolerance to noise but at the same time reduces the distinguishing margin between the correct key and the others and vice versa.

The problem of selecting the optimal $\mathbf{v}_k$ remains open at this stage of our work. Without loss of generality, here we propose a robust strategy by selecting two halves of $\mathbf{V}_k$, denoting the lower half $\mathbf{v}_k^L$ and higher half $\mathbf{v}_k^H$, respectively. The corresponding multisets are therefore:

$$\begin{aligned} \mathcal{V}_k^L &= \{v_i : v_i \in \mathbf{V}_k, 1 \leq i \leq |N|/2\} \\ \mathcal{V}_k^H &= \{v_i : v_i \in \mathbf{V}_k, |N|/2 + 1 \leq i \leq |N|\} \end{aligned} \tag{5.23}$$

where v_i denotes the i-th component of $\mathbf{V}_k$.

We then compute their sub distinguishing scores respectively as:

$$\begin{aligned} d_k^L &= -H(\mathcal{V}_k^L) \\ d_k^H &= -H(\mathcal{V}_k^H) \end{aligned} \tag{5.24}$$

And finally summing them as the final distinguishing score for a key guess k:

$$D_k = d_k^L + d_k^H \tag{5.25}$$

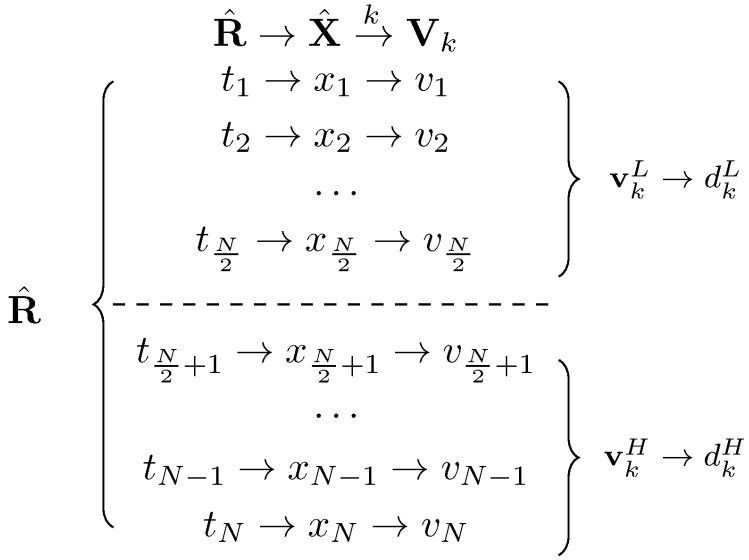


Figure 5.1: Implementation of Ordinal Distinguisher

where the higher D_k, the more likely $k = k^*$.

Referring to Example 3, for the correct key guess $k = 5$, from Equation (5.16) we have:

$$\begin{aligned} \mathcal{V}_5^L &= \{0, 0, 0, 0, 2, 2, 2, 2\} \\ \mathcal{V}_5^H &= \{1, 1, 1, 1, 3, 3, 3, 3\} \end{aligned} \tag{5.26}$$

Hence

$$\begin{aligned} d_5^L &= -H(\mathcal{V}_5^L) = -1 \\ d_5^H &= -H(\mathcal{V}_5^H) = -1 \end{aligned} \tag{5.27}$$

And finally

$$D_5 = d_5^L + d_5^H = -2 \tag{5.28}$$

By comparison, for $k = 0$, we have $D_0 = -3.331 < -2 = D_5$ which suggests that $k = 5$ is more likely to be the correct key than $k = 0$.

Figure 5.1 summarises the implementation of Ordinal-Entropy. An equivalent pseudocode that returns the best key guess k_G is also provided in Algorithm 6.

Algorithm 6 Entropy based Ordinal Distinguisher

function ORDINAL_H($\mathcal{T} = \{(x,t)\}$)
  $\hat{\mathbf{R}} = (t_1, t_2, ..., t_{N-1}, t_N) :=$ Sort($\{t : (x,t) \in \mathcal{T}\}$), where $p < q \implies t_p \leq t_q$;
  $\hat{\mathbf{X}} := (x_1, x_2, ..., x_{N-1}, x_N)$ where $(x_i, t_i) \in \mathcal{T}$ for $1 \leq i \leq N$.
  $\hat{\mathbf{X}}_L := (x_1, x_2, ..., x_{\frac{N}{2}})$
  $\hat{\mathbf{X}}_H := (x_{\frac{N}{2}}, x_{\frac{N}{2}+1}, ..., x_N)$
  for $k \in \mathcal{K}$ **do**
    $\mathcal{V}_k^L := \{v : v \in F_k(\hat{\mathbf{X}}_L)\}$
    $\mathcal{V}_k^H := \{v : v \in F_k(\hat{\mathbf{X}}_H)\}$
    $D_k = -H(\mathcal{V}_k^L) - H(\mathcal{V}_k^H)$
  end for
  return k_G where $D_{k_G} = \max(\{D_k\})$
end function

5.4.2 Ordinal-Variance-of-Positions

Another observation regarding the repetitive structure in Equation (5.13) is that the same target intermediates are "clustered", i.e. they are positioned next to each other in Equation (5.13). The second distinguisher is thus based on the idea of detecting the repetitive structure by testing the dispersion of positions of each target intermediate within each $\mathbf{V}_k$. To this end, we introduce the position function $Pos_{\mathbf{V}_k}(v)$ that returns the set of positions (starting from 1) of target intermediate v in $\mathbf{V}_k$:

$$Pos_{\mathbf{V}_k}(v) = \{i | \text{the } i\text{-th component of } \mathbf{V}_k = v\} \tag{5.29}$$

The dispersion of $Pos_{\mathbf{V}_k}(v)$ is then quantified by the variance of $Pos_{\mathbf{V}_k}(v)$, denoted as $Var(Pos_{V_k}(v))$.

Observe that, due to the repetitive structure of $\mathbf{V}$, for any $v \in \mathbf{V}$, $Pos_{\mathbf{V}}(v)$ returns the set constituting of $|\mathcal{C}_v|$ consecutive natural numbers:

$$Pos_V(v) = \{z, z+1, z+2, ..., z+|\mathcal{C}_v|-1\} \tag{5.30}$$

where z is the position of the first appearance of v in $\mathbf{V}$.

Hence:

$$Var(Pos_{\mathbf{V}}(v)) = (|\mathcal{C}_{v_i}|^2 - 1)/12 \tag{5.31}$$

for any $v \in \mathbf{V}$. Due to the fact that positions are unique natural numbers, it is provable that $Var(Pos_{\mathbf{V}}(v))$ is also the lower bound of $Var(Pos_{\mathbf{V}_k}(v))$ for arbitrary $\mathbf{V}_k$ and v:

$$\forall \mathbf{V}_k, v : Var(Pos_{\mathbf{V}}(v)) \leq Var(Pos_{\mathbf{V}_k}(v)) \tag{5.32}$$

Hence we have:

$$\forall \mathbf{V}_k : \sum_{v}^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}}(v)) \leq \sum_{v}^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}_k}(v)) \tag{5.33}$$

Equation (5.33) implies that $\sum_{v}^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}_k}(v))$ reaches its minimum when $k = k^*$. We thus exploit this property and define the distinguishing score for a key guess k as:

$$D_k = -\sum_{v}^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}_k}(v)) \tag{5.34}$$

where the negative sign is added so that higher scores indicate greater likelihood, consistent with existing distinguishers.

Referring to Example 3, from Equation (5.16) we have:

$$Pos_{\mathbf{V}_5}(0) = \{1, 2, 3, 4\}$$
$$Pos_{\mathbf{V}_5}(1) = \{9, 10, 11, 12\}$$
$$Pos_{\mathbf{V}_5}(2) = \{5, 6, 7, 8\}$$
$$Pos_{\mathbf{V}_5}(3) = \{13, 14, 15, 16\}$$

So that:

$$D_5 = -\sum_{v}^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}_5}(v)) = -6.667$$

which indeed is the lower bound confirmed by Equation (5.33).

By comparison, when $k = 0$, from Equation (5.17) we have:

$$Pos_{\mathbf{V}_0}(0) = \{2, 4, 12, 14\}$$

$$Pos_{\mathbf{V}_0}(1) = \{3, 6, 8, 15\}$$

$$Pos_{\mathbf{V}_0}(2) = \{10, 11, 13, 16\}$$

$$Pos_{\mathbf{V}_0}(3) = \{1, 5, 7, 9\}$$

Thus $D_0 = -79.333 < -0.667 = D_5$. Therefore we conclude that 5 is more likely to be the correct key.

Compared with Ordinal-Entropy, we consider Ordinal-VP more robust as it circumvents the issue of selecting $\mathbf{v}$. We show its corresponding pseudocode in Algorithm 7.

Algorithm 7 Ordinal-VP

function ORDINAL_VP($\mathcal{T} = \{(x, t)\}$)
  $\hat{\mathbf{R}} = (t_1, t_2, ..., t_{N-1}, t_N) :=$ Sort($\{t : (x, t) \in \mathcal{T}\}$), where $p < q \implies t_p \leq t_q$;
  $\hat{\mathbf{X}} := (x_1, x_2, ..., x_{N-1}, x_N)$ where $(x_i, t_i) \in \mathcal{T}$ for $1 \leq i \leq N$.
  for $k \in \mathcal{K}$ **do**
    $\mathbf{V}_k = F_k(\hat{\mathbf{X}})$
    $D_k = -\sum_v^{v \in \mathcal{O}} Var(Pos_{\mathbf{V}_k}(v))$
  end for
  return k_G where $D_{k_G} = \max(\{D_k\})$
end function

5.5 Real Trace Experiments

The real trace experiments are performed on the SCALE board described in Section 2.4. We selected two typical target functions for our experiments which are the modular addition in the unprotected SPARX C reference implementation and the S-Box output in an AES implementation based on AES Furious [139]. Even though the modular addition in SPARX is

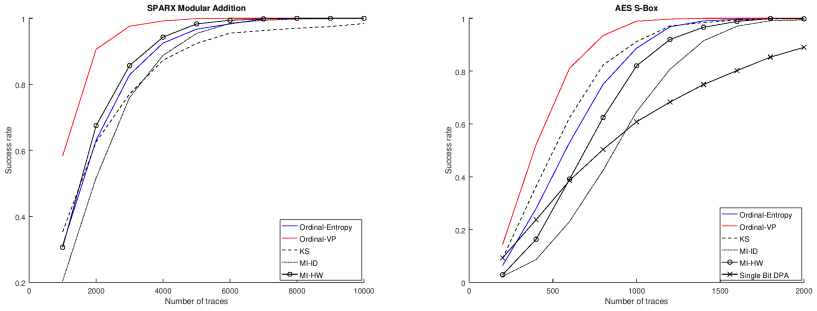
non-injective by nature, we reduced our target to the 4 LSB of the modular sum in Figure 4.1a for a practical enumeration space. For the AES S-Box we used the 4 LSB, i.e.we dropped the 4 MSB as explained in Section 5.2.3, for a balanced number of partition (or intermediate) and traces in each partitions (or intermediates). We compared our distinguishers to the popular ones in the literatures which are Kolmogorov-Smirnov (KS) and Mutual Information with identity and HW models (MI-ID and MI-HW). The success rates in our experiments are evaluated from 1000 repeated experiments. We additionally tested single bit DPA against the LSB of S-Box output in the AES experiments.

Figure 5.2a and Figure 5.2b show the results of our real trace experiments. The success rates of all distinguishers have eventually converged to one except for single bit DPA in the AES experiments. We also noticed that the SPARX modular addition is a relatively more difficult target than AES S-Box in our experiments in terms of required number of traces to achieve the same success rate. Both ordinal distinguishers we described in Section 5.4 have shown to be effective in both experiments. Ordinal-VP turns out to be the most trace efficient distinguisher, with a clear margin ahead of the others in both experiments. In contrast, MI-ID had the worst performance among all generic distinguishers and only outperformed single bit DPA for at least 1000 traces in the AES experiment. MI-HW and KS are generally the second best distinguishers respectively, with Ordinal-Entropy being third in both cases.

5.6 Simulations

In this section we repeat same experiments of Section 5.5 on simulated traces, to demonstrate the performance of these distinguishers under different leakage models and SNR.

Four leakage models are considered in our simulations. The first two represent typical leakage models commonly seen on real devices:



(a) SPARX modular addition (b) AES S-Box

Figure 5.2: Generic distinguishers (and single bit DPA) on SCALE traces

HW Leakage defined as the HW of the target intermediate. The HW model has been widely used to predict hypothetical leakage values in correlation attacks. It is also a typical form of real leakage.

Randomly weighted bits This model assumes that each bit independently leaks its HW weighted by a constant coefficient randomly chosen from $[-1, 1]$. This type of leakage has been reported on some ARM processors [131].

The other two are theoretic leakage models representing some extreme cases:

Binary The leakage is defined as the XOR of all bits of the target intermediate. This leakage model represents the case where the entropy of leakage is minimum[1].

Strongly non-linear The leakage function is defined as an S-Box transformation of target intermediate v:

$$M_D(v) = SBox(v)$$

This model represents the case where the leakage depends on every

[1] Assuming leakage values are uniformly distributed over their range.

bit of v. The S-Boxes are instantiated as those of PRESENT[140] and AES[17] for 4 and 8 bit target intermediates respectively.

