Side Channel Attack on Low Power FPGA Platform

by

Mustafa Faraj

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2016

Declaration

Abstract

In today's advanced electronic age, people have become accustomed to using electronic devices to store and process their information. There is a general belief that the information is safe, due to the use of mathematically proven cryptographic systems in critical devices. However, in recent years, various side channel attacks have been used to break the security of systems that were thought to be completely safe. Side channel attacks are based on information gained through the physical implementation of a cryptosystem, rather than its mathematical construction.

In this thesis work, an investigation is carried out to examine the susceptibility of the Hash-based Message Authentication Code standard based on the Secure Hash Algorithm (HMAC-SHA256) cryptosystem to a known correlation power analysis attack. For the purpose of this investigation, the cryptosystem was implemented on a low power Xilinx Field-Programmable Gate Array (FPGA) on the Side Channel Attack Standard Evaluation Board (SASEBO) platform. A secondary objective of the research work was to explore whether the SASEBO platform used may be easily modified to run side channel attacks on different cryptosystems.

Four different side channel attacks were carried out on the HMAC-SHA256 implementation on the Xilinx Virtex-5 FPGA; two were based on power consumption measurements and two on electromagnetic (EM) emanation above the FPGA chip. This thesis has shown that SAESBO platform can be used as a testbed for examining the power side channel analysis of different cryptosystems with a small percentage of FPGA overhead. Although the EM emanations from SAESBO are not viable for side channel analysis, power from the on-chip core can be utilized. In addition the previously researched carry-propagate and pre averaging techniques have been verified and found to be useful on this low power FPGA chip, requiring approximately 43776 traces for the guess of the correct secret intermediate values to reach among the top 5 ranked guesses.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Professor Catherine Gebotys for her time and support. Her expertise and continuous guidance were key factors in allowing me to finish this research work and achieve target results.

Many thanks Dr. Edgar Mateos and Brian White for setting up the ground work for this thesis project. A special thank you to Dr. Mateos for providing me the very valuable training on the lab equipment and procedures.

I would also like to thank my colleagues from Professor Gebotys' research group: Najam Jose, Karim Amin, Haohao Liao, Mahmoud Khalafalla and Caio Hoffman for their technical discussions, friendship and encouragement.

Finally and most importantly, I would like to thank my parents, Osam Faraj and Maysoon Ali, my brothers, Murtadha Faraj and Anwar Abdul-Nabi, and my sister, Mina Faraj for their unconditional love, support and encouragement.

Dedication

This thesis is dedicated to my parents: Osam Faraj and Maysoon Ali; for they have always stood by me, supported me and inspired me to strive for the best in any life endeavor.

Table of Contents

List of Tables

List of Figures

Acronyms

AES Advanced Encryption Standard.

CMOS Complementary Metal-Oxide Semiconductor.

CPA Corrrelation Power Analysis.

DES Data Encryption Standard.

DPA Differential Power Analysis.

EM Electromagnetic.

FIPS Federal Information Processing Standards.

FPGA Field Programmable Gate Array.

HMAC Hash-Based Message Authentication Code.

I/O Input/Output.

IC Integrated Circuit.

ISE Integrated Synthesis Environment.

LUT Look Up Table.

MAC	Message Authentication Code.
MD5	Message Digest 5.
PC	Personal Computer.
PROM	Programmable Read Only Memory.
SCA	Side Channel Attack.
SHS	Secure Hash Standard.
SPA	Simple Power Analysis.
USB	Universal Serial Port.

Chapter 1

Introduction

In today's advanced technological age, people have come to rely on the use of electronic devices to process much of the information they possess. There is a very wide variety of electronic devices used on a daily basis, ranging from the classical personal computer and smartphone to internet connected fridges, cars and even personal health trackers. A lot of the information processed through these devices is assumed to be secure, and in some cases it is extremely vital that this assumption holds, as the data may be personal and sensitive.

This led to the development of the field of digital security, which can be discussed using the four concepts of: *Confidentiality, Integrity, Authentication and Nonrepudiation* [10]. A brief description of those concepts is given below:

- **Confidentiality:** The concept of restricting access to information. Only authorized entities may access said information. This is typically achieved through the use of *encryption*.
- **Integrity:** The concept of protecting data from unauthorized tampering. Data may not be modified in an unauthorized or undetected manner.

- **Authentication:** The concept of verifying the claim of identity; it can be applied to users to verify they are who they claim to be, and it can be applied to data to verify its origin matches the claimed source. Data authentication can be achieved through *Message Authentication Code (MAC)* or *digital signatures* .

- **Nonrepudiation:** The concept of assuring that an authentication can be asserted to be genuine. This is inherently a legal concept adopted to the field of digital security to provide proof of integrity. It is typically achieved through *digital certificates.*

The implementation of these digital security concepts in digital electronics is done through *cryptographic algorithms and/or protocols*, which will be generally referred to as *cryptographic systems.* The design and implementation of such systems rely on advanced knowledge in the fields of mathematics, computer science and electrical engineering.

Naturally, due to their widespread use, electronic devices can process valuable information, such as banking and health information, which motivates some parties to gain unauthorized access to that information. To do so, an *attack* must be performed on the cryptographic system to get past on or more of the concepts on which digital security is built.

1.1 Side Channel Attacks

In the past, most attacks on cryptographic systems were based on theoretical weaknesses in the system or a brute force approach. Those attacks rely on knowing or attempting to retrieve the ciphertext or plaintext of the encryption operation. That is because cryptographic devices were perceived as a black box unit whose only output is a ciphertext when given a plaintext to operate on. Attacks such as the *known plaintext attack* and the *chosen plaintext attack* fall into that category of attacks.

The name *Side Channel Attacks (SCAs)* is given to any attack on a cryptographic system that relies on additional information generated by the physical implementation of a cryptographic system, rather than on a weakness in the cryptographic algorithm.

Side channel attacks are motivated by the fact that mathematically secure cryptographic algorithms need to be implemented in hardware or software to be useful for real world applications. Studies in this field have shown that the electronic devices used to implement encryption and decryption operations tend to leak information that may be used to compromise the security of the system. Today it is known that implementations of cryptographic systems leak information such as timing information, power consumption and electromagnetic emanation. Side channel attacks make use some or all of this information to break the security of the system by retrieving some internal state of the system.

An internal state of a cryptographic system can be defined as the intermediate values and results of the cryptographic operation that are not included in the output in normal operations of the cryptosystem [20]. Side channels in the physical implementation of systems can usually leak information, which when combined with knowledge about the underlying cryptographic algorithm, can reveal valuable information about the internal state of the system.

1.2 Message Authentication Code

In many applications, secure communication over insecure channels is required. Many encryption schemes are available to provide secrecy; however they do not provide integrity. An encrypted message may be modified in transit, and without a method of authenticating the message, the receiver might not recognize that the message had been tampered with.

A *MAC* is a computed tag used to authenticate a message. More specifically, it is used to verify that a message came from the expected sender and that it has not been changed in transit. Formally speaking, a MAC consists of three probabilistic polynomial-time algorithms (KeyGen, Mac, Verify) [3]:

- **Keygen:** A key generation algorithm that selects, at random, a *key* (to be used by Mac) from the defined key space.
- **Mac:** An algorithm that computes a *tag* given a key and a *message*.
- **Verify:** An algorithm that uses the *message*, *key* and *tag* to verify whether the message was tampered with in transit.

MAC algorithms are typically constructed from existing cryptographic primitives. Hash-based Message Authentication Code (HMAC), for example, is based on Cryptographic Hash Functions. Cipher Block Chaining Message Authentication Code (CBC-MAC) is a MAC algorithm based on Block Cipher. In the case of HMAC, any cryptographic hash function may be used in the calculation of a MAC. Message Digest Algorithm 5 (MD5) and Secure Hash Algorithm-2 (SHA-2) are examples of commonly used hash functions to make up HMACs.

1.3 Thesis Overview

The purpose of this thesis work is twofold: to examine the susceptibility of the HMAC-SHA256 algorithm implemented on the low power Xilinx Virtex-5 FPGA to correlation power analysis attacks, and to assess the reusability of the SASEBO-GII platform in side channel experiments other than the differential power analysis attack for which the board was designed as a part of the DPA Contest. To that end, the contents of the thesis chapters are summarized below.

Chapter two presents background information related to the attack performed in this thesis project. It contains a discussion of practical side channel attacks, focusing on the different techniques to perform power side channel attacks. Further it contains a description of the HMAC-SHA256 algorithm.