For each leakage model, we generated traces with SNR $\in \{2^{-8}, 2^{-5}, 2^{-3}, 2^{0}, 2^{1}\}$.

We noticed that all distinguishers failed to recover the key in the AES simulations using the binary and the strongly non-linear models. We consider that this was due to the drawback of the bit dropping trick: the trick inherently assumes the leakage contributed by the dropped bits can be treated as part of the noise in the bit dropped target function, which does not hold in the above scenarios. All distinguishers were shown successful in all the other scenarios. Similar to the results on real traces, Ordinal-VP seemed to be the most trace efficient one among all successful experiments, outperforming the others with a clear margin.

Similar to the No Free Lunch theorem [141] in case of machine learning, there exist no optimal generic distinguisher in general: the performance of a distinguisher is determined by a combination of factors including device leakage behaviour, SNR and the structure of target function as explained in [125]. Specifically, for generic distinguishes that requires estimating the leakage distribution, e.g. MI, it is proved in [134] that there exists no optimal estimator in general. Further more, for MI and KS to achieve their best performance, the adversary should provide a "good" prediction model to the distinguisher [135] [142]. This indeed contradicts with the premise of a non-profiling attack scenario, as such prediction model cannot be obtained without a profiling stage. In comparison, the fact that Ordinal-VP turned out to be best distinguisher in our representative experiments suggests that Ordinal-VP could arguably be a more robust method than MI and KS as it strips the errors that would be induced by distribution estimation and the inaccuracy of prediction model.

Specifically, it is an intriguing fact that Ordinal-VP outperform MI even the later is provided an accurate power model of HW (Figure 5.3 and

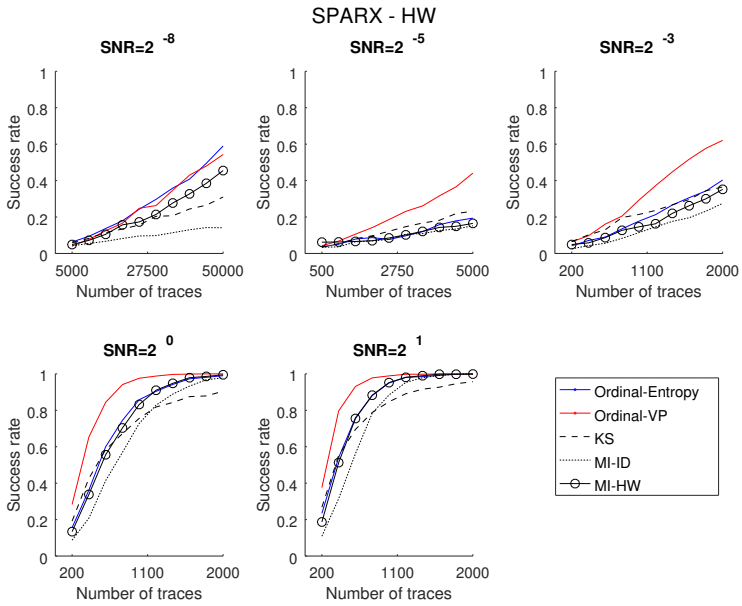


Figure 5.3: Generic distinguishers on SPARX modular addition traces simulated by HW

Figure 5.7). This raises an open question to the conclusion drawn in a recent work [143] where the authors proved the asymptotic optimality of MI under certain circumstances: could there be any factor that has been overlooked in transforming the theoretical results into practice?

5.6.1 Non-uniform Target Intermediate

We additionally performed a group of simulations against a target function that is altered from SPARX ARX-Box, named XOR-then-multiply, where the target intermediate v is defined as:

$$v = (x \oplus \alpha) * (y \oplus \beta) \mod 2^n \tag{5.35}$$

where (α, β) are the keys, (x, y) the inputs and n the operand size.

We consider XOR-then-multiply an interesting target on account of the following properties:

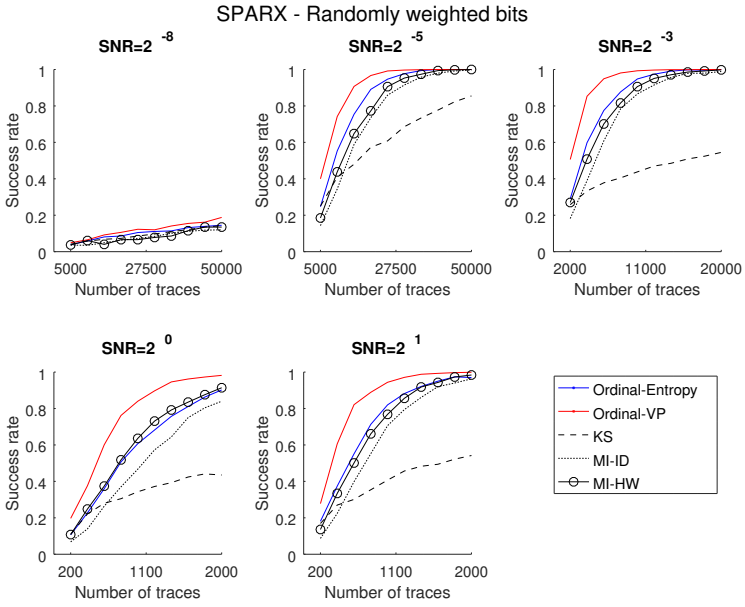


Figure 5.4: Generic distinguishers on SPARX modular addition traces simulated by randomly weighted bits leakage

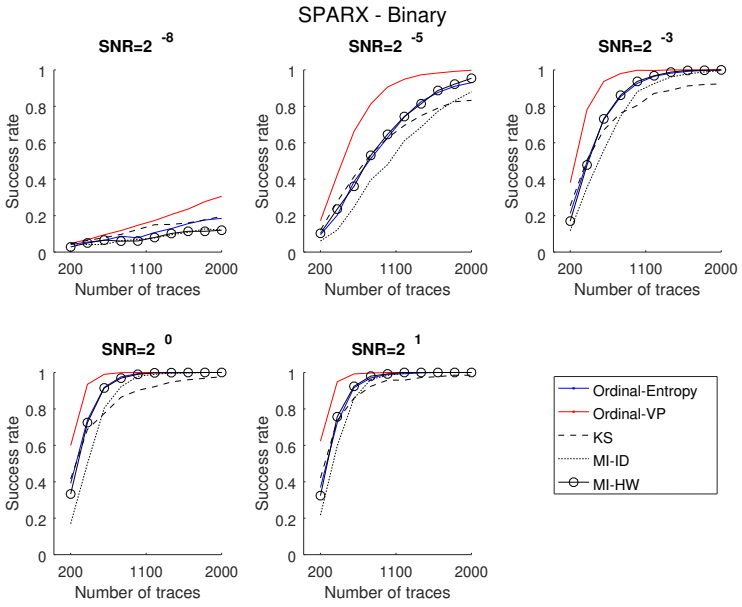


Figure 5.5: Generic distinguishers on SPARX modular addition traces simulated by binary leakage

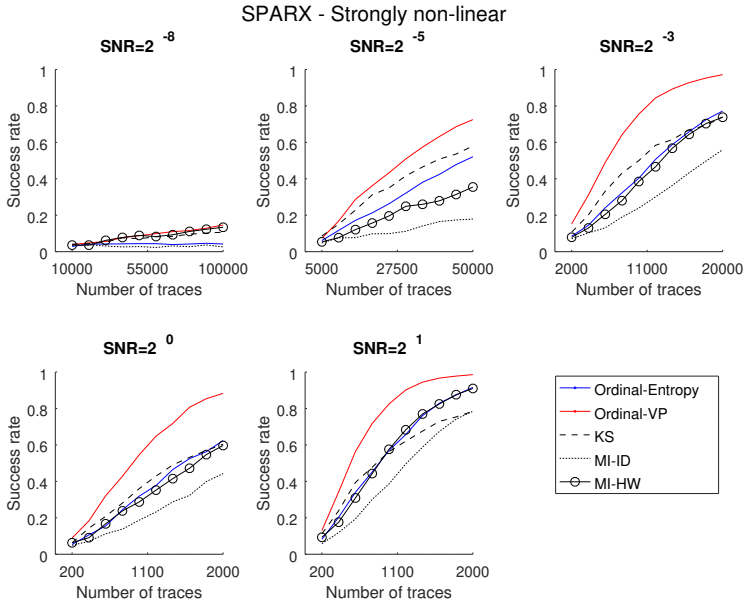


Figure 5.6: Generic distinguishers on SPARX modular addition traces simulated by strongly non-linear (PRESENT S-Box) leakage

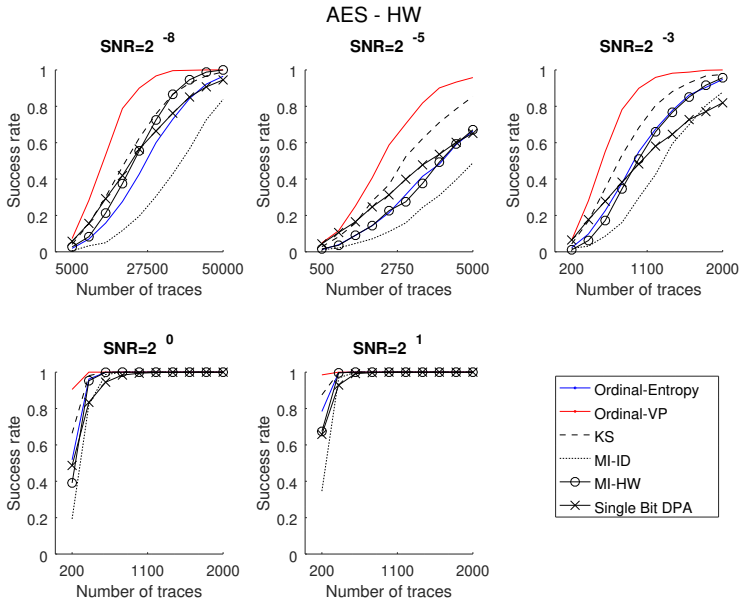


Figure 5.7: Generic distinguishers using 4 LSB and single bit DPA using LSB on AES S-Box traces simulated by HW leakage

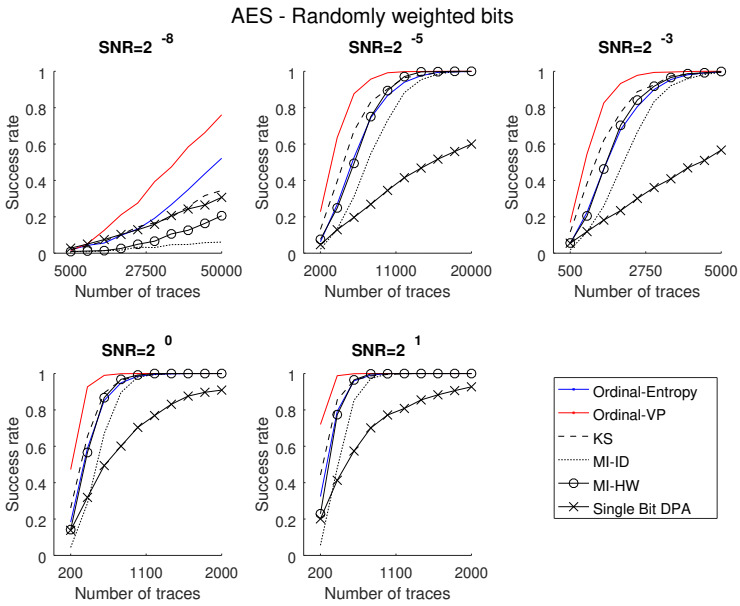


Figure 5.8: Generic distinguishers using 4 LSB and single bit DPA using LSB on AES S-Box traces simulated by randomly weighted bits leakage

- Unlike the SPARX and AES experiments, the output of XOR-then-multiply is not uniformly distributed over its range.

- It holds that the distribution of v is independent of the keys (α, β).

The generic distinguishers were applied in their original forms without any modification. All distinguishers failed against the binary leakage model, and Figure 5.9 to Figure 5.11 show the success rates in other scenarios. KS emerged as the most effective distinguisher in these experiments, especially in the low SNR setups. We suppose this is due to the fact that the definition of th KS distinguisher inherently normalised the sub scores for each partition and thus the fact that v is not uniform over $\mathcal{O}$ has been taken into account. As the SNR was increased, all other distinguishers started to recover, with Ordinal-VP raised to the second best distinguisher after KS.

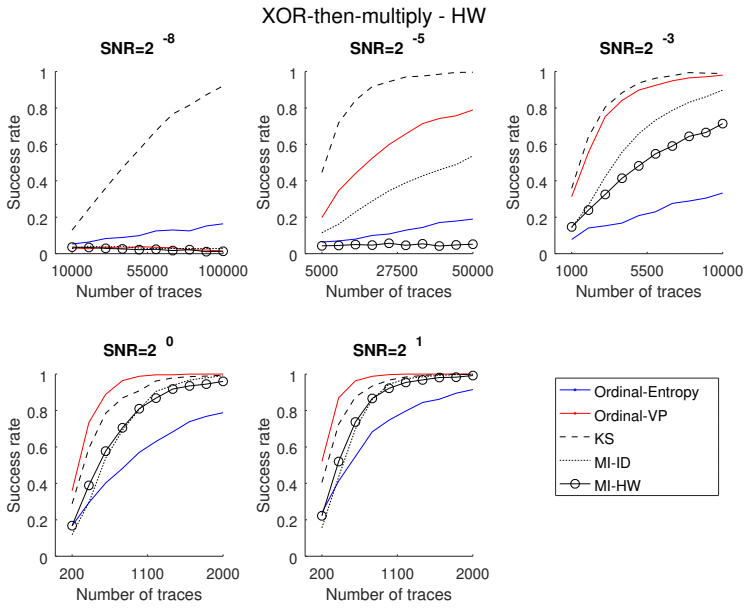


Figure 5.9: Generic distinguishers on XOR-then-multiply traces simulated by HW leakage

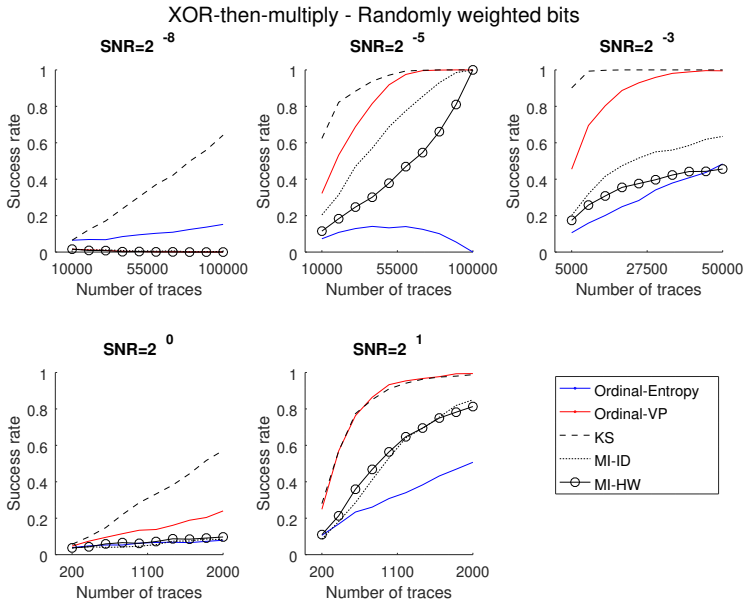


Figure 5.10: Generic distinguishers on XOR-then-multiply traces simulated by randomly weighted bits leakage

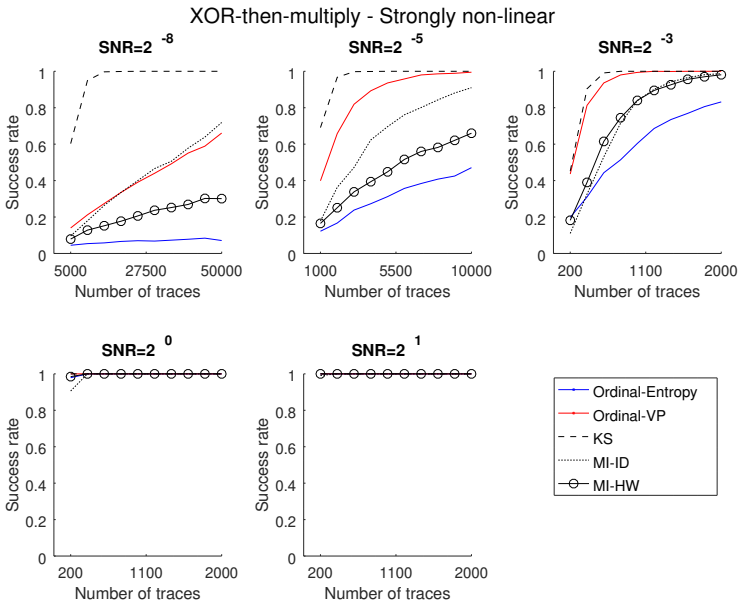


Figure 5.11: Generic distinguishers on XOR-then-multiply traces simulated by strongly non-linear (PRESENT S-Box) leakage

5.7 Summary

In this chapter, we proposed the idea of constructing generic distinguishers exploiting the ordering of leakage. We further instantiated two distinguishers based on this idea, namely Ordinal-Entropy and Ordinal-VP. We then tested the distinguishers on two target functions in SPARX and AES, and showed that Ordinal-VP experimentally has the best trace efficiency using both real and simulated traces. However, when applied to the target function XOR-then-multiply, where the target intermediate is not uniformly distributed over its range, KS seemed to be the best distinguisher.

Chapter 6

Exploiting Package Feature

6.1 Introduction

Connectivity finds the greatest revolution in IoT devices which distinguishes them from general embedded devices. In IoT applications, devices are no longer stand-alone: they cooperate with each other within a network which eventually connects to the Internet. Whereas the typical TCP/IP network stack produces significant overhead to achieve quality of service for applications that are based on it, the nature of many IoT "things" is such that a full implementation of the protocols would not be practical. Often 'things' are sensors, which are devices that have to function on little resources (most importantly power). Thus a whole host of new networking protocols have been developed over the years to cater for such resource constrained devices: 6LoWPAN is the 'tiny' version of IPv6, UDP tends to be used instead of TCP, DTLS can be used for end-to-end security (or one can directly invoke 802.15.4 security which is part of 6LoWPAN), and finally CoAP(s) is the replacement for HTTP(s). Thus there are two options (802.15.4, and DTLS) to secure communications between the 'things' and a server/gateway.

Securing IoT applications is a difficult task. In addition to the side channel attacks that have been well understood as severe security threats against

embedded devices [30], many other attacks use protocol level information (such as packet lengths, types of packets or protocol messages) to recover information about plaintexts, devices in the network, or the network itself. There exists a considerable body of work in the context of conventional, i.e. HTTPs over TCP/IP network, but not much literature we have found with respect to the security implication of these attacks for 6LoWPAN. This is the gap that we address with this work. We stipulate that most of these attacks may still be applicable, as we intend to demonstrate in this chapter. As of writing the thesis, this is so far the first work that explores the feasibility of traffic analysis techniques over 6LoWPAN networks to our knowledge.

This chapter is structured as follows: after reviewing some relevant attack paths for HTTPs over TCP/IP in the following subsection, we provide a brief introduction to the necessary protocol and network features in Section 6.2.1. We discuss the impact of packet length leakage in Section 6.3, followed by an analysis of the response time leakage in Section 6.4. We summarise our work in Section 6.5. This work was done in conjunction with E. Oswald and T. Tryfonas. It has been published in:

Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Exploring Potential 6LoWPAN Traffic Side Channels'. In: *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. EWSN ’18. Madrid, Spain: Junction Publishing, 2018, pp. 270–275. ISBN: 978-0-9949886-2-1. URL: `http://dl.acm.org/citation.cfm?id=3234847.3234911`[9]

As the main author I was responsible for all main aspects of the work. This includes investigating the related protocol standardisation documents together with their implementations in Contiki OS[13]. I was also responsible for proposing the novel idea of an application fingerprinting attack as well as conducting all the experiments in this work.

6.2 Preliminaries

6.2.1 A Typical IoT Protocol Stack

There are large number of protocols which have been proposed for different IoT applications adapting to various requirements. For example, some smart houses simply use WiFi for connectivity, and VANETs[1] may adopt DSRC[144].

In this chapter we focus on 6LoWPAN[22] which is based on the 802.15.4[16] standard. These standards are designated for constrained environments such as Wireless Sensor Networks, but other competing standards exist at different layers. Bluetooth Low Energy(BLE)[145] is a strong competitor to 802.15.4 as well as the LiFi[146] technology. Zigbee[21] was originally intended as a collective protocol over 802.15.4 but it has been recently adapted to an IP-based network in ZigbeeIP[147]. The RIME stack[148] proposes a set of non-layered primitives over 802.15.4 but it is likely to be phased-out due to the lack of of interoperability with the TCP/IP protocol stack.

6LoWPAN thus is the most popular standard for low power networks, and thus it is supported by several competing IoT Operating Systems, including Contiki OS[13], OpenWSN[24], FreeRTOS[149] and the recent RIOT[150]. We chose Contiki OS for our experiments because it is easy to customise.

With regard to the aspect of protocol design, the recent paper [151] summarised some known flaws of 6LoWPAN, including its susceptibility to the Fragmentation Attack[152], Sinkhole Attack[153], Hello Flood Attack[154], Wormhole Attack[155] and Blackhole Attack[156]. In addition, [157] reported certain problematic designs in 802.15.4 security[16]. However we do not discuss further these particular design flaws as they touch on a different aspect of the security issues in 6LoWPAN compared to what we address in this chapter.

[1] Vehicular ad hoc networks

Physical	802.15.4
Link	
Network	6LoWPAN
Transmission	UDP
	DTLS*
Application	CoAP / CoAPs*

Table 6.1: Protocol stack for our experiments (* are optinal)

6.2.2 Our Experimental Network

Our experimental network is constructed using two different devices; a TelosB[15] and a CC2538[14]. The TelosB is a low cost sensor powered by an MSP430 with an AES co-processor. It represents typical low-end devices. The CC2538 is the high end device powered by an ARM Cortex-M3 with multiple cryptographic processors including AES, RSA, SHA-2 and ECC, suggesting that it is suitable to develop secure applications.

Both devices are supported by the Contiki OS. We adopted the default settings of the Contiki OS, except for enabling 802.15.4 security[16] for some experiments. Note that the Contiki MAC[158] is chosen by default over TSCH[159]. For Layer 4[160] and above protocols, we went with the widely accepted combination of CoAP[161], and DTLS[162](optional) over UDP[163][2]. Table 6.1 summarises our choice of protocol stack.

802.15.4 and DTLS

In our setting, there are two standards available for packet encryption: 802.15.4 security[16] and DTLS[162]. 802.15.4 security is provided by the noncoresec[164] API, which implements 802.15.4 authenticated encryption with AES-128 CCM*[165] using a hard-coded key shared by the whole 6LoWPAN network. We chose tinyDTLS[166] as library for the DTLS protocols, because it provides a minimum DTLS implementation that supports two ciphersuites which are TLS_PSK_WITH_AES_128_CCM_8[167] and

[2]CoAPs is equivalent to CoAP over DTLS.

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8[167] respectively. Evidently, they both utilise AES-128 CCM* as the packet encryption method.

6.3 Exploiting Packet Length Information

As our brief survey of traffic analysis via exploiting packet lengths showed in Section 6.2, the packet length has proven to be a powerful side channel for the classical Internet protocols. It is worth noting that this side channel is 'noisy' in the classical Internet setting; websites or web applications in this setting typically feature advertisements, which impact on packet lengths; TCP/IP allows to fragment packets and then reassembles them, a feature which is not presented in UDP. Thus, due to the nature of UDP, exploiting the packet length as a side channel should be easier in the IoT setting.

Clearly then, any web-application-style implementations involving an IoT device will be extremely vulnerable to attacks such as [32]. In the absence of this scenario for state-of-the art IoT applications, it still sends a cautionary warning to developers; binary responses (e.g. 'yes' vs. 'no', or 'on' vs. 'off') must always be coded via a binary variable and not via strings because these will have different lengths, which are directly visible via the packet length.

In the remainder of this section we highlight further problems that arise if packet lengths leak information.

6.3.1 Distinguishing ICMP Messages

The Internet Control Message Protocol(ICMP)[168] performs the management tasks in a network, such as link establishment and routing information exchange. As explained before, we utilise the open source system Contiki, which supports a (sub)set of the ICMP standard (we list the supported ICMP messages further below). Many ICMP messages are ideal for network discovery and exploration, although the purpose of ICMP is to send

error messages to the source IP address if standard IP packets fail to be transmitted correctly.

- **DAG Information Object (DIO)**
  DIO contains the 6LoWPAN global information. It could be periodically broadcasted for network maintenance, or unicasted to a new joining node as a reply to DIS (see below).

- **DAG Information Solicitation (DIS)**
  DIS is sent by a newly started node to probe any existing 6LoWPANs. A DIO would be replied if the DIS is received by any neighbour nodes.

- **Destination Advertisement Object (DAO)**
  DAO is sent by a child node to its precedents (The 6LoWPAN DODAG topology is defined in [169]) to propagate its routing information.

- **Neighbour Solicitation (NS) and Neighbour Advertisement (NA)**
  NS and NA are the ARP replacement in IPv6, where NS queries a translation and NA answers one. In addition, they are also used for local link validity checks.

- **Echo Request and Echo Response (PING)**
  Echo Request and Echo Response are well known as the PING packets. They are mostly used for diagnostic purposes, such as connectivity test or Round Trip Time (RTT) estimation. Echo Request may contain arbitrary user defined data and Echo Response simply echoes its corresponding request.