Chapter three presents setup of the experiment used in this project. It contains an overview of the equipment used and how the equipment were setup. Further it contains a description of the particular hardware architecture that was attacked, and a description of the attack itself and how it was executed.

Chapter four presents the experimental results of this study. In particular, it will present four sets of results for four attacks that were performed on the hardware architecture in question. Three of the attacks were not successful (correct key was not recovered with certainty). However one attack, CPA on power consumption with internal register values sent off-chip, was successful and secret key of HMAC-SHA256 was recovered.

Chapter 2

Background and Related Work

This chapter is divided into three main sections and introduces background details about the thesis work. The first section provides a review of practical side channel attacks in the, with an emphasis on power analysis attacks. The second section provides a review of the Hash-Based Message Authentication Code (HMAC) and SHA256 algorithms. Finally, the third section contains a literature review on side channel attack attempts on the HMAC-SHA256 algorithm.

2.1 Practical Side Channel Attacks

As described in section 1.1, physical implementations of cryptographic systems often leak information through side channels which can be used to compromise the security of the system. The most commonly used side channel information in such attacks are: timing information, power consumption information and electromagnetic emanation. The focus of this paper will be power consumption attacks, however a brief introduction is provided next on the other forms of attacks.

Timing Attacks:
In a general sense, timing attacks work by measuring the amount of time a system takes to perform certain operations. Cryptosystems, and computer systems in general, take marginally different amounts of time for the processing of different inputs. One of the main reasons behind this variability is the different execution paths followed for different inputs; however other factors such as specific optimizations, cache hits and multi-cycle instructions also contribute to the variability.

In [21] computationally inexpensive timing attacks are demonstrated which can be used to find fixed Diffie-Hellman exponents and factor RSA keys against vulnerable systems. Measured timing information is used along with knowledge about the underlying cryptosystem, and with the help of statistical models at times, to guess part of or the entire secret key.

Power Consumption Analysis:
There are various proven side channel attacks that rely on analyzing the power consumption of the cryptographic device. Those attacks typically exploit the relationship between the internal information being processed by the device and the power consumption profile of the device during processing the data. In order to discuss those attacks, a brief background on the power consumption model is presented next.

Complementary Metal-Oxide Semiconductor (CMOS) is the most commonly used technology for constructing digital Integrated Circuit (IC). The inverter gate, consisting of a P-type transistor and a N-type transistor, shown in Figure 2.1 is the most basic gate built using CMOS technology; it will be used to define an approximation of the power consumption model. More details on CMOS power consumption can be found in [35] and [36].

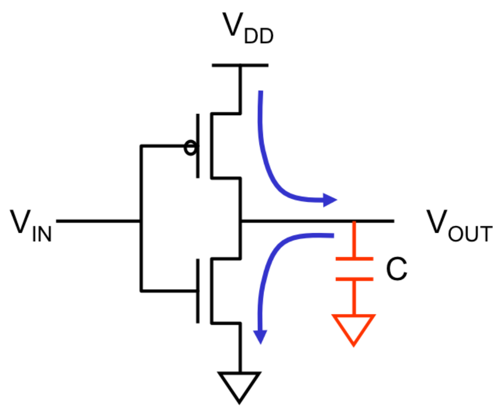


Figure 2.1: Basic CMOS Inverter Gate

The total power consumption of a CMOS gate is defined as the sum of static and dynamic power through the gate:

$$P_{total} = P_{static} + P_{dynamic}$$

The static power consumed by the gate, P_{static}, is due to the leakage currents in both transistors. Leakage in semiconductor devices usually increases as the thickness of the insulating region of the transistor decreases. The amount of leakage current is typically very small and consistent across different modes of operation of the transistor, and therefore can be treated a constant and ignored.
Dynamic power consumption, $P_{dynamic}$, is the sum of switching power and short-circuit consumption:

$$P_{dynamic} = P_{switching} + P_{short-circuit}$$

The short-circuit power is the amount of power dissipated when both the PMOS and NMOS devices are turned on, which happens for a very brief amount of time when the input of the gate changes. The short-circuit power is negligible when compared to the switching power, which is the amount of power used to charge and discharge the load capacitance. The dynamic power of a CMOS gate can be expressed as follows:

$$P_{dynamic} = C_L V_{DD}^2 A f$$

where C_L is the load capacitance, V_{DD} is the supply voltage, A is the switching activity factor and f is the clock frequency. The activity factor is the number of a logic 0 to 1 transitions. This equation for the dynamic power may be used to estimate the total power consumed by a CMOS device. It shows that the total power consumed is proportional to the number of 0 to 1 transitions. In other words that the power consumed by CMOS devices is data-dependent, hence the data leakage through the power side-channel depends on the number of bits switching from one state to the other [4] [6] [7]. There are three commonly used attacks to exploit the power consumption data-dependence in cryptographic systems; namely the Simple Power Analysis (SPA) attack, the Differential Power Analysis (DPA) attack and the Corrrelation Power Analysis (CPA) attack.

2.1.1 Simple Power Analysis Attacks

In SPA, information about the internal state of a cryptographic system is deduced directly from a power trace. Assuming the attacker has access to the physical device and an oscilloscope, a power consumption trace may attained and analyzed. Combining the information gathered from the power trace with knowledge about the implementation of the system may reveal parts or all of the key.

Devices running cryptographic algorithms tend to use a few primitive machine instructions (such as LOAD, STORE, MULT) repeatedly during a computation. This regularity can often be seen in power traces and be used to break cryptographic implementations in which the execution path is dependent on the data being processed. In [23], a few cases are presented in which this technique was used, including:

- Data Encryption Standard (DES) key schedule: This computation involves rotating the contents of the key register. A conditional branch is used after the rotate operation to test whether the bit shifted off the end of the register is a 0 or 1. The power trace of the branch, and subsequent operations, differs depending on the value of the bit.

- Multipliers: Modular multiplication circuits tend to have a large side channels; the information they leak tend to be strongly correlated to the operand values they are processing.

Many cryptographic systems have been tested and were found vulnerable to SPA attacks; however [22] suggests easy to implement countermeasures to prevent the detection of valuable key information through SPA attacks.

2.1.2 Differential Power Analysis Attacks

DPA is a more powerful attack than SPA and is generally more difficult to prevent. While SPA attacks rely mainly on visual inspections of power traces to identify features in consumption power fluctuations, DPA attacks use more advanced statistical analysis and error correction techniques to extract information correlated to the secret key.

DPA attacks typically require a large set of power consumption traces on which statistical analysis is performed to guess an intermediate result inside the cryptographic device, which is a function of plaintext/ciphertext and the secret key. Due to this nature of the attack, no detailed knowledge about the system under attack is required to perform a successful DPA attack. Implementation details of DPA attacks will not be presented here; a thorough discussion of the steps of the attack are given in [22] and [24].

Many successful implementations of DPA attacks have been published since Paul Kocher's initial paper on the topic. For example, a 1-bit, 4-bit and 8-bit DPA attacks on DES S-box are presented in [27] to extract the key input to the S-box. A more similar attack to this thesis work is presented in [26], where DPA was used to extract the internal state of the SHA256 algorithm implemented on a Xilinx Spartan 3 Field Programmable Gate Array (FPGA) using 4000 power traces.

2.1.3 Correlation Power Analysis Attacks

The main idea behind CPA attacks is correlating power (or EM) measurements with the hamming weight or hamming distance of the key guess. Much like DPA, the execution of a CPA attack requires collecting a large set of power consumption (or electromagnetic emanation) traces while the cryptographic device processes varying input data. The hamming weight of the key guess is correlated with each sample point in the traces to yield a set of correlation coefficients. The correct key guess is expected to have a significantly higher correlation coefficient than the other guesses.

Many methods can be used to calculate the correlation coefficient, however previous researches in the field has shown that Spearmans rank correlation is more powerful than Pearson correlation [2] [4] in side channel analysis. The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables. Pearsons correlation is used to assess linear relationship, while Spearmans correlation is used to assess monotonic relationships.

Spearmans rank correlation was used in the CPA attack described in [11]. That attack was successful at retrieving a byte of two internal registers in an implementation of HMAC-SHA-256 on an Altera FPGA.

2.1.4 Electromagnetic Emanation Analysis Attacks

Electromagnetic (EM) emanation analysis is very similar to power analysis methods discussed above; however instead of measuring the power consumption of the cryptographic device, EM emanation is measured. Similar to DPA and CPA attacks, EM emanations are measured while the device is performing the several encryption operations of different known plaintexts with an unknown secret key. Analyzing the recorded measurements may reveal parts of the secret key.

This type of analysis relies heavily on the profound principles of EM theory, which are outside the scope of this study. A brief simplification of those principles is described below:

1. An electrical current moving through a conductor generates a magnetic field around the conductor. The magnitude of the magnetic field is related to the magnitude of the current.
2. Changing magnetic flux induces an electromotive force in a closed loop conductor.

When an electronic device is operating and processing information, varying amounts of current flows in the internal circuity of the device. These currents induce a magnetic field around the device, the magnitude of which may be measured by a closed loop EM probe. This approach has been successfully used to attack cryptographic implementations of DES and RSA modular exponentiation [9]. Further, in [32] it was shown that EM attacks can be more precise that power attacks due to the fact that power attacks typically use global power consumption measurements of the device, whereas in EM attacks, localized regions of cryptographic chip may be targeted. Finally, a local EM side channel attack on an FPGA implementation of Advanced Encryption Standard (AES) was demonstrated in [5].

2.2 The HMAC-SHA256 Algorithm

This section contains an overview of the HMAC and SHA-256 algorithms. It will describe the computational steps for each of those algorithms.

2.2.1 Hash Based Message Authentication Code (HMAC)

HMAC is the standard for keyed hashing, providing message authentication, which was defined in the Federal Information Processing Standards (FIPS) publication 198 in 2002 [30]. One of the main goals behind the construction of this standard is to allow the use of available hash cryptographic functions and to allow the replicability of the underlying hash functions, should faster or more secure hash functions be defined. As such, any existing iterative hash function, such as Message Digest 5 (MD5), SHA-1 and SHA-2, may be used. The HMAC function is defined as follows:

$$MAC(m)_t = HMAC(k, m)_t = H((k \oplus opad), H(k \oplus ipad), m)_t$$

Where $H()$ is the hash function, m is the message, k is the key, $ipad = (363636)_{16}$ and $opad = (5C5C5C)_{16}$ are constants defined by the standard. The subscript t in the above equation denotes that only the most significant t bytes out of the result are extracted, such that $L/2 \leqslant t \leqslant L$, where L is the output length of the hash function, $H()$. The key used in HMAC must be B bytes long, padding is used as necessary. The message should be n-bits long where $0 \leqslant n \leqslant 2^B - 8B$, as defined by the standard. The HMAC algorithm consists of the following seven steps [19] [30]:

1. Key Pre-processing: $k = \begin{cases} k & k = B \\ k||(0..)_{B-K} & k < B \\ H(k)||(0..)_{B-L} & k > B) \end{cases}$
2. Compute: $(k \oplus ipad)$
3. Concatenate: $(k \oplus ipad)||m$
4. Compute Hash: $H((k \oplus ipad)||m)$
5. Compute: $(k \oplus opad)$
6. Concatenate: $(k \oplus opad)||H((k \oplus ipad)||m)$
7. Compute Hash: $H((k \oplus opad)||H((k \oplus ipad)||m))$

2.2.2 Secure Hash Algorithm-2 (SHA-2)

The SHA-2 is a family of hash functions consisting of four algorithms, namely, SHA-224, SHA-256, SHA-384 and SHA-512. This family of functions was proposed as the Secure Hash Standard (SHS), in FIPS Publication 180-3 in 2008 [31], for computing a condensed representation of message data referred to as the message digest. The length of the message digest for SHA-x is x-bits, for example SHA-256 computes a 256-bit message digest for any valid input. A valid input for SHA224/256, is any message with length less than 264 bits, whereas a valid input for SHA384/512 is any message of length less than 2128 bits.

The execution of SHA-2 algorithms is done in two parts: Pre-processing, which consists of padding the message, parsing the padded message and setting initial hash values, and performing the hash computation. An outline for those two stages is presented below for the SHA-256 algorithm:

Pre-processing:
The input message of length L is split into N blocks of 512 bits. The input blocks are denoted as $M^{(1)}, M^{(2)}, ., M^{(N)}$. If L is not a multiple of 512, padding is performed as follows: bit 1 is appended to the end of the message, followed by k 0 bits, where k is the smallest non-negative solution to the equation $L + 1 + k \equiv 448 \bmod 512$. Finally, a 64-bit block representing the message length, L, is appended to have a final message length that is a multiple of 512. The pre-processing stage also includes setting the initial hash value. The initial hash value for SHA-256 consists of eight 32-bit words $H_0^{(0)}, H_1^{(0)}...H_7^{(0)}$ which are defined by the standard in [31].

Hash Computation:
The SHA-256 hash computation stage makes use of the six logical functions shown below. In those equations, $ROR^n(x)$ is the bit-wise rotate-right operation on input x by n positions and $SHR^n(x)$ is the bit-wise right shift on input x by n positions. Additionally, the $\oplus$ operation represents the bit-wise XOR, $\wedge$ is the bit-wise AND, and $\overline{x}$ is the bit-wise complement of x. All addition (+) operations are performed modulo 2^{32}.

$$
\begin{aligned}
Ch(x,y,z) &= (x \wedge y) \oplus (\overline{x} \wedge y) \\
Maj(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\
\sum_0 (x) &= ROR^2(x) \oplus ROR^{13}(x) \oplus ROR^{22}(x) \\
\sum_1 (x) &= ROR^6(x) \oplus ROR^{11}(x) \oplus ROR^{25}(x) \\
\sigma_0(x) &= ROR^7(x) \oplus ROR^{18}(x) \oplus SHR^3(x) \\
\sigma_1(x) &= ROR^{17}(x) \oplus ROR^{19}(x) \oplus SHR^{10}(x)
\end{aligned}
$$

The variables involved in the hash computation are:

- The words of the message schedule: $W_0, W_1, ...W_{63}$
- Eight working variables: a, b, c, d, e, f, g and h
- Words of the hash value: $H_0^{(i)}, H_1^{(i)}, , H_7^{(i)}$. Initially, $H^{(0)}$, is defined by the standard, and is replaced by each successive intermediate hash value, $H^{(i)}$, ending in the final value $H^{(N)}$
- Two temporary words, T_1 and T_2, as defined below

For each message block $i, 1 \leqslant i \leqslant N$, the following four-step digest is performed:

1. Initialize the eight working variables:

$$
\begin{aligned}
&a = H_0^{(i-1)}, b = H_1^{(i-1)}, c = H_2^{(i-1)}, d = H_3^{(i-1)}, \\
&e = H_4^{(i-1)}, f = H_5^{(i-1)}, g = H_6^{(i-1)}, h = H_7^{(i-1)}
\end{aligned}
$$

2. Prepare the message schedule:

$$
W_t = \begin{cases} M_t^{(i)}, & 0 \leqslant t \leqslant 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & 16 \leqslant t \leqslant 63 \end{cases}
$$

3. For rounds $t = 0$ to $t = 63$

$$\begin{gathered} T_1 = h + \sum_1(e) + Ch(e,f,g) + K_t + W_t, \\ T_2 = \sum_0(a) + Maj(a,b,c), \\ h = g, \\ g = f, \\ f = e, \\ e = d + T_1, \\ d = c, \\ c = b, \\ b = a, \\ a = T_1 + T_2 \end{gathered}$$

4. Compute the ith intermediate hash value $H^{(i)}$

$$H_0^{(i)} = a + H_0^{(i-1)}, H_1^{(i)} = b + H_1^{(i-1)}, H_2^{(i)} = c + H_2^{(i-1)}, H_3^{(i)} = d + H_3^{(i-1)},$$
$$H_4^{(i)} = e + H_4^{(i-1)}, H_5^{(i)} = f + H_5^{(i-1)}, H_6^{(i)} = g + H_6^{(i-1)}, H_7^{(i)} = h + H_7^{(i-1)}$$

After all N blocks of message m are processed, the final message digest is the concatenation of the hash values:

$$SHA256(m) = H_0^{(N)}||H_1^{(N)}||H_2^{(N)}||H_3^{(N)}||H_4^{(N)}||H_5^{(N)}||H_6^{(N)}||H_7^{(N)}$$

2.3 Previous Research in HMAC-SHA256 Side Channel Attacks

HMAC is a very famous and widely utilized MAC scheme. As previously described, HMAC may be used with any cryptographic hash function. The cryptographic strength of HMAC is highly dependent on the cryptographic strength of the underlying hash function. In the past it was believed that as long as the underlying hash function utilized in HMAC is secure, the security of HMAC is guaranteed; however research efforts in the field have disproved that assumption. An analysis of the security of HMAC based on 12 cryptographic hash functions is presented in [29]. Their research shows that HMAC is vulnerable to DPA side channel attacks for 11 of the 12 underlying hash functions evaluated, despite the hash functions themselves being resistant to side channel attacks. Those striking results indicate that extensive security analysis of cryptographic systems is warranted, even if the system is believed to be secure.