Generally, ICMP messages can be protected by either using the secure ICMP messages as described in [168], or relying on the lower layer encryption provided by 802.15.4. Contiki OS does not have the former implemented, hence 802.15.4 security is currently the only option. We simulated a

	Packet Size (bytes)	MAC Destination
DIS	85	broadcast
DIO	118/123	broadcast/unicast
DAO	97	unicast
NS	87	broadcast/unicast
NA	87	unicast
PING	$101 + x$	unicast
UDP Multicast	$85 + x$	broadcast
UDP Unicast	$107 + x$	unicast

Table 6.2: 6LoWPAN packet features

6LoWPAN network with 802.15.4 security enabled (with strongest encryption and authentication). We configured the nodes to also generate random UDP packets. Despite the fact that all ICMP messages were encrypted, our experiments show that several ICMP messages can be identified by their packet size and MAC destination. Table 6.2 summarises the packet features. The value x denotes the size of user defined data in bytes.

Among the unicast packets, PING and UDP have at least 101 and 108 bytes[3]. Therefore, DAO can be uniquely identified as the shorter unicast packet of 97 bytes. For the same reason NA and unicast NS can also be distinguished from other packets by filtering packets of 87 bytes. Considering that NA is sent as a response to NS according to the protocol, one can always identify the first being NS and second being NA.

Similarly, unicast DIO can be identified as the 123 bytes packet followed by DIS, where the latter has a unique 85 byte size. However, there is a potential of false positive induced by PING or UDP packets with user defined data crafted to have the same packet length[4]. PING could be recognised by its pair-wised appearance, as the response would have nearly the same meta data as the original request, except the exchanged source and destination. For broadcast packets, DIS can be easily identified by its unique 85 bytes

[3]PING can be sent without user defined data and UDP packets requires at least 1 byte.
[4]22 bytes for PING and 16 bytes for UDP.

packet size. Others like broadcast NS can be identified by the followed characteristic NA response; and packets of 118 bytes that are periodically broadcasted are likely to be DIOs.

In summary, among all the packets, DAO, NA, NS, DIS can be identified with certainty. DIO and PING cannot be identified with certainty but they both have significant characters. Notice that the above contained all ICMPv6 messages supported by Contiki; therefore UDP packets can be reversely filtered, although in some cases they get mixed with DIO and PING.

Although leakage in ICMP messages does not directly lead to any breach of application data, it would still be harmful by providing the adversary with information about the state of the network, including which nodes recently joined etc. Specifically DAO is always sent from a child to its parent and can be uniquely identified; therefore together with MAC addresses the adversary may exploit it to draw a graph that shows the parental relations in the network. In addition, this information can also be exploited by attacks as in [170].

6.3.2 Distinguishing Different Devices

In the classical Internet world, ICMP has been well known for its use for OS fingerprinting[171]. In the case of the IoT, this could be possible as well (as different OS support different subsets of ICMP), however an additional attack vector exists. This is because different IoT devices have different hardware limitations or drivers. We noticed that our TelosB[15] discards all packets exceeding 127 bytes[5] whereas our CC2538 handles packets even up to 160 bytes. Therefore an adversary can immediately rule out TelosB whenever a packet larger than 127 bytes processed by the target.

[5]MTU specified by 802.15.4 standard.

6.4 Exploiting Response Time Information

The response time is another major feature that has been previously exploited in Internet traffic analysis attacks. Like in the case of exploiting packet lengths, we would expect that the same attacks (as in the classical Internet setting) can be applied to 6LoWPAN traffic. Indeed, like in the previous section, we would expect that they will work even better because the accuracy of timing measurements can be greatly improved for 6LoWPAN traffic as there are fewer noise sources in the traffic. Since the devices are physically close to each other and uses RF to communicate, the adversary can remove the RTT noises by measuring the packets on the server side. Also the constrained performance of the devices also gives a better resolution of the execution time.

6.4.1 Distinguishing Different Sensors

The first application of timing analysis that we describe is to distinguish between different sensors that are accessed on a device. For this purpose we set up an experiment on a classic device , namely CC2538, that has three on-board sensors: Vdd, temperature, and an Ambient Light Sensor (short ALS). We access these via CoAP[161], a protocol designed for constrained devices that provides an universal interface for accessing resources. CoAPs is the secure version which stands for CoAP with DTLS.

Due to the different physical characteristics of the sensors, there could be a variance of time that is required for reading the measurements. We investigated whether such variances could be observed through the packet response latency. If this was the case, then an adversary could learn the nature/purpose of sensors on a network by observing their response time.

We thus set up an experiment on CC2538, using all three sensors from "cc2538-demo". We used CoAP from the "er-rest-example" in the Contiki OS source code, as there is no CoAPs implementation available. Although DTLS

	Average (ms)	Range(ms)
Vdd	9.622	[9.388, 10.318]
Temperature	9.835	[9.525, 10.318]
ALS	11.651	[11.338, 12.031]

Table 6.3: CoAP response latency for sensor readings on CC2538

processing would definitely have an impact on the response latency, we argue that such impact would be independent to the sensors being accessed; hence similar result can be equally expected for CoAPs. We carefully controlled other factors, including URIs, data representation and code flow, to be uniform for all three sensors in order to guarantee a controlled environment.

Table 6.3 summarises the result. It shows that ALS takes about 2ms longer and hence can be easily distinguished. Vdd and temperature have stronger overlapping distributions, and thus are more difficult to distinguish. Nevertheless, these results confirm our hypothesis: different sensors have different latencies and these leak through the response time. An adversary whom is interested in finding out information about devices on a network might thus be able to match the (known) behaviour of 'interesting' sensors to what they observe on the network. We remark that this could be useful even in the setting where the sensors transmit their data unencrypted; after all they might return only some reading without a unit of measurement; thus seeing their return data might not as such reveal their nature.

6.4.2 Distinguishing Different Devices

As we observed before, different devices have different underlying hardware and thus different computational power. This implies that there could be the potential that different devices take different amounts of time to process the same message. Because ICMP messages are standardised, they are particularly suitable for this purpose. Among the different ICMP messages, PING is especially ideal for two reasons:

	CC2538	TelosB
Average(ms)	9.56	17.03
Range(ms)	[9.16, 10.06]	[16.49, 17.68]

Table 6.4: PING response latency

1. It is mandatory in the ICMP standard.

2. It only swaps the source and destination address of the packet; thus minimises different code path in protocol processing.

Table 6.4 shows the PING response latency on CC2538 and TelosB. The result confirms that these devices can be distinguished by PING response latency.

6.4.3 Distinguishing Programs

We remarked before that the functionality of a sensor is potentially valuable information. For instance some sensors might be predominantly passive, e.g. they might read the temperature and report it back periodically, whereas some sensors might control something upon receiving commands. Thus, knowing the functionality enables an adversary to make (more) sense of the observed traffic in the network. This could be done if a 'fingerprint' could be produced for different programs. From an adversary's perspective a positive result would imply that they could 'fingerprint' products which are on the market and thus use this information to infer what program is running on a target device.

To illustrate why this might work, we now look at Figure 6.1. It illustrates two sensors receiving the same service request. In our example, at the time of receiving the request, Sensor Node 1 was idle and hence responded immediately, whilst Sensor Node 2 postponed the request for reading a sensor. Clearly, the response time on Sensor Node 2 would appear longer than that of Sensor Node 1.

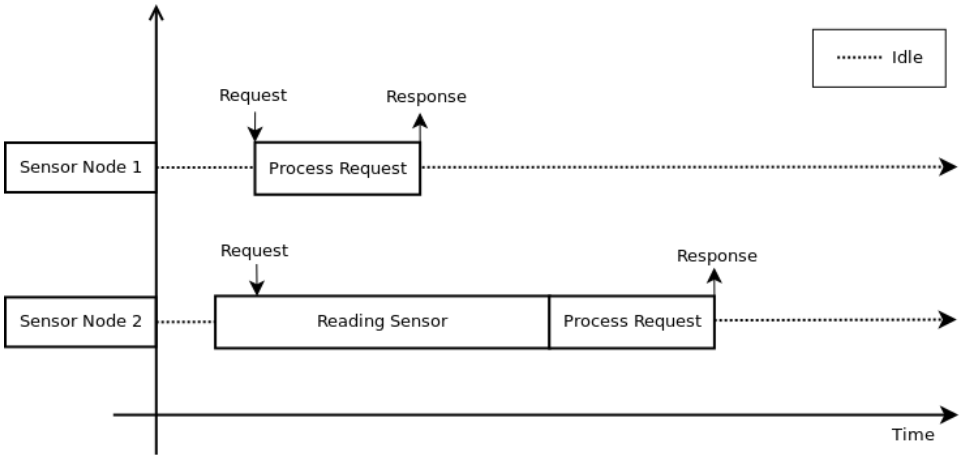
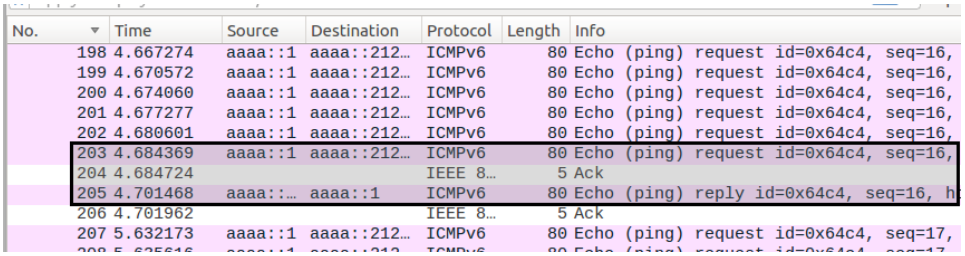


Figure 6.1: Variations in response time

In real life, most sensors are programmed in a loop; therefore the same code fragments are repeated through the life time of a sensor. Each code fragment takes a different mount of time to execute and hence the response times vary. This behaviour could be statistically analysed and the resulting distribution could be stored as a 'fingerprint' .

For this fingerprinting scenario, we must assume the adversary has the pre-knowledge of potential programs and can fingerprint them (or that they have access to a database that contains this information). To identify an unknown program running on target sensor, the adversary collects a new fingerprint and then matches it to available fingerprints. Clearly, to effectively launch the attack, the adversary needs to be able to send the request to a targeted sensor (requests with short predictable processing time are preferable as they induce less noise).

In practice, the request can be instantiated by several messages defined in the sensor network protocols. PING is exceptionally ideal as it is mandatory in the ICMP standard[172] and has only negligible computation. Other options are Heartbeat in DTLS[173], Reset in CoAP[161], etc.



No.	Time	Source	Destination	Protocol	Length	Info
198	4.667274	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
199	4.670572	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
200	4.674060	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
201	4.677277	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
202	4.680601	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
203	4.684369	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=16,
204	4.684724			IEEE 8…	5	Ack
205	4.701468	aaaa::…	aaaa::1	ICMPv6	80	Echo (ping) reply id=0x64c4, seq=16, h
206	4.701962			IEEE 8…	5	Ack
207	5.632173	aaaa::1	aaaa::212…	ICMPv6	80	Echo (ping) request id=0x64c4, seq=17,

Figure 6.2: Example PRI

Extracting Fingerprints

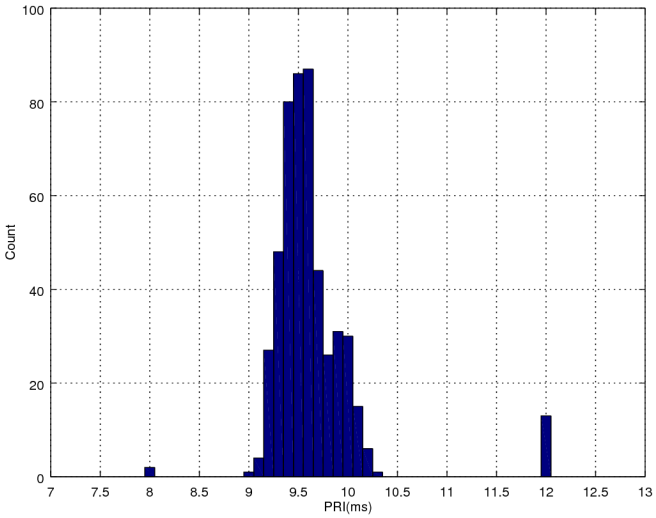
We explored the feasibility of fingerprinting programs on an CC2538 running Contiki OS by using the PING command.

Figure 6.2 shows an example of captured packets. Contiki MAC[158] sends duplicated PING requests. The response time, which refers to PING Response Interval, PRI, is defined to be the time between a PING response and its last paired request. The highlighted Packets 205 and 203 shows such an example.

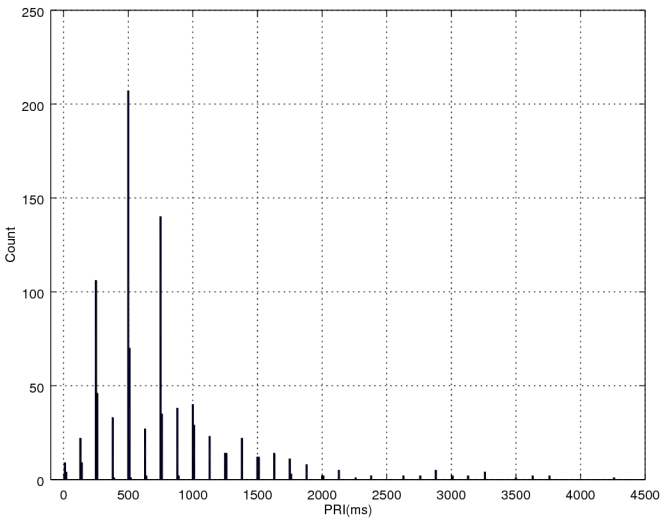
Figure 6.3a shows the histogram of PRIs collected on the "helloworld" example from Contiki OS. Values $\geq$12ms are collected at 12ms. The result shows that most PRIs are clustered around 9.5ms which consists with our result in Table 6.4. The majority, roughly ranged [9.0, 10.3]ms, corresponds to the usual response time as depicted by Sensor Node 1 in Figure 6.1.