In a separate study, the security of HMAC based on the SHA-2 hashing family was evaluated. In the research paper [26], an explicit DPA attack strategy for HMAC-SHA256 was presented. The strategy was capable of retrieving internal register states of the SHA256 algorithm when tested on an implementation of the algorithm on the Xilinx Spartan-3 FPGA board. The research paper goes on to suggest the use of random masking as countermeasures for the attack.

While some counter measures exist against side channel attacks on HMAC, they many not always be effective. In [8] a template attack, based on EM traces, against HMAC-SHA-1 is shown to work even with some countermeasures implemented in the hardware design. A template attack is a powerful type of side channel attack that works by first creating a *profile* of the device under attack, following that individual traces collected during an attack are compared to the profile to extract secret information. With a comprehensive profile of the device under attack, the template attack presented in the research paper was successful at retrieving secret information by monitoring a single execution of HMAC-SHA1.

More recently, a ranked CPA attack was executed on a single-cycle round implementation of the compressor part of HMAC-SHA256 on the Altera Cyclone II FPGA [11]. This attack proposed the use of a pre-averaging technique to speed up the attack time and reduced the number of required traces to successfully retrieve the secret key. Only 1024 traces were required for the correct key guess to be ranked 1st by the ranked CPA algorithm.

The HMAC-SHA256 implementation targeted utilizes hardware parallelism to achieve single-cycle rounds, meaning each of the 64 rounds of SHA256 is computed in a single clock cycle. The side channel analysis of this architecture is challenging due to the complexity and discrimination problem, as discussed in [11]. The targeted architecture is presented in [19] and will be further discussed in section 3.1.

2.4 Summary

This chapter introduced the background and related work to this thesis work. It began by introducing different types of practical side channel attacks, emphasizing on varying forms of power analysis attacks. Following that the HMAC and SHA256 algorithms, which are the targeted algorithms in this thesis work, were described. Finally, in section 2.3 previous research work on side channel attacks targeting HMAC was presented.

Chapter 3

Experiment Setup

This chapter contains the discussion about the implementation of the attack carried out in this research project. It starts by giving an overview of the attack, including the particular architecture and the hardware platform targeted, and the lab setup used to execute it. The lab setup overview is followed by sections which detail the hardware and software components involved in the attack. Finally, the attack procedure is presented along with the attack implementation challenges faced and the workarounds for those challenges.

3.1 Attack Overview

The HMAC and SHA256 algorithms were described in sections 2.2.1 and 2.2.2 respectively; references will be made to those sections in the description of the attack.

The attack attempts to retrieve the intermediate hash values,$H_0^1 ... H_7^1$, at the end of the first iteration of the algorithm described in the Hash Computation part of section 2.2.2. The first iteration of the algorithm processes a function of the 512-bit key material, specifically it is the key exclusive-ored with the ipad (see step 4 of the HMAC algorithm described in section 2.2.1). The intermediate hash values at the end of this iteration,$H_0^1 ... H_7^1$, are then used throughout the remainder of the SHA256 algorithm to compute the hash.

The second block to be processed by SHA256, referred to as $M^{(2)}$ in the pre-processing stage, contains the first 512 bits of the input message, m. The secret intermediate hash values from the first message block,$H_0^1 ... H_7^1$, are initialized into the eight working variables, $a - h$, as can be seen from step 1 of the SHA256 Hash Computation algorithm described in section 2.2.2, and used to compute the hash values for the second iteration of the algorithm. This process is repeated for all remaining input blocks until the final hash, $H^{(N)}$, is computed. Therefore, knowing the intermediate hash values at the end of the first, key material, iteration allows for the generation of MAC tags for varying input messages.

In order to retrieve the intermediate hash values, measurements from the cryptographic device must be taken at a specific attack point. The next section will present the architecture of the SHA256 hardware block implemented on the cryptographic device and specify the attack point.

3.1.1 SHA256 Single-Cycle Round Architecture

As mentioned in section 2.3, a single-cycle round complete implementation of the entire SHA256 hardware core will be targeted in this project. Specifically, the design presented in [11] implemented only the compressor part of SHA256. Figure 3.1 below shows the hardware architecture of the SHA256 core[11], which can be divided into four main blocks: Intermediate Hash Computation, Compressor, Constant Memory and Message Scheduler. The rectangles in the diagram represent 32-bit registers, and the Constant Memory block is used to store the sixty-four constant words $K_0 ... K_{63}$ defined by the standard.

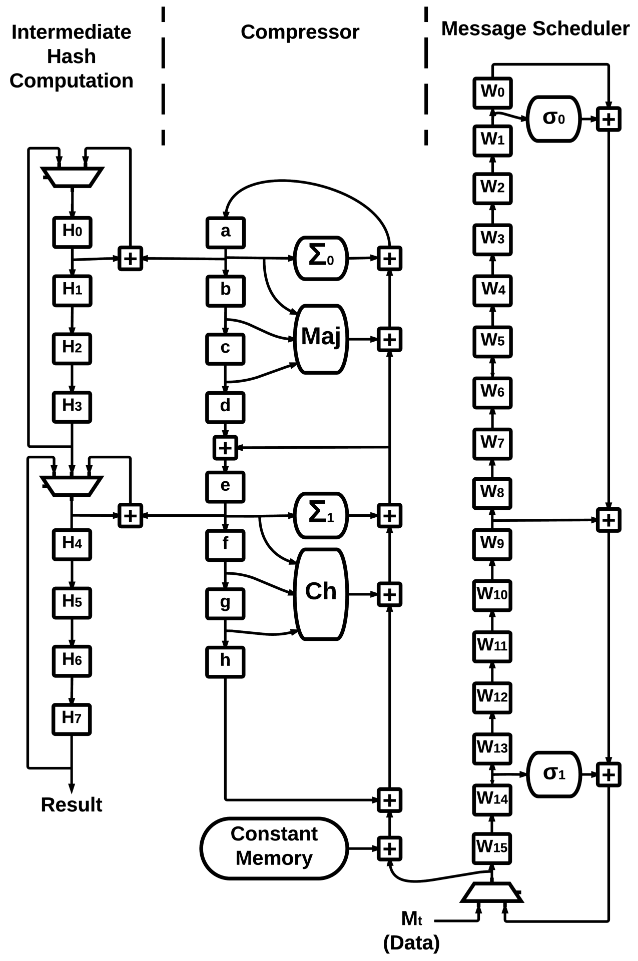


Figure 3.1: SHA256 Single-cycle Round Architecture

The attack point on this architecture is the instant of time at which the a and e registers are updated in the first round (out of the 64 rounds performed in part 3 of the algorithm described in section 2.2.2) during the processing of the second block, $M^{(2)}$, which is the first message block following the key material block. The process of updating a register consumes a marginally varying amount of power, depending on the hamming weight of the value being written [25]. This is a side channel that can leak information about the data values being written to registers a and e.

The values written to the a and e registers at the specified attack point are a function of the intermediate hash values from the previous, key material, round and the input plaintext message. Oscilloscope traces are recorded at the specified attack point with varying input plaintext messages. The power traces are then analyzed to determine the values of the a and e registers, 8-bits at a time. In other words, at least four attack rounds are required to retrieve the full 32-bit values of the a and e registers, as described in [11].

Once the values of the registers a and e are known, the values of the remaining working variables of SHA256, namely b, c, d, f, g and h may be calculated. This is due to the fact that, as can be seen in Figure 3.1 and in the SHA256 algorithm in section 2.2.1, that the value in register a is shifted into registers b, c and d in subsequent rounds of the algorithm. Similarly, the value of register e is shifted into registers f, g and h in subsequent rounds. Figure 3.2 shows the relationship between the SHA256 working variables across the rounds of the algorithm via the vertical arrows.

In Figure 3.2, the circled variables may be determined using the CPA attack described above. The variables inside the square may be calculated based on the values of the circled variables; the approach to calculate those values is explained in [11]. For example, the sixth column from the left illustrates that $e4 = f5$ and that $d3 = c2 = b1$. Using this approach and multiple iterations of the CPA attack, the values of variables $a1 - h1$ may be determined and used to forge a MAC tag for any plaintext message.