We further plotted the upper outliers, mostly ranged [12, 2000]ms, in Figure 6.3b. Although we were not be to able to identify the exact cause of such delay, we suppose these outliers correspond to the extended response time as depicted by Sensor Node 2 in Figure 6.1. The distribution described by Figure 6.3b is the fingerprint of the "helloworld" example.

The result in Figure 6.3 shows a clear gap between the usual PRIs and extended PRIs. In fact, experimented applications showed the same property. This implies that an adversary can easily draw a threshold by observing the whole PRI distribution and then filter out the fingerprint. In our experiments



(a) PRIs of helloworld



(b) PRIs outliers of helloworld

Figure 6.3: helloworld PRIs

the threshold was set to 12ms but any other values within the gap would also work.

We collected the fingerprints for three programs taken from the Contiki OS examples:

broadcast This program periodically broadcasts a constant message.

powertrace This program records the power consumption and broadcasts a constant message.

Sensorpayload This program is based on the "er-rest-example" embedded together with sensor accesses taken from "cc2538-demo". It captures a real case scenario where three different sensors, namely Temperature, Vdd and ALS, are being accessed through CoAP.

Specifically for "Sensorpayload" we collected fingerprints for 8 different scenarios where different sensors are being accessed. For each program we independently collected 2 fingerprints for comparison.

Table A.1 summarises the total 20 fingerprints we collected for the experiment. The source code for the device is published in [174].

Fingerprint Matching

During the experiments we realised that most of the fingerprints do not adhere to common distributions; therefore we used a non parametric test, the Kolmogorov-Smirnov Distance[175], as our test statistic. This is a well understood statistic with previous uses in side channel analysis[176]. Table B.1 summarises the relative KS distances computed on each pair of fingerprints in our experiments.

Even though fingerprints collected on the same application were rejected by the Kolmogorov-Smirnov Test, we noticed that their KS distance still tends to be smaller comparing to fingerprints collected on different programs, as the **bold** cells marked in Table B.1.

By adapting our distinguisher to utilise the minimum KS distance, we were able to identify 13 out of 20 fingerprints successfully. The 'overlapping' fingerprints are mainly due to the "Sensorpayload" program, which access different sensors, but otherwise has identical program code. Thus we did expect that the different instantiations of it would lead to very similar fingerprints.

6.5 Conclusion

In this chapter we explored the use of packet lengths and response times, which are protocol level side channels, as means to recover information about IoT 'things'. We do this experimentally, which we base on two extremely popular devices running on a popular open source OS, with a typical stack of protocols. Our results show that it is possible (in principle) to recover information about a device and its function (i.e. the hardware and the software that runs on it) via inspecting encrypted traffic that it produces. We also point out that ICMP messages can be distinguished from each other despite the use of encryption.

In order to mitigate the leakage that is given by packet lengths, previous works [48] recommend the usage of different padding schemes such as pad to fixed length, threshold padding and padding to MTU, etc. We echo this recommendation. Whilst padding to MTU is considered inefficient for the Internet, it is in fact highly appropriate for 6LoWPAN because:

- It completely hides the length of original plaintext.
- 6LoWPAN has only a low MTU of 127 bytes; therefore the overhead is acceptable.
- It induces negligible computational overhead.

With regard to the leaking information about the device or OS, we suggest

strictly applying the standard MTU to eliminate the differences in drivers. Although there is a potential of performance downgrade, it will also improve the compatibility among different devices.

In order to mitigate the leakage given by response times, the natural countermeasure is to write time-constant code, which is known to be notoriously difficult. But two approaches are available to a software developer:

- Randomly delay the response. This essentially adds noise to the measurements of the adversary.
- Use a threshold response time, i.e. a request is either responded at a predefined time or not responded at all.

Within the context of 6LoWPAN the second method is recommended as most 6LoWPAN applications would tolerate missing packets and timers are available on most platforms. However, the threshold must be carefully chosen to preserve the functionality of the 6LoWPAN applications.

Chapter 7

Concluding Remarks

This thesis set out to examine security issues in IoT applications from various aspects related to side channel attacks. Three vulnerabilities have been identified and each of them could pose a great threat to the security of IoT applications.

The case study in Chapter 3 revealed a flawed design of RNG in a popular device used by many IoT projects. Our result could be significant to any secure IoT application built on that device as any cryptographic implementation cannot be be secure when the underlining RNG is predictable to the adversary. The importance and necessity of a properly designed RNG should be reflected on in any secure IoT design in the future.

The threat of DPA in IoT applications was the second focus in this thesis. This part of the work begins in Chapter 4 with an inspection towards the "inherent" side channel resilience of ARX ciphers. Whilst part of our results confirmed that the perceived vulnerable instruction, modular addition, is hard to attack by conventional DPA techniques, the perceived secure operations could still be vulnerable when implemented inappropriately. Although a chosen message DPA strategy dedicated to modular addition was also proposed, the result was negative on real devices partly due to mismatched prediction of leakage models.

Chapter 5 studied generic distinguishers which are specifically useful against devices with unknown leakage models in DPA attacks. A novel methodology of constructing generic distinguishers was proposed in our work and one of our new distinguishers has demonstrated superior performance in several experiments. The results in Chapter 4 were also extended by this study, as generic distinguishers were found effective against modular addition.

The last part of the thesis moved on to side channel attacks at a higher level in IoT applications in Chapter 6. This research explored the potential of traffic analysis attacks being ported to the IoT scenarios. The outcome was concerning, as these attacks proved to be only more effective in these settings.

Indeed, building a secure IoT application is difficult, not only because of the nature of constrained resources, but also the fact that not much designers would have the awareness of all the security threats at different levels. Sadly, probably inherited from the "no-cross-layer-cooperation" tradition of Internet, many people working in related fields are reluctant to revisit the problems from others' perspectives.

For instance, it is not hard to imagine the cause of the bad RNG design in Chapter 3: the designers very likely made the choice of reusing existing components over a dedicated RNG simply for a lower cost, regardless of the damage it could cause and the fact that the product is advertised for security purposes. A similar situation can be seen for the protocols we inspected in Chapter 6 where designers working in the communication aspect tends to prohibit any attempt that may cause an overhead, including some of the effective countermeasures against traffic analysis. After all, many security loopholes could easily be fixed at earlier stages if they got noticed by people from the relevant areas of expertise.

7.1 Future Work

Various aspects of the side channel security in IoT have been studied in this thesis and each of thesis subject could be further extended.

The case study of RNG in Chapter 3 was purely based on black box experiments. Since the same design was also adopted by other products in the series, a thorough study of how the circuits are designed would give a better understanding to the addressed problem. We also noticed that the later models in that series (e.g. CC2650) have provided a dedicated RNG which might also worth to be thoroughly studied.

For the popularity of ARX gained in recent years, we hope our results in Chapter 4 would provide the cryptographic community a better understanding to the side channel related properties of ARX ciphers when proposing new designs. It is also an interesting research topic to see whether these results can contribute in designing side channel countermeasures for ARX in practice.

The Ordinal distinguishers may have the most open questions among this thesis. For example, could there be a better statistical method to detect the distinctive repetitive structure than our straightforward approaches? How should we address the issue of non-uniformly distribute intermediates as in the case of XOR-then-Multiply? Extending Ordinal distinguishers to multi dimension also seems to be an interesting open problem.

The leakages described in Chapter 6 mostly utilised a proof-of-concept set up with the ICMPv6 protocol which does not have any immediate threat to the upper layer applications. Naturally a next question is then what are the issues if the same attacks are carried on to upper layer applications where the targeted data are more sensitive.

Bibliography

[1] https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#1c60ea0292d5.

[2] http://ijpr.org/post/internet-things#stream/0.

[3] http://www.libelium.com/libelium-smart-world-infographic-smart-cities-internet-of-things/.

[4] https://www.forbes.com/sites/joemckendrick/2016/11/13/with-internet-of-things-and-big-data-92-of-everything-we-do-will-be-in-the-cloud/#16ed62ec4ed5.

[5] A. Freier, P. Karlton and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. RFC6101. Aug. 2011. URL: `http://tools.ietf.org/rfc/rfc6101.txt`.

[6] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC5246. Aug. 2008. URL: `http://tools.ietf.org/rfc/rfc5246.txt`.

[7] https://csrc.nist.gov/projects/lightweight-cryptography.

[8] Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Cryptographic randomness on a CC2538: A case study'. In: *IEEE International Workshop on Information Forensics and Security, WIFS 2016, Abu Dhabi, United Arab Emirates, December 4-7, 2016*. IEEE, 2016, pp. 1–6. ISBN: 978-1-5090-1138-4. DOI: `10.1109/WIFS.2016.7823912`. URL: `https://doi.org/10.1109/WIFS.2016.7823912`.

[9] Yan Yan, Elisabeth Oswald and Theo Tryfonas. 'Exploring Potential 6LoWPAN Traffic Side Channels'. In: *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. EWSN ’18. Madrid, Spain: Junction Publishing, 2018, pp. 270–275. ISBN: 978-0-9949886-2-1. URL: http://dl.acm.org/citation.cfm?id=3234847.3234911.

[10] Alan Graham. *Statistics / Alan Graham.* eng. London: Teach Yourself Books, 1999. ISBN: 0340753587.

[11] John R Pierce. *An introduction to information theory: symbols, signals and noise*. Courier Corporation, 2012.

[12] Stefan Mangard. 'Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness'. In: *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*. Ed. by Tatsuaki Okamoto. Vol. 2964. Lecture Notes in Computer Science. Springer, 2004, pp. 222–235. ISBN: 3-540-20996-4. DOI: 10.1007/978-3-540-24660-2_18. URL: https://doi.org/10.1007/978-3-540-24660-2_18.

[13] URL: http://http://www.contiki-os.org/.

[14] URL: http://www.ti.com/product/CC2538/description.

[15] Online: http://www.willow.co.uk/html/telosb_mote_platform.php.

[16] IEEE 802.15.4 Working Group. *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. Tech. rep. IEEE 802.15.4 Working Group, 2006, 0_1–305.

DOI: 10.1109/ieeestd.2006.232110. URL: http://dx.doi.org/10.1109/ieeestd.2006.232110.

[17] PUB FIPS. '197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, US Department of Commerce (November 2001)'. In: *Link in: http://csrc. nist. gov/publications/fips/fips197/fips-197. pdf* (2001).

[18] Quynh H Dang. *Secure hash standard*. Tech. rep. 2015.

[19] K. Moriarty, B. Kaliski, J. Jonsson et al. *PKCS #1: RSA Cryptography Specifications Version 2.2*. RFC8017. Nov. 2016. URL: http://tools.ietf.org/rfc/rfc8017.txt.

[20] A. Langley, M. Hamburg and S. Turner. *Elliptic Curves for Security*. RFC7748. Jan. 2016. URL: http://tools.ietf.org/rfc/rfc7748.txt.

[21] Zigbee Alliance. *ZigBee specification*. Tech. rep. Zigbee Alliance, June 2005.

[22] G. Montenegro, N. Kushalnagar, J. Hui et al. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC4944. Sept. 2007. URL: http://tools.ietf.org/rfc/rfc4944.txt.

[23] URL: http://www.ti.com/lsds/ti/wireless_connectivity/zigbee/products.page\#.

[24] URL: https://openwsn.atlassian.net/wiki/pages/viewpage.action?pageId=688187.

[25] D. Page. *SCALE: Side-Channel Attack Lab. Exercises*. URL: http://www.github.com/danpage/scale.

[26] Matthew McClintic, Devon Maloney, Michael Scires et al. 'Keyshuffling Attack for Persistent Early Code Execution in the Nintendo

3DS Secure Bootchain'. In: *CoRR* abs/1802.00092 (2018). arXiv: 1802.00092. URL: http://arxiv.org/abs/1802.00092.

[27] Michael Scire, Melissa Mears, Devon Maloney et al. 'Attacking the Nintendo 3DS Boot ROMs'. In: *CoRR* abs/1802.00359 (2018). arXiv: 1802.00359. URL: http://arxiv.org/abs/1802.00359.

[28] https://latesthackingnews.com/2018/09/13/researchers-discover-vulnerability-in-tesla-model-s-key/.

[29] Billy Bob Brumley and Nicola Tuveri. 'Remote Timing Attacks Are Still Practical'. In: *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*. Ed. by Vijay Atluri and Claudia Díaz. Vol. 6879. Lecture Notes in Computer Science. Springer, 2011, pp. 355–371. ISBN: 978-3-642-23821-5. DOI: 10.1007/978-3-642-23822-2_20. URL: https://doi.org/10.1007/978-3-642-23822-2_20.