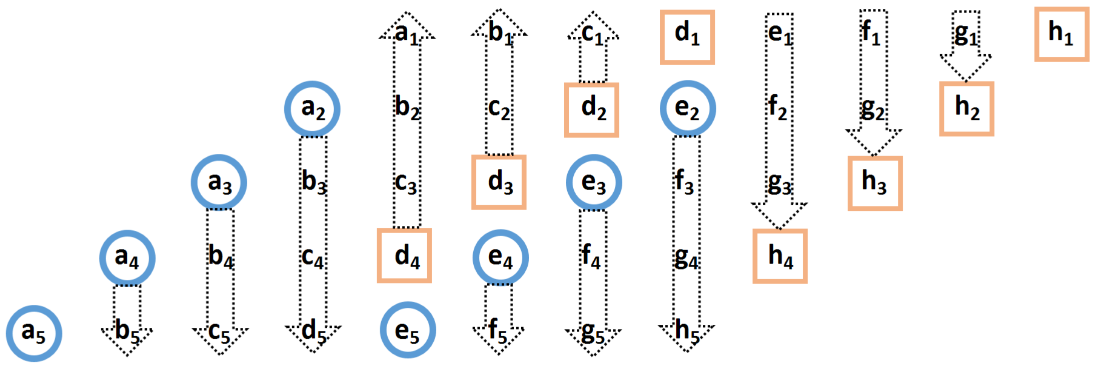


Figure 3.2: SHA256 Intermediate Registers Shifting

3.2 Lab Setup Overview

This section presents an overview of the lab setup required to perform the attack described in section 3.1. Figure 3.3 below shows a system level diagram of the lab setup used in the experiment. The components of the system are as follow:

1. *Host Personal Computer (PC):* Used to program the hardware design to be loaded onto the FPGA board, and to analyze the traces collected by the oscilloscope. Any modern PC with a Universal Serial Port (USB) port may be used. The software components of the setup will be discussed in section 3.4.

2. *Xilinx Platform Cable USB II:* [17]: Used to download the design onto the Programmable Read Only Memory (PROM) on the Xilinx FPGA board.

3. *SASEBO-GII:* [28] The FPGA board used in the experiment. This board was designed for the purpose of side channel attack experiments and provides an easy interface to measure the power consumption of the cryptographic FPGA. More details on this platform will be provided in section 3.3.

4. *Tektronix TDS7254 oscilloscope:* a 4-channel, 2.5 GHz oscilloscope with a maximum sampling rate of 20 Giga Samples per Second (GS/s) [14]. This oscilloscope was used

to collect EM and power traces from the cryptographic FPGA board. The *FastFrame* feature was used to collect multi-trace acquisitions. Three probes were used in this experiment:

- Tektronix P6139B Passive Voltage Probe [15]: Used to detect the trigger signal.
- Rohde and Swartz (R&S) HZ-15 Electromagnetic Probe with R&S HZ-16 amplifier [33] [34]: Used to collect EM emanations from the FPGA chip for the EM attacks.
- SMA to BNC Passive Voltage Probe: Used to collect the FPGA power consumption measurements from on-board SMA connector for power measurements; more details on this will be provided in section 3.3.2.

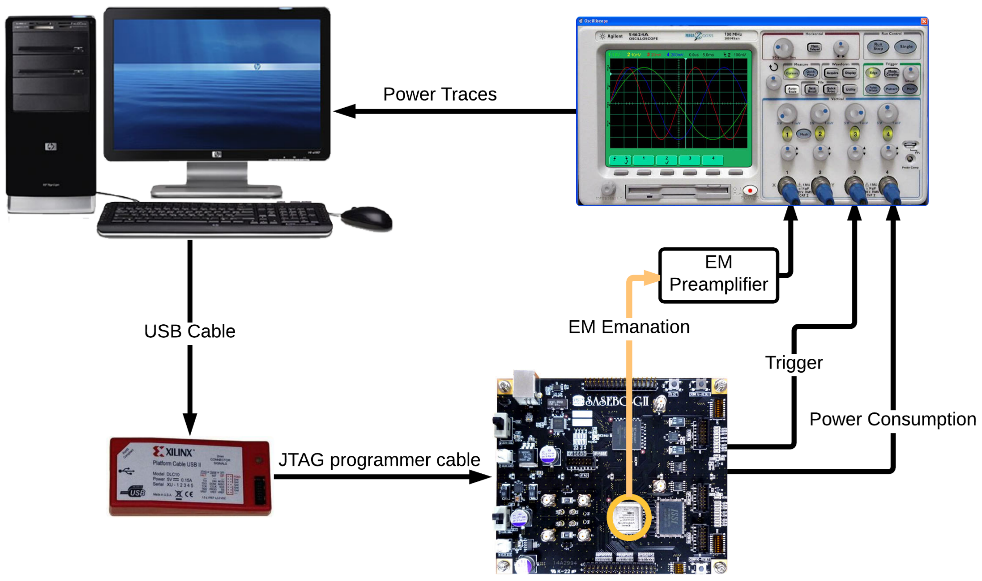


Figure 3.3: Lab Equipment Setup System Diagram

3.3 Hardware

This section describes the hardware platform used in the experiment. It begins by describing the SASEBO-GII board and discussing its relevant features to the attack. Following that, the board's power circuit will be presented in order to clarify where the power consumption measurements for the cryptographic core are taken from. Finally, the setup of the trigger signal generation on the FPGA board will be described.

3.3.1 SASEBO-GII Platform

The Side channel Attack Standard Evaluation Board (SASEBO-GII) is an FPGA board designed and developed by the Japanese National Institute of Advanced Industrial Science and Technology [28]. The board was designed with the goal of making it a standardized hardware platform for security evaluation of various cryptographic systems. As such, the board was designed with features, such as configurable power systems and shunt resistors for power measurements, that are meant to simplify executing side channel experiments.

Figure 3.4 shows an annotated top view of the SASEBO-GII board. The relevant features of this board to the attack are:

- Two Xilinx FPGAs:
  - Cryptographic FPGA (Virtex-5 series): XC5VLX30-1FFG324 [18]
  - Control FPGA (Spartan-3A series): XC3S400A-4FTG256 [16]
  - A configurable 38-bit general-purpose bus connecting the two FPGAs, to allow for communication and data transfer between the two FPGAs and the host PC.
- A 1 Ω shunt resistor on the cryptographic FPGA power supply rail. Resistor part number RK73BW2HTTD1R0J supplied by Koa Speer Electronics was used [13].
- An SMA connector, tied to the VCC_{INT} rail, for taking power consumption measurements
- On-board power regulators that convert 5V from an external source into low-noise 3.3 V, 1.8 V, 1.2 V and 1.0 V power rails to be used by various components on the

board. The cryptographic FPGA core voltage of 1.0 V can also be directly supplied by an external power source. On-board switches allow dynamic configuration of the power supply network.

- USB connector and a USB controller IC to allow communication between the hose PC and the control FPGA.

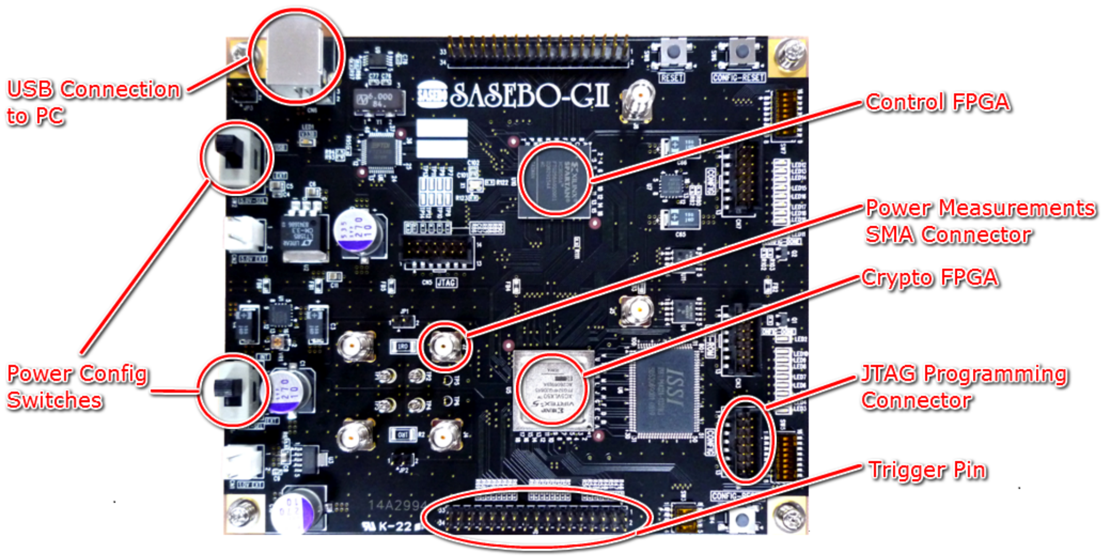


Figure 3.4: Top Level View of the SASEBO-GII Board

The SASEBO-GII board was first used in the DPA Contest v2 [1], an academic contest for evaluating the effectiveness of different Differential Power Analysis attacks on Advanced Encryption Standard (AES-128) implemented on an FPGA platform. The contest organizers had published Verilog source files for the control and cryptographic FPGAs on the board. The control FPGA contains the necessary blocks to enable communication with the host PC (through the the on board USB IC) and the cryptographic FPGA (through the local general purpose bus). The cryptographic system under attack (AES-128 in the case of the DPA contest) is to be implemented on the cryptographic FPGA.