[30] Stefan Mangard, Elisabeth Oswald and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. ISBN: 978-0-387-30857-9.

[31] Dawn Xiaodong Song, David A. Wagner and Xuqing Tian. 'Timing Analysis of Keystrokes and Timing Attacks on SSH'. In: *10th USENIX Security Symposium, August 13-17, 2001, Washington, D.C., USA*. Ed. by Dan S. Wallach. USENIX, 2001. URL: http://www.usenix.org/publications/library/proceedings/sec01/song.html.

[32] Shuo Chen, Rui Wang, XiaoFeng Wang et al. 'Side-channel leaks in web applications: A reality today, a challenge tomorrow'. In: *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE. 2010, pp. 191–206.

[33] Paul C. Kocher, Joshua Jaffe and Benjamin Jun. 'Differential Power Analysis'. In: *CRYPTO*. 1999, pp. 388–397.

[34] Stefan Mangard, Elisabeth Oswald and François-Xavier Standaert. 'One for all - all for one: unifying standard differential power analysis attacks'. In: *IET Information Security* 5.2 (2011), pp. 100–110. DOI: 10.1049/iet-ifs.2010.0096. URL: https://doi.org/10.1049/iet-ifs.2010.0096.

[35] Luke Mather, Elisabeth Oswald and Carolyn Whitnall. 'Multi-target DPA Attacks: Pushing DPA Beyond the Limits of a Desktop Computer'. In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, 2014, pp. 243–261. ISBN: 978-3-662-45610-1. DOI: 10.1007/978-3-662-45611-8_13. URL: https://doi.org/10.1007/978-3-662-45611-8_13.

[36] Suresh Chari, Josyula R. Rao and Pankaj Rohatgi. 'Template Attacks'. In: *CHES*. 2002, pp. 13–28.

[37] François-Xavier Standaert, Benedikt Gierlichs and Ingrid Verbauwhede. 'Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices'. In: *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*. Ed. by Pil Joong Lee and Jung Hee Cheon. Vol. 5461. Lecture Notes in Computer Science. Springer, 2008, pp. 253–267. ISBN: 978-3-642-00729-3. DOI: 10.1007/978-3-642-00730-9_16. URL: https://doi.org/10.1007/978-3-642-00730-9_16.

[38] Suresh Chari, Josyula R. Rao and Pankaj Rohatgi. 'Template Attacks'. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. ISBN: 3-540-00409-2. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5_3.

[39] Christian Rechberger and Elisabeth Oswald. 'Practical Template Attacks'. In: *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*. Ed. by Chae Hoon Lim and Moti Yung. Vol. 3325. Lecture Notes in Computer Science. Springer, 2004, pp. 440–456. ISBN: 3-540-24015-2. DOI: 10.1007/978-3-540-31815-6_35. URL: https://doi.org/10.1007/978-3-540-31815-6_35.

[40] Carolyn Whitnall and Elisabeth Oswald. 'Robust Profiling for DPA-Style Attacks'. In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 3–21. ISBN: 978-3-662-48323-7. DOI: 10.1007/978-3-662-48324-4_1. URL: https://doi.org/10.1007/978-3-662-48324-4_1.

[41] Stjepan Picek, Ioannis Petros Samiotis, Annelie Heuser et al. *On the Performance of Convolutional Neural Networks for Side-channel Analysis*. Cryptology ePrint Archive, Report 2018/004. https://eprint.iacr.org/2018/004. 2018.

[42] Alexander Schaub, Emmanuel Schneider, Alexandros Hollender et al. 'Attacking Suggest Boxes in Web Applications Over HTTPS Us-

ing Side-Channel Stochastic Algorithms'. In: *Risks and Security of Internet and Systems*. Springer, 2014, pp. 116–130.

[43] Luke Mather and Elisabeth Oswald. 'Pinpointing side-channel information leaks in web applications'. In: *Journal of Cryptographic Engineering* 2.3 (2012), pp. 161–177.

[44] Scott Coull and Kevin Dyer. 'Privacy Failures in Encrypted Messaging Services: Apple iMessage and Beyond'. In: *arXiv preprint arXiv:1403.1906* (2014).

[45] Charles V Wright, Lucas Ballard, Scott E Coull et al. 'Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations'. In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 35–49.

[46] Raheem Beyah Chris Wampler A. Selcuk Uluagac. *Information Leakage in Encrypted IP Video Traffic*. ieee-globecom 2015. 2015.

[47] Dominik Herrmann, Rolf Wendolsky and Hannes Federrath. 'Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier'. In: *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM. 2009, pp. 31–42.

[48] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart et al. 'Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail'. In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 332–346. ISBN: 978-0-7695-4681-0. DOI: 10.1109/SP.2012.28. URL: http://dx.doi.org/10.1109/SP.2012.28.

[49] Charles V Wright, Scott E Coull and Fabian Monrose. 'Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.' In: *NDSS*. 2009.

[50] Xiapu Luo, Peng Zhou, Edmond WW Chan et al. 'HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows.' In: *NDSS*. 2011.

[51] Kevin P Dyer, Scott E Coull, Thomas Ristenpart et al. 'Protocol misidentification made easy with format-transforming encryption'. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 61–72.

[52] Zhe Liu, Thomas Pöppelmann, Tobias Oder et al. 'High-Performance Ideal Lattice-Based Cryptography on 8-Bit AVR Microcontrollers'. In: *ACM Trans. Embedded Comput. Syst.* 16.4 (2017), 117:1–117:24. DOI: 10.1145/3092951. URL: http://doi.acm.org/10.1145/3092951.

[53] Thomas Pöppelmann and Tim Güneysu. 'Area optimization of lightweight lattice-based encryption on reconfigurable hardware'. In: *IEEE International Symposium on Circuits and Systemss, ISCAS 2014, Melbourne, Victoria, Australia, June 1-5, 2014*. IEEE, 2014, pp. 2796–2799. ISBN: 978-1-4799-3431-7. DOI: 10.1109/ISCAS.2014.6865754. URL: https://doi.org/10.1109/ISCAS.2014.6865754.

[54] Tobias Oder, Thomas Pöppelmann and Tim Güneysu. 'Beyond ECDSA and RSA: Lattice-based Digital Signatures on Constrained Devices'. In: *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*. ACM, 2014, 110:1–110:6. ISBN: 978-1-4503-2730-5. DOI: 10.1145/2593069.2593098. URL: http://doi.acm.org/10.1145/2593069.2593098.

[55] James Howe, Thomas Pöppelmann, Máire O'Neill et al. 'Practical Lattice-Based Digital Signature Schemes'. In: *ACM Trans. Embedded Comput. Syst.* 14.3 (2015), 41:1–41:24. DOI: 10.1145/2724713. URL: http://doi.acm.org/10.1145/2724713.

[56] Tanushree Banerjee and M Anwar Hasan. *Energy consumption of candidate algorithms for NIST PQC standards*. Tech. rep. Technical report, Centre for Applied Cryptographic Research (CACR) at the University of Waterloo http://cacr. uwaterloo. ca/, Accessed 26 June, 2018.

[57] Alex Biryukov and Léo Perrin. 'State of the Art in Lightweight Symmetric Cryptography'. In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 511. URL: http://eprint.iacr.org/2017/511.

[58] Emmanuel Prouff. 'DPA Attacks and S-Boxes'. In: *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. Ed. by Henri Gilbert and Helena Handschuh. Vol. 3557. Lecture Notes in Computer Science. Springer, 2005, pp. 424–441. ISBN: 3-540-26541-4. DOI: 10.1007/11502760_29. URL: https://doi.org/10.1007/11502760_29.

[59] Annelie Heuser, Stjepan Picek, Sylvain Guilley et al. 'Side-Channel Analysis of Lightweight Ciphers: Does Lightweight Equal Easy?' In: *Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers*. Ed. by Gerhard P. Hancke and Konstantinos Markantonakis. Vol. 10155. Lecture Notes in Computer Science. Springer, 2016, pp. 91–104. ISBN: 978-3-319-62023-7. DOI: 10.1007/978-3-319-62024-4_7. URL: https://doi.org/10.1007/978-3-319-62024-4_7.

[60] Akihiro Shimizu and Shoji Miyaguchi. 'FEAL - Fast Data Encipherment Algorithm'. In: *Systems and Computers in Japan* 19.7 (1988), pp. 20–34. DOI: 10.1002/scj.4690190703. URL: https://doi.org/10.1002/scj.4690190703.

[61] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. RFC7539. May 2015. URL: http://tools.ietf.org/rfc/rfc7539.txt.

[62] Daniel J. Bernstein. 'The Salsa20 Family of Stream Ciphers'. In: *New Stream Cipher Designs - The eSTREAM Finalists*. Ed. by Matthew J. B. Robshaw and Olivier Billet. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 84–97. ISBN: 978-3-540-68350-6. DOI: 10.1007/978-3-540-68351-3_8. URL: https://doi.org/10.1007/978-3-540-68351-3_8.

[63] Jean-Philippe Aumasson, Luca Henzen, Willi Meier et al. 'Sha-3 proposal blake'. In: *Submission to NIST* (2008).

[64] Niels Ferguson, Stefan Lucks, Bruce Schneier et al. 'The Skein hash function family'. In: *Submission to NIST (round 3)* 7.7.5 (2010), p. 3.

[65] Ray Beaulieu, Douglas Shors, Jason Smith et al. 'The SIMON and SPECK lightweight block ciphers'. In: *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015, 175:1–175:6. ISBN: 978-1-4503-3520-1. DOI: 10.1145/2744769.2747946. URL: http://doi.acm.org/10.1145/2744769.2747946.

[66] Daniel Dinu, Léo Perrin, Aleksei Udovenko et al. 'Design Strategies for ARX with Provable Bounds: Sparx and LAX'. In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 484–513. ISBN: 978-3-662-53886-9. DOI: 10.1007/978-3-662-53887-6_18. URL: https://doi.org/10.1007/978-3-662-53887-6_18.

[67] Olivier Benoît and Thomas Peyrin. 'Side-Channel Analysis of Six SHA-3 Candidates'. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 140–157. ISBN: 978-3-642-15030-2. DOI: 10.1007/978-3-642-15031-9_10. URL: https://doi.org/10.1007/978-3-642-15031-9_10.

[68] https://www.cosic.esat.kuleuven.be/ecrypt/courses/albena11/slides/nicky_mouha_arx-slides.pdf.

[69] S. V. Dilip Kumar, Sikhar Patranabis, Jakub Breier et al. 'A Practical Fault Attack on ARX-Like Ciphers with a Case Study on ChaCha20'. In: *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2017, Taipei, Taiwan, September 25, 2017*. IEEE Computer Society, 2017, pp. 33–40. ISBN: 978-1-5386-2948-2. DOI: 10.1109/FDTC.2017.14. URL: https://doi.org/10.1109/FDTC.2017.14.

[70] Alex Biryukov, Daniel Dinu and Johann Großschädl. 'Correlation Power Analysis of Lightweight Block Ciphers: From Theory to Practice'. In: *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi and Steve Schneider. Vol. 9696. Lecture Notes in Computer Science. Springer, 2016, pp. 537–557. ISBN: 978-3-319-39554-8. DOI: 10.1007/978-3-319-39555-5_29. URL: https://doi.org/10.1007/978-3-319-39555-5_29.

[71] Michael Zohner, Michael Kasper and Marc Stöttinger. 'Butterfly-Attack on Skein's Modular Addition'. In: *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE*

2012, Darmstadt, Germany, May 3-4, 2012. Proceedings. Ed. by Werner Schindler and Sorin A. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer, 2012, pp. 215–230. ISBN: 978-3-642-29911-7. DOI: 10.1007/978-3-642-29912-4_16. URL: https://doi.org/10.1007/978-3-642-29912-4_16.

[72] Tobias Schneider, Amir Moradi and Tim Güneysu. 'Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware'. In: *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*. 2015, pp. 559–578.

[73] https://keccak.team/files/NoteSideChannelAttacks.pdf.

[74] Jean-Sébastien Coron and Louis Goubin. 'On Boolean and Arithmetic Masking against Differential Power Analysis'. In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture Notes in Computer Science. Springer, 2000, pp. 231–237. ISBN: 3-540-41455-X. DOI: 10.1007/3-540-44499-8_18. URL: https://doi.org/10.1007/3-540-44499-8_18.

[75] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi et al. 'Conversion from Arithmetic to Boolean Masking with Logarithmic Complexity'. In: *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*. Ed. by Gregor Leander. Vol. 9054. Lecture Notes in Computer Science. Springer, 2015, pp. 130–149. ISBN: 978-3-662-48115-8. DOI: 10.1007/978-3-662-48116-5_7. URL: https://doi.org/10.1007/978-3-662-48116-5_7.

[76] Bernhard Jungk, Richard Petri and Marc Stöttinger. 'Efficient Side-Channel Protections of ARX Ciphers'. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.3 (Sept. 2018), pp. 627–653. DOI: 10.13154/tches.v2018.i3.627-653. URL: https://tches.iacr.org/index.php/TCHES/article/view/7289.

[77] Alex Biryukov, Daniel Dinu, Yann Le Corre et al. 'Optimal First-Order Boolean Masking for Embedded IoT Devices'. In: *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*. Ed. by Thomas Eisenbarth and Yannick Teglia. Vol. 10728. Lecture Notes in Computer Science. Springer, 2017, pp. 22–41. ISBN: 978-3-319-75207-5. DOI: 10.1007/978-3-319-75208-2_2. URL: https://doi.org/10.1007/978-3-319-75208-2_2.

[78] Taher El Gamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *IEEE Trans. Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074. URL: https://doi.org/10.1109/TIT.1985.1057074.

[79] Claus-Peter Schnorr. 'Efficient Identification and Signatures for Smart Cards'. In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 239–252. ISBN: 3-540-97317-6. DOI: 10.1007/0-387-34805-0_22. URL: https://doi.org/10.1007/0-387-34805-0_22.

[80] Oded Regev. 'On lattices, learning with errors, random linear codes, and cryptography'. In: *J. ACM* 56.6 (2009), 34:1–34:40. DOI: 10.1145/1568318.1568324. URL: http://doi.acm.org/10.1145/1568318.1568324.

[81] Jintai Ding. 'A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem'. In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 688. URL: http://eprint.iacr.org/2012/688.

[82] Erdem Alkim, Léo Ducas, Thomas Pöppelmann et al. 'Post-quantum key exchange - a new hope'. In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1092. URL: http://eprint.iacr.org/2015/1092.

[83] Joppe W. Bos, Craig Costello, Léo Ducas et al. 'Frodo: Take off the ring! Practical, Quantum-Secure Key Exchange from LWE'. In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 659. URL: http://eprint.iacr.org/2016/659.

[84] URL: https://rdist.root.org/2010/01/11/smart-meter-crypto-flaw-worse-than-thought/.

[85] Mario Stipcevic and Çetin Kaya Koç. 'True Random Number Generators'. In: *Open Problems in Mathematics and Computational Science*. Ed. by Çetin Kaya Koç. Springer, 2014, pp. 275–315. ISBN: 978-3-319-10682-3. DOI: 10.1007/978-3-319-10683-0_12. URL: https://doi.org/10.1007/978-3-319-10683-0_12.

[86] Patrick Haddad, Viktor Fischer, Florent Bernard et al. 'A physical approach for stochastic modeling of TERO-based TRNG'. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2015, pp. 357–372.

[87] Mehrdad Majzoobi, Farinaz Koushanfar and Srinivas Devadas. 'FPGA-based true random number generation using circuit metastability with adaptive feedback control'. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2011, pp. 17–32.

[88] Michal Varchola and Milos Drutarovsky. 'New high entropy element for FPGA based true random number generators'. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 351–365.

[89] *random.c – A strong random number generator*. URL: `https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git/tree/drivers/char/random.c?id=refs/tags/v3.15.6#n52`.

[90] Suleyman Gokhun Tanyer, Sitki Cagdas Inam and Kumru Didem Atalay. 'Analysis of true random number generation method utilizing FM radio signals'. In: *Signal Processing and Communications Applications Conference (SIU), 2015 23th*. IEEE. 2015, pp. 1590–1593.

[91] Ganesh K Balachandran and Raymond E Barnett. 'A 440-nA true random number generator for passive RFID tags'. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55.11 (2008), pp. 3723–3732.

[92] Çetin Kaya Koç, ed. *Open Problems in Mathematics and Computational Science*. Springer, 2014. ISBN: 978-3-319-10682-3. DOI: `10.1007/978-3-319-10683-0`. URL: `https://doi.org/10.1007/978-3-319-10683-0`.

[93] Jean Paul Degabriele, Kenneth G. Paterson, Jacob C. N. Schuldt et al. 'Backdoors in Pseudorandom Number Generators: Possibility and Impossibility Results'. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. Lecture Notes in Computer Science. Springer, 2016, pp. 403–432. ISBN: 978-3-662-53017-7. DOI: `10.1007/978-3-662-53018-4\_15`. URL: `https://doi.org/10.1007/978-3-662-53018-4\_15`.

[94] Marc Fischlin and Jean-Sébastien Coron, eds. *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016. ISBN: 978-3-662-49889-7. DOI: 10.1007/978-3-662-49890-3. URL: https://doi.org/10.1007/978-3-662-49890-3.

[95] Daniel Hutchinson. 'A Robust and Sponge-Like PRNG with Improved Efficiency'. In: *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. Lecture Notes in Computer Science. Springer, 2016, pp. 381–398. ISBN: 978-3-319-69452-8. DOI: 10.1007/978-3-319-69453-5_21. URL: https://doi.org/10.1007/978-3-319-69453-5_21.

[96] Elaine B. Barker and John M. Kelsey. *SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. Tech. rep. Gaithersburg, MD, United States, 2012.

[97] Lawrence E Bassham III, Andrew L Rukhin, Juan Soto et al. 'Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications'. In: (2010).

[98] URL: http://www.ti.com/lit/ds/symlink/cc2430.pdf.

[99] URL: http://www.ti.com/lit/ug/swru191f/swru191f.pdf.

[100] URL: http://travisgoodspeed.blogspot.co.uk/2009/12/prng-vulnerability-of-z-stack-zigbee.html.

[101] URL: http://www.ti.com/tool/z-stack.

[102] URL: http://www.ti.com/lit/ug/swru319c/swru319c.pdf.

[103] URL: http://www.ti.com/lit/ds/symlink/cc2520.pdf.

[104] URL: http://www.ti.com/lit/ds/symlink/cc2538.pdf.

[105] URL: http://www.openmote.com/.

[106] Juan Pablo Alegre Pérez, Santiago Celma Pueyo and Belén Calvo López. *Automatic Gain Control*. Springer, 2011.

[107] Iulian Rosu. *Automatic Gain Control(AGC) in Receivers*. URL: http://www.qsl.net/va3iul/Files/Automatic_Gain_Control.pdf.

[108] URL: http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion.

[109] URL: https://greatscottgadgets.com/hackrf/.

[110] William Wesley Peterson and Daniel T Brown. 'Cyclic codes for error detection'. In: *Proceedings of the IRE* 49.1 (1961), pp. 228–235.

[111] URL: https://sourceforge.net/projects/tinydtls/.

[112] P. Eronen and H. Tschofenig. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*. RFC4279. Dec. 2005. URL: http://tools.ietf.org/rfc/rfc4279.txt.

[113] S. Blake-Wilson, N. Bolyard, V. Gupta et al. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*. RFC4492. May 2006. URL: http://tools.ietf.org/rfc/rfc4492.txt.

[114] SECG SEC. '2: Recommended elliptic curve domain parameters'. In: *Standards for Efficient Cryptography Group, Certicom Corp* (2000).

[115] Don Johnson, Alfred Menezes and Scott Vanstone. 'The elliptic curve digital signature algorithm (ECDSA)'. In: *International Journal of Information Security* 1.1 (2001), pp. 36–63.

[116] L. Bassham, W. Polk and R. Housley. *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC3279. Apr. 2002. URL: `http://tools.ietf.org/rfc/rfc3279.txt`.

[117] URL: `https://chromium.googlesource.com/native_client/nacl-newlib/+/master/newlib/libc/stdlib/rand_r.c`.

[118] URL: `https://github.com/vancegroup-mirrors/avr-libc/blob/master/avr-libc/libc/stdlib/rand.c`.

[119] URL: `https://sourceforge.net/p/mspgcc/gcc/ci/master/tree/libiberty/random.c`.

[120] URL: `http://www.ti.com/lit/ds/symlink/cc2420.pdf`.

[121] URL: `http://www.ti.com.cn/cn/lit/ug/swcu117d/swcu117d.pdf`.

[122] https://en.wikipedia.org/wiki/Skein_(hash_function)#/media/File:Skein_permutation.png

[123] Hasindu Gamaarachchi, Harsha Ganegoda and Roshan Ragel. 'Breaking Speck cryptosystem using correlation power analysis attack'. In: *Journal of the National Science Foundation of Sri Lanka* 45.4 (2017).

[124] Emmanuel Prouff. 'DPA Attacks and S-Boxes'. In: *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. 2005, pp. 424–441.

[125] Carolyn Whitnall, Elisabeth Oswald and François-Xavier Standaert. 'The Myth of Generic DPA... and the Magic of Learning'. In: *Topics in Cryptology – CT-RSA 2014*. Ed. by Josh Benaloh. Cham: Springer International Publishing, 2014, pp. 183–205.

[126] Kerstin Lemke, Kai Schramm and Christof Paar. 'DPA on n-Bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction'. In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cam-*

bridge, MA, USA, August 11-13, 2004. Proceedings. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 205–219. ISBN: 3-540-22666-4. DOI: 10.1007/978-3-540-28632-5_15. URL: https://doi.org/10.1007/978-3-540-28632-5_15.

[127] https://developer.arm.com/products/processors/cortex-m/cortex-m0.

[128] https://www.cryptolux.org/index.php/SPARX.

[129] Sonia Belaïd, Pierre-Alain Fouque and Benoît Gérard. 'Side-Channel Analysis of Multiplications in GF(2128) - Application to AES-GCM'. In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Springer, 2014, pp. 306–325. ISBN: 978-3-662-45607-1. DOI: 10.1007/978-3-662-45608-8_17. URL: https://doi.org/10.1007/978-3-662-45608-8_17.

[130] Sonia Belaïd, Jean-Sébastien Coron, Pierre-Alain Fouque et al. 'Improved Side-Channel Analysis of Finite-Field Multiplication'. In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 395–415. ISBN: 978-3-662-48323-7. DOI: 10.1007/978-3-662-48324-4_20. URL: https://doi.org/10.1007/978-3-662-48324-4_20.

[131] David McCann, Carolyn Whitnall and Elisabeth Oswald. 'ELMO: Emulating Leaks for the ARM Cortex-M0 without Access to a Side Channel Lab'. In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 517. URL: http://eprint.iacr.org/2016/517.

[132] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon et al. 'A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices'. In: *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 109–128. ISBN: 978-3-642-20464-7. DOI: 10.1007/978-3-642-20465-4_8. URL: https://doi.org/10.1007/978-3-642-20465-4_8.

[133] Carolyn Whitnall and Elisabeth Oswald. 'A fair evaluation framework for comparing side-channel distinguishers'. In: *J. Cryptographic Engineering* 1.2 (2011), pp. 145–160. DOI: 10.1007/s13389-011-0011-1. URL: https://doi.org/10.1007/s13389-011-0011-1.

[134] Liam Paninski. 'Estimation of Entropy and Mutual Information'. In: *Neural Computation* 15.6 (2003), pp. 1191–1253. DOI: 10.1162/089976603321780272. URL: https://doi.org/10.1162/089976603321780272.

[135] Benedikt Gierlichs, Lejla Batina, Pim Tuyls et al. 'Mutual Information Analysis'. In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 426–442. ISBN: 978-3-540-85052-6. DOI: 10.1007/978-3-540-85053-3_27. URL: https://doi.org/10.1007/978-3-540-85053-3_27.

[136] Nicolas Veyrat-Charvillon and François-Xavier Standaert. 'Generic Side-Channel Distinguishers: Improvements and Limitations'. In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer

Science. Springer, 2011, pp. 354–372. ISBN: 978-3-642-22791-2. DOI: 10.1007/978-3-642-22792-9_20. URL: https://doi.org/10.1007/978-3-642-22792-9_20.

[137] Oscar Reparaz, Benedikt Gierlichs and Ingrid Verbauwhede. 'Generic DPA Attacks: Curse or Blessing?' In: *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*. Ed. by Emmanuel Prouff. Vol. 8622. Lecture Notes in Computer Science. Springer, 2014, pp. 98–111. ISBN: 978-3-319-10174-3. DOI: 10.1007/978-3-319-10175-0_8. URL: https://doi.org/10.1007/978-3-319-10175-0_8.

[138] Julia Borghoff, Anne Canteaut, Tim Güneysu et al. 'PRINCE - A Low-latency Block Cipher for Pervasive Computing Applications (Full version)'. In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 529. URL: http://eprint.iacr.org/2012/529.

[139] *AES Furious*.

[140] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander et al. 'PRESENT: An Ultra-Lightweight Block Cipher'. In: *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. Lecture Notes in Computer Science. Springer, 2007, pp. 450–466. ISBN: 978-3-540-74734-5. DOI: 10.1007/978-3-540-74735-2_31. URL: https://doi.org/10.1007/978-3-540-74735-2_31.

[141] David H. Wolpert and William G. Macready. 'No free lunch theorems for optimization'. In: *IEEE Trans. Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893. URL: https://doi.org/10.1109/4235.585893.

[142] Carolyn Whitnall, Elisabeth Oswald and Luke Mather. 'An Exploration of the Kolmogorov-Smirnov Test as a Competitor to Mutual Information Analysis'. In: *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*. Ed. by Emmanuel Prouff. Vol. 7079. Lecture Notes in Computer Science. Springer, 2011, pp. 234–251. ISBN: 978-3-642-27256-1. DOI: 10.1007/978-3-642-27257-8_15. URL: https://doi.org/10.1007/978-3-642-27257-8_15.