The design published with the DPA Contest was used as the starting point for the hardware implementation in this thesis work. In particular, the implementation of the control FPGA was used as is and the cryptographic core was modified to implement SHA256 instead of AES-128.

The design on the cryptographic core consists of the following modules; for each module a brief description of the changes made to adapt the design to the SHA256 core is give:

- **Top Level Module** (*chip_sasebo_gii_aes_comp.v*): Instantiates other modules in the design and makes necessary connections between them. In modifying the design to work with SHA256, the previously instantiated AES module was removed and replaced with an instance of SHA256. The SAH256 module was designed to use the same port names as the original AES module implementation, to simplify swapping the two modules.

- **Cryptosystem Definition** (*AES_comp.v*): This modules defines the high-level interface of the cryptosystem that runs on the cryptographic FPGA, and all its necessary sub-modules. In the case of AES, sub-modules were defined for *Sbox, MixColumns* and *SubByes* among others such as *AES_Encryption* and *AES_Decryption*. All the modules that define the AES cryptosystem were removed and replaced with ones that define the single-cycle round implementation of SHA256.
  It is important to note that the high-level interface of the SHA256 module is kept the same as that of the AES module to simplify the design work; both modules share the same input and output port names. The two lines below show the module declarations for AES and SHA256, respectively:

  ```
  module AES_Comp(Kin, Din, Dout, Krdy, Drdy, EncDec, RSTn, EN, CLK, BSY, Kvld, Dvld);
  module SHA_256(Kin, Din, Dout, Krdy, Drdy, EncDec, RSTn, EN, CLK, BSY, Kvld, Dvld, Ftrig, RegIO);
  ```

  Among the ports defined in the module declarations, Kin, Din and $Dout$ are 128-bit data ports used for key input, data input and data output, respectively. Key and data inputs come from the board interface application, which will be discussed in the subsequent section, and the data output is sent back to the board interface

application. The remaining signals are all control signals used to control the logic flow of the cryptosystem; for example $RSTn$ is a reset signal. Most of the control signals are relevant to both AES and SHA256 modules.
The $EncDec$ signal was originally used by the AES core to distinguish between encryption and decryption operations. That signal is no longer needed in the SHA core, however the port is left unconnected to simplify the migration from AES to SHA256.
Aside from the original data and control signals in the AES module declaration, the $Ftrig$ signal was added to the SHA256 module to be used as the trigger signal. The signal is driven from within the module and is set to transition at the specified attack point. Finally. the $RegIO$ port is used in the *Off-chip* attacks, as described in chapter 4, to bring the values of the intermediate a and e variables from the SHA256 algorithm to off-chip I/O ports. A complete code listing of the SHA256 module is provided in Appendix B.

- **Local Bus** (*syncfifo_8x2047.v*): A synchronous First In First Out (FIFO) bus module. It implements the physical layer (registers and wires) of the local bus used to communicate between the two FPGAs on the board. This module was left unchanged in adapting the design to SHA256.

- **Local Bus Interface** (*lbus_if.v*): This module defines the local bus interface for the cryptosystem on the FPGA. It defines the communication standard and ties the cryptosystem to the local bus. This module was left unchanged in adapting the design to SHA256.

- **Local Bus Manager** (*ctrl_lbus.v*): This modules defines the manager for the local bus. It maintains the state of the bus and drives the control signals to control the data flow through the bus, such as *bus_full*. This module was left unchanged in adapting the design to SHA256.

3.3.2 Board Power Circuit

When collecting power consumption measurements for the purpose of side channel analysis, it is important to identify a suitable location to probe the power circuit such that only the consumption of the cryptographic device is measured. This is especially true for the SASEBO-GII board used in this study, since the board has two FPGAs and many other IC chips. In this case, measuring the total power consumption of the board would likely yield noisy power traces, which makes it difficult to identify the marginal power consumption changes due to the hamming weight of the values loaded into the a and e registers, as described in section 3.1.1.

Figure 3.5 below shows a block diagram of the configuration of the power circuit used in the experiment. In particular, it shows that the board was powered by a 5.0V USB interface (selected by switch SW2). A network of voltage regulators is then used to step down the 5.0V into 3.3V, 2.5V, 1.2V and 1.0V power rails to be used to power different components on the board.

The Xilinx XC5VLX30-1FFG324 (cryptographic FPGA) has a 1.0V, 2.5V and a 3.3V rails connected to its VCC_{INT}, VCC_{AUX} and VCC_O, respectively. A brief description of the different power rails is provided below:

- $\mathbf{VCC_{INT}}$: Voltage supply for the internal core logic, such as Configurable Logic Blocks (CLBs) and block Random Access Memory (RAM).
- $\mathbf{VCC_{AUX}}$: Voltage supply for auxiliary logic, such as clock manager circuits, dedicated configuration pins and various other auxiliary circuits.
- $\mathbf{VCC_{O}}$: Voltage supply for the output buffer drivers of Input/Output (I/O) banks,

The FPGA draws power from the internal supply voltage rail, VCC_{INT}, to run its internal circuity while it is processing data. Therefore VCC_{INT} is the power rail that leaks side channel information about the data values being loaded to internal registers, thus it is the rail from which power consumption measurements will be taken in this experiment.

Measuring the power consumption of the cryptographic FPGA is made easy on the SASEBO-GII board, since it is the only component connected to the 1.0V rail. As shown in Figure 3.5, the 1.0V rail is supplied by the on-board voltage regulator, $U1$, and may be measured through Test Point 2 (TP2). Appendix A contains a section of the SASEBO-GII board schematic that shows in more details regulator U1, switch SW1, the shunt resistor and the location of TP2.

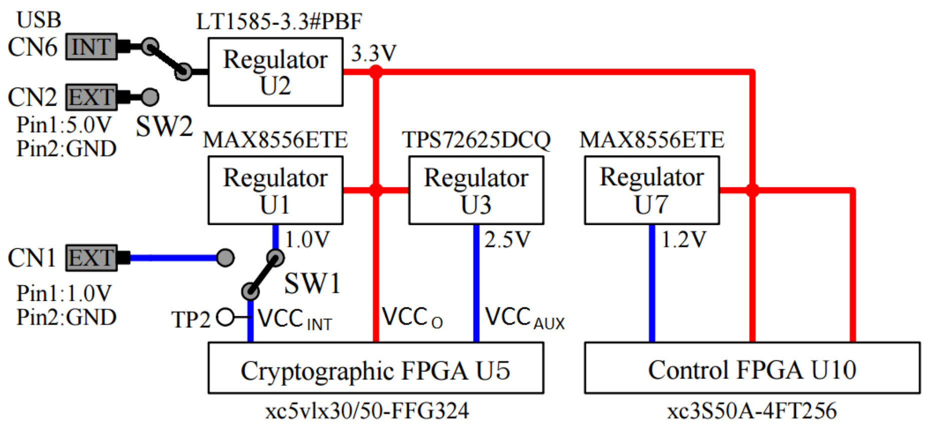


Figure 3.5: SASEBO-GII Board Power Distribution

3.3.3 Trigger Setup

In order to capture the power consumption at the attack point, a trigger signal is used. The signal is high by default, and is cleared and reset one clock cycle prior to loading the a and e registers. The cryptographic FPGA outputs the trigger signal on one of the I/O pins on the SASEBO-GII board, and the Tektronix P6139B voltage probe is used to capture the signal. This signal is fed into the oscilloscope to indicate when it should capture power (or EM) measurements, in other words this signal defines the start of the attack point.