[143] Eloi de Chérisey, Sylvain Guilley, Annelie Heuser et al. 'On the optimality and practicability of mutual information analysis in some scenarios'. In: *Cryptography and Communications* 10.1 (2018), pp. 101–121. DOI: 10.1007/s12095-017-0241-x. URL: https://doi.org/10.1007/s12095-017-0241-x.

[144] John B Kenney. 'Dedicated short-range communications (DSRC) standards in the United States'. In: *Proceedings of the IEEE* 99.7 (2011), pp. 1162–1182.

[145] Availabe at https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy.

[146] Sridhar Rajagopal, Richard D Roberts and Sang-Kyu Lim. 'IEEE 802.15. 7 visible light communication: modulation schemes and dimming support'. In: *IEEE Communications Magazine* 50.3 (2012).

[147] ZigBee Alliance. *ZigBee IP and 920IP*. 2014.

[148] Adam Dunkels, Fredrik Österlind and Zhitao He. 'An adaptive communication architecture for wireless sensor networks'. In: *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, SenSys 2007, Sydney, NSW, Australia, November 6-9, 2007*. Ed. by Sanjay Jha. ACM, 2007, pp. 335–349. ISBN: 978-1-59593-763-6.

DOI: 10.1145/1322263.1322295. URL: http://doi.acm.org/10.1145/1322263.1322295.

[149] Availabe at http://www.freertos.org/.

[150] Availabe at https://www.riot-os.org/.

[151] Pavan Pongle and Gurunath Chavan. 'A survey: Attacks on RPL and 6LoWPAN in IoT'. In: *Pervasive Computing (ICPC), 2015 International Conference on*. IEEE. 2015, pp. 1–6.

[152] René Hummen, Jens Hiller, Hanno Wirtz et al. '6LoWPAN fragmentation attacks and mitigation mechanisms'. In: *Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC'13, Budapest, Hungary, April 17-19, 2013*. Ed. by Levente Buttyán, Ahmad-Reza Sadeghi and Marco Gruteser. ACM, 2013, pp. 55–66. ISBN: 978-1-4503-1998-0. DOI: 10.1145/2462096.2462107. URL: http://doi.acm.org/10.1145/2462096.2462107.

[153] Ioannis Krontiris, Thanassis Giannetsos and Tassos Dimitriou. 'Launching a sinkhole attack in wireless sensor networks; the intruder side'. In: *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,* IEEE. 2008, pp. 526–531.

[154] Virendra Pal Singh, Aishwarya S Anand Ukey and Sweta Jain. 'Signal strength based hello flood attack detection and prevention in wireless sensor networks'. In: *International Journal of Computer Applications* 62.15 (2013).

[155] Yih-Chun Hu, Adrian Perrig and David B Johnson. 'Wormhole attacks in wireless networks'. In: *Selected Areas in Communications, IEEE Journal on* 24.2 (2006), pp. 370–380.

[156] Mohammad Wazid, Avita Katal, Roshan Singh Sachan et al. 'Detection and prevention mechanism for Blackhole attack in Wireless Sensor Network'. In: *Communications and Signal Processing (ICCSP), 2013 International Conference on*. IEEE. 2013, pp. 576–581.

[157] Naveen Sastry and David A. Wagner. 'Security considerations for IEEE 802.15.4 networks'. In: *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004*. Ed. by Markus Jakobsson and Adrian Perrig. ACM, 2004, pp. 32–42. ISBN: 1-58113-925-X. DOI: 10.1145/1023646.1023654. URL: http://doi.acm.org/10.1145/1023646.1023654.

[158] Adam Dunkels. 'The contikimac radio duty cycling protocol'. In: *SICS Report* (2011).

[159] Pascal Thubert, Thomas Watteyne, Maria Rita Palattella et al. 'IETF 6TSCH: Combining IPv6 Connectivity with Industrial Performance.' In: *IMIS*. Ed. by Leonard Barolli, Ilsun You, Fatos Xhafa et al. IEEE Computer Society, 2013, pp. 541–546. ISBN: 978-0-7695-4974-3. URL: http://dblp.uni-trier.de/db/conf/imis/imis2013.html#ThubertWPVW13.

[160] International Organization for Standardization ISO. *ISO/IEC 7498-1Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. Tech. rep. ISO, June 1994, pp. 34–65+.

[161] Z. Shelby, K. Hartke and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC7252. June 2014. URL: http://tools.ietf.org/rfc/rfc7252.txt.

[162] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC6347. Jan. 2012. URL: http://tools.ietf.org/rfc/rfc6347.txt.

[163] J. Postel. *User Datagram Protocol*. RFC0768. Aug. 1980. URL: `http://tools.ietf.org/rfc/rfc0768.txt`.

[164] Konrad-Felix Krentz, Hosnieh Rafiee and Christoph Meinel. '6LoWPAN security: adding compromise resilience to the 802.15.4 security sublayer'. In: *Proceedings of the International Workshop on Adaptive Security, ASPI@UbiComp 2013, Zurich, Switzerland, September 8, 2013*. ACM, 2013, 1:1–1:10. ISBN: 978-1-4503-2543-1. DOI: `10.1145/2523501.2523502`. URL: `http://doi.acm.org/10.1145/2523501.2523502`.

[165] Morris J. Dworkin. *SP 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. Tech. rep. Gaithersburg, MD, United States: National Institute of Standards & Technology, 2004.

[166] URL: `https://projects.eclipse.org/projects/iot.tinydtls`.

[167] D. McGrew and D. Bailey. *AES-CCM Cipher Suites for Transport Layer Security (TLS)*. RFC6655. July 2012. URL: `http://tools.ietf.org/rfc/rfc6655.txt`.

[168] A. Conta, S. Deering and M. Gupta. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. RFC4443. Mar. 2006. URL: `http://tools.ietf.org/rfc/rfc4443.txt`.

[169] T. Winter, P. Thubert, A. Brandt et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC6550. Mar. 2012. URL: `http://tools.ietf.org/rfc/rfc6550.txt`.

[170] Vijay Kumar, George Oikonomou and Theo Tryfonas. 'Traffic forensics for IPv6-based Wireless Sensor Networks and the Internet of Things'. In: *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*. IEEE. 2016, pp. 633–638.

[171] Franck Veysset, Olivier Courtay, Olivier Heen et al. 'New tool and technique for remote operating system fingerprinting'. In: *Intranode Software Technologies* 4 (2002).

[172] M. Kulkarni, A. Patel and K. Leung. *Mobile IPv4 Dynamic Home Agent (HA) Assignment*. RFC4433. Mar. 2006. URL: `http://tools.ietf.org/rfc/rfc4433.txt`.

[173] R. Seggelmann, M. Tuexen and M. Williams. *Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension*. RFC6520. Feb. 2012. URL: `http://tools.ietf.org/rfc/rfc6520.txt`.

[174] URL: `https://github.com/Salties/Traffic-Analysis-on-6LoWPAN/blob/master/experiments/coaptime/er-example-server.c`.

[175] Andrej N Kolmogorov. *Sulla determinazione empirica di una legge di distribuzione*. na, 1933.

[176] Carolyn Whitnall, Elisabeth Oswald and Luke Mather. 'An Exploration of the Kolmogorov-Smirnov Test as a Competitor to Mutual Information Analysis'. In: *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*. Ed. by Emmanuel Prouff. Vol. 7079. Lecture Notes in Computer Science. Springer, 2011, pp. 234–251. ISBN: 978-3-642-27256-1. DOI: `10.1007/978-3-642-27257-8\_15`. URL: `https://doi.org/10.1007/978-3-642-27257-8\_15`.

[177] Tim Güneysu and Helena Handschuh, eds. *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015. ISBN: 978-3-

662-48323-7. DOI: 10.1007/978-3-662-48324-4. URL: https://doi.org/10.1007/978-3-662-48324-4.

[178] Emmanuel Prouff, ed. *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*. Vol. 7079. Lecture Notes in Computer Science. Springer, 2011. ISBN: 978-3-642-27256-1. DOI: 10.1007/978-3-642-27257-8. URL: https://doi.org/10.1007/978-3-642-27257-8.

Appendix A

Fingerprint Experiment Programs

Index	Program	Total Size	Fingerprint Size	Note
1	broadcast	6489	593	
2	broadcast	6164	639	
3	powertrace	7142	539	
4	powertrace	7079	561	
5	Sensorpayload	7338	987	Temperature + ALS
6	Sensorpayload	7963	934	Temperature + ALS
7	Sensorpayload	7143	1195	Temperature only
8	Sensorpayload	7316	1096	Temperature only
9	Sensorpayload	7895	827	ALS only
10	Sensorpayload	7867	789	ALS only
11	Sensorpayload	7428	1138	No reading
12	Sensorpayload	7462	833	No reading
13	Sensorpayload	6565	1391	Vdd only
14	Sensorpayload	7193	1111	Vdd only
15	Sensorpayload	7672	955	Temperature, Vdd and ALS
16	Sensorpayload	7790	1023	Temperature, Vdd and ALS
17	Sensorpayload	7864	931	Vdd + ALS
18	Sensorpayload	7936	987	Vdd + ALS
19	Sensorpayload	7217	1222	Temperature + Vdd
20	Sensorpayload	7050	1228	Temperature + Vdd

Table A.1: Fingerprint Experiment Programs

Appendix B

Fingerprint Experiment Relative KS-Distances

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	N/A	**7.8**	75	78.9	48.3	58.4	19.2	15.8	62.2	58.5	34.2	34	13.2	10.3	54.1	57	52.8	58.7	19.4	13.3
2	**7.8**	N/A	81.7	85.6	54.8	65.1	25.6	22.4	68.7	65	40.8	40.4	14.9	16.7	60.7	63.7	59.3	65.4	26	19.3
3	75	81.7	N/A	**6.9**	30.3	22.9	56.3	59.5	16.2	21.2	41.2	41.5	67	65	21.5	22.8	23.2	29.8	55.8	62.5
4	78.9	85.6	**6.9**	N/A	32.3	22.5	60.3	63.6	17.9	20.9	45.2	45.6	71	69.3	25.2	22.5	26.7	27.4	59.8	66.7
5	48.3	54.8	30.3	32.3	N/A	**11.3**	29.4	32.7	15.1	14	14.5	14.7	40.1	38.3	14.8	16.1	17.1	23.8	28.9	35.6
6	58.4	65.1	22.9	22.5	11.3	N/A	39.5	42.9	9	**4.9**	24.4	24.7	50.3	48.4	14	16	16.8	23.7	39.2	45.8
7	19.2	25.6	56.3	60.3	29.4	39.5	N/A	**4.4**	43.2	39.5	19.7	17	14.5	9.6	35.3	38.5	33.7	39.7	10.8	9.6
8	15.8	22.4	59.5	63.6	32.7	42.9	**4.4**	N/A	46.6	42.8	19.5	19.1	14.9	6.2	38.4	41.7	37.1	43.2	10.9	9
9	62.2	68.7	16.2	17.9	15.1	9	43.2	46.6	N/A	**5.4**	28.2	29	54	52.1	10.3	12.4	12.3	19	42.8	49.7
10	58.5	65	21.2	20.9	14	**4.9**	39.5	42.8	5.4	N/A	24.5	25.4	50.2	48.4	11.5	13.4	14.4	20.9	39.1	46.4
11	34.2	40.8	41.2	45.2	14.5	24.4	19.7	19.5	28.2	24.5	N/A	**7.4**	28.2	24.7	20	23.1	19	24.7	15	23.9
12	34	40.4	41.5	45.6	14.7	24.7	17	19.1	29	25.4	**7.4**	N/A	25.7	23.9	20.7	23.7	19	25	14.9	21.7
13	13.2	14.9	67	71	40.1	50.3	14.5	14.9	54	50.2	28.2	25.7	N/A	11.7	45.9	49.2	44.5	50.6	15.7	**8.9**
14	10.3	16.7	65	69.3	38.3	48.4	9.6	**6.2**	52.1	48.4	24.7	23.9	11.7	N/A	44	47.2	42.8	48.7	10.5	7.2
15	54.1	60.7	21.5	25.2	14.8	14	35.3	38.4	10.3	11.5	20	20.7	45.9	44	N/A	**3.8**	6.7	11.2	34.8	41.5
16	57	63.7	22.8	22.5	16.1	16	38.5	41.7	12.4	13.4	23.1	23.7	49.2	47.2	**3.8**	N/A	8.8	8.9	37.9	44.7
17	52.8	59.3	23.2	26.7	17.1	16.8	33.7	37.1	12.3	14.4	19	19	44.5	42.8	**6.7**	8.8	N/A	11.5	33.6	40
18	58.7	65.4	29.8	27.4	23.8	23.7	39.7	43.2	19	20.9	24.7	25	50.6	48.7	11.2	**8.9**	11.5	N/A	39.4	46.1
19	19.4	26	55.8	59.8	28.9	39.2	10.8	10.9	42.8	39.1	15	14.9	15.7	10.5	34.8	37.9	33.6	39.4	N/A	**10.3**
20	13.3	19.3	62.5	66.7	35.6	45.8	9.6	9	49.7	46.4	23.9	21.7	8.9	**7.2**	41.5	44.7	40	46.1	10.3	N/A

Table B.1: Relative KS-Distances of Experimented Fingerprints (Multiplied by 100 for readability. Indexes refer to Table A.1 Minimum in each row marked as **bold**.)