Figure 3.6 below shows an oscilloscope capture of the trigger and power consumption signals. The trigger signal, shown in yellow, goes low for one clock cycle, two clock cycles prior to loading new values into the registers. The power consumption signal, shown in green, has very small fluctuations that are the result of writing different values to registers. The trigger is intentionally set to go low a clock cycle prior to loading the registers to isolate the power fluctuation caused by driving the trigger signal from the power fluctuation caused by loading the registers.

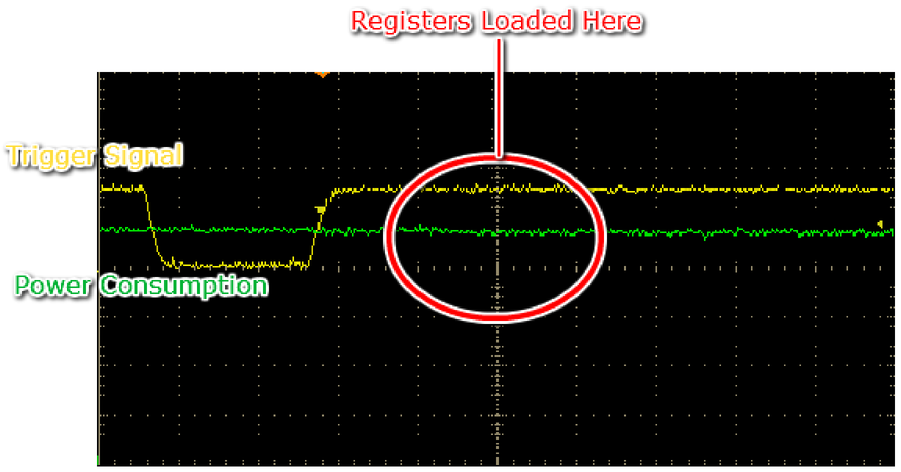


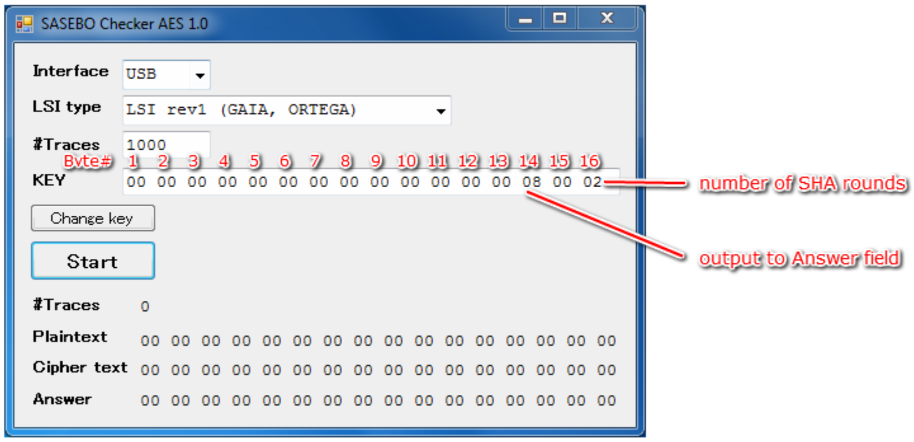
Figure 3.6: Oscilloscope Capture Showing the Trigger Signal and the Time at which Registers a and e are Written

3.4 Software

This section introduces the software components used in executing the attack. The major components will be described in separate subsections. All the software was executed on the host PC used for the attack.

3.4.1 Board Interface Application

The attack requires capturing many power traces from the cryptographic FPGA while it processes varying input plaintext messages. In particular, one byte of the input plaintext message must range in values $0x00$ to $0xFF$, 0x indicating hexadecimal values, while all other bytes of the message remain constant. A C# application running on the host PC was used for two way communication with the cryptographic FPGA. The application allowed sending varying plaintext messages to the FPGA and receiving data back from it. The user interface of the board interface application is shown in Figure 3.7 below.



```
Interface    USB
LSI type     LSI rev1 (GAIA, ORTEGA)
#Traces      1000
Byte#        1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
KEY          00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 02

#Traces      0
Plaintext    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Cipher text  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Answer       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 3.7: Board Interface Application User Interface

This application was initially designed to be used by the DPA Contest v2 [1], an academic contest for evaluating the effectiveness of different Differential Power Analysis attacks on Advanced Encryption Standard (AES-128) implemented on the SASEBO-GII platform. The application was modified to be used for the current experiment by changing the functionality of the *KEY* input field. In particular, this field was no longer used to input a key into the AES-128 algorithm, rather only two bytes of it were used as follows:

- *Byte 14:* This byte was used as the select input of a multiplexer in the hardware design that chooses which of the SHA256 intermediate working variables, $a - h$, should be sent to the board interface application. The selected variable is displayed in the *Answer* field on the application. This functionality was needed to verify that the SHA256 hardware design was functional in the design phase.

- *Byte 16:* This bye was used to determine the number of rounds of SHA256 to be executed. The attack carried out in this experiment targeted the first round of computation in the first message block following the key material block. During the attack phase, the number of SHA256 rounds is kept to two in order to limit the power consumption of the board and reduce noise. Limiting the computation to two rounds also has the added advantage of not raising the FPGA temperature rapidly, which may affect the power consumption of the device.

3.4.2 Correlation Analysis Scripts

The CPA attack for this experiment was implemented in a MATLAB scripts. The implementation of the scripts follows the algorithms described in [11]. In fact, this experiment made use of the same MATLAB scripts that were used in writing the research paper; in particular, two scripts were used:

- *proc3_multidc_MOD2(DAT_file, HDR_file):*
  This is the pre-averaging script. It accepts an oscilloscope acquisition header file (*HDR_file*) and data file (*DAT_file*). The header file contains information about the contents of the acquisition such as the number of acquired data points, time between data samples and the number of traces contained within the acquisition; whereas the data file contains the actual power or EM measurements. The number of traces within an acquisition must be a multiple of 256, since traces are collected for varying one byte of the input from 0x00 to 0xFF, which is 256 distinct values. The pre-averaging script outputs a compressed acquisition files set (*DAT_compressed* and *HDR_compressed*) that averages out all the traces in the input files into one set of 256 traces.

Figure 3.8 below shows a pictorial representation of the function of the pre-averaging script. In the figure, output trace # 1 is the average of input traces $1, 257, 513...(N-1) \times 256+1$. Performing the pre-averaging in this manner has proven to significantly reduce the runtime of the *SHA256_HW_RankCPA* script and improve the results.

- *SHA256_HW_RankCPA(DAT_file, HDR_file):*
  This is the ranked correlation analysis script. It processes an acquisition file set, whether it is compressed by *proc3_multidc_MOD2* or not, and determines the rank of the correct guess for one byte of the a register and one byte of the e register.
  In a practical attack scenario, the script outputs a configurable number of guesses with the highest correlation ranks; those guesses can then be verified to determine which of them is the correct guess. In the experimental setup, however, the correct bytes of the a and e registers are known beforehand and are used by the script to compute the rank for the correct guess, which simplifies the output.

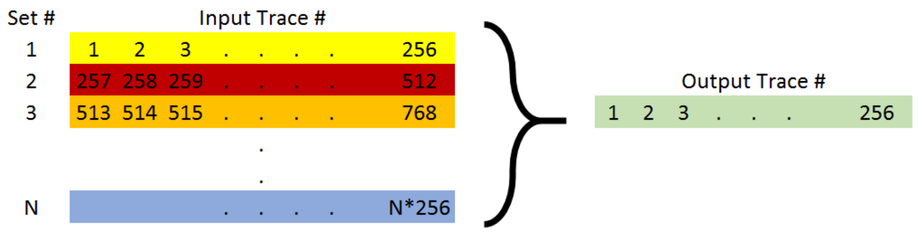


Figure 3.8: Pictorial Representation of the Pre-averaging Script Function

3.4.3 Xilinx Design Tools

Since the attack was executed on a Xilinx FPGA, some Xilinx design tools were used. In particular, the following tools were used:

- **Xilinx Integrated Synthesis Environment (ISE) 12.4**: Used as the development environment, it comes with synthesis and analysis tools for the Hardware Description Language (HDL) designs. The Register Transfer Level (RTL) schematic viewer built into the development environment was used to view the generated schematic of the design.

- **ISE iMPACT**: This tool is used to download a bitstream (generated by the ISE) onto the FPGA chip via the Xilinx Platform Cable.

- **PlanAhead 12.4**: This tool was used to view the resource utilization of the entire design and specific modules within. The utilization figures reported in Table 4.1 were obtained using this tool

3.5 Complete Attack Procedure

This section describes the procedure of executing the Correlation Power Analysis attack on the cryptographic FPGA of the SASEBO-GII platform.

Before executing an attack, preliminary experiments were carried out to determine the most suitable oscilloscope configuration. In particular, a configuration needed to be found that allows capturing the maximum number of traces per acquisition, while collecting at least 50 samples per FPGA clock cycle. The FPGA runs on a 24 MHz clock source, i.e. the clock period is approximately 42 ns.

Table 3.1 shows the candidate oscilloscope configurations that were considered. The horizontal scale parameter was swept from 5 ns to 80 ns; for each setting, the record length was kept to the minimum value allowed by the scope, and the resulting maximum trace count (that is a multiple of 256) was recorded. Further, the table shows the number of FPGA clock cycles captured by each horizontal scale setting, the resulting samples per clock and the scope acquisition time.

Table 3.1: Candidate Oscilloscope Configurations

Horizontal Scale (ns)	**Min. Record Length**	**Max. Trace Count**	**Clock Cycles Captured**	**Samples Per Clock**	**Acquisition Time**
5	500	5120	1.2	416	2 min 50 sec
10	1000	4608	2.4	416	2 min 30 sec
20	500	14592	4.8	104	8 min 7 sec
40	500	14592	9.6	52	8 min 5 sec
80	500	14592	19.2	26	8 min 6 sec

The most suitable configuration can be achieved with a horizontal scale of 20 ns; which captures 4.8 clock cycles at 105 samples per clock period. Further, this setting allows capturing 14592 traces per acquisition in approximately 8 minutes.

With that in mind, the list below explains the steps carried out to execute the attack; following the list a discussion of the issues faced during the experiment will be presented.

1. Download a design of the SHA256 single-cycle round architecture onto the cryptographic FPGA (Virtex-5) on the SASEBO board.
2. Run the board interface application and set two bytes in the *KEY* field as follows: Byte 14 = 08, byte 16 = 02. Byte 14 sets the output of the hardware multiplexer to the *a* working variable of SHA256 and byte 16 limits the number of rounds to be executed in the SHA256 algorithm. Further, set the # *Traces* field to 14592.
3. Configure the Tektronix TDS 7254 Oscilloscope as shown in Table 3.2.
4. Start sending plaintext messages to the cryptographic FPGA by pressing *Start* on the board interface application.
5. Wait for the scope to finish capturing 14592 traces; save the acquisition using the MATLAB format (two files: *acq.dat* and *acq.hdr*) and move the acquisition files to host PC.

6. Run the MATLAB pre-averaging script on the acquisition files. The script outputs two files (*compressed.dat* and *compressed.hdr*)

7. Run the MATLAB CPA script on the compressed files. This script will output three metrics: the correlation coefficient of the correct guess, the maximum correlation coefficient calculated and the rank of the correct byte.

Table 3.2: Tektronix TDS 7254 Setup for the Attack

Name	**Value**	**Description**
Horizontal Scale	20 ns/div	Results in a 200 ns window size. Can display 4.7 clock cycles (of 42 ns each)
Ch 4 Vertical Scale	200 mV/div	Ch 4 is the power measurement channel
Ch 1 Vertical Scale	2 V/div	Ch 1 is the trigger channel
Trigger Edge	Positive Edge	Triggers when the signal goes from low to high
Trigger Level	1.6 V	Roughly the half way point of a 3.3V signal
Trigger Mode	Single Mode	Capture a single (FastFrame) acquisition
Record Length	500	500 samples per 4.7 clock cycles in the capture window. Roughly 105 samples per clock cycle
FastFrame: Number of Traces	14592	Maximum allowed number of traces that's a multiple of 256

3.6 Unique Challenges Faced

During the design and setup of this experiment, some issues were faced that slowed down the progress and affected the results at times. The major issues faced can be split into two categories: Hardware/Software used and oscilloscope measurements.

The main goal of this experiment was to verify whether the CPA attack with pre-averaging presented in [11] may be used to retrieve bytes of the intermediate secret variables of SHA256 implemented on a low power FPGA. The low power FPGA chosen was the Xilinx Virtex-5 found on the SASEBO-GII board.

The SASEBO-GII board was originally used in the DPA contest version 2 [1]. As a part of the contest, the organizers published hardware designs for both the cryptographic and control FPGAs on the board, as well as software components to enable communication with the board.

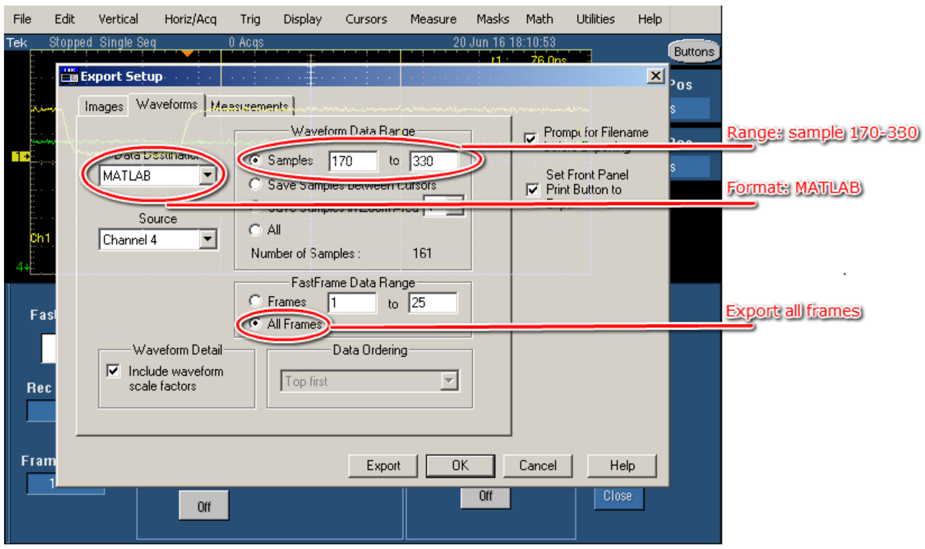
In an effort to minimize the setup time for this experiment, the hardware designs and software from the DPA contest were used as a starting point for the development of the attack platform. In order to do so, the implementation on the cryptographic core was changed from AES to the single-cycle round implementation of SHA256. In addition, functional changes to the board interface software were made, as discussed in section 3.4.2. Those changes proved to be more time consuming than originally thought, especially changes to the hardware due to the complex nature of the initial implementations published for the DPA contest.

As for the oscilloscope, ideally the scope used would be able to capture a time window equivalent to one clock cycle on the cryptographic FPGA with a high sample rate and a high trace count per acquisition. In reality, since the scope has limited resources, a balance needed to be struck between those variables to ensure getting good acquisitions.

As described in the previous section, the most suitable horizontal scale was deemed to be 20 ns/div due to scope limitations, which captures 4.7 clock cycles of the cryptographic FPGA. This presents a problem since the attack point, as described in section 3.1.1, is the instant of time at which the internal registers are loaded. Referring to Figure 3.6 in section

3.3.3, the attack point is marked by the red circle. The figure shows that the attack point makes up a small portion of the capture window. Having a capture window significantly larger than the attack point negatively affected the results of the CPA MATLAB script.

This issue was overcome by using the *Waveform Data Range* feature on the oscilloscope acquisition export setup. This feature, as the name suggests, allows exporting a specific range of the trace samples rather than the entire 500 samples. Figure 3.9 below is a screen capture from the oscilloscope showing the controls for the Waveform Data Range feature. Note that the figure shows that samples 170-330 of all 14592 traces will be exported in MATLAB format.



Data Destination: MATLAB
Source: Channel 4
Waveform Data Range: [✓ Samples] 170 to 330; Save Samples between Cursors; Save Samples in Zoom Area; All
Number of Samples : 161
FastFrame Data Range: Frames 1 to 25; [✓ All Frames]
Waveform Detail: [✓] Include waveform scale factors
Data Ordering: Top first
[✓] Prompt for Filename
[✓] Set Front Panel Print Button to

Figure 3.9: Tektronix TDS 7254 Acquisiton Export Window Showing the 'Waveform Data Range' Feature

Another issue faced with the oscilloscope is the maximum number of traces in a Fast-Frame acquisition. The scope memory only allows for 14592 traces, however more traces are needed for the correlation analysis to yield good results.

This issue was mitigated by capturing multiple acquisitions and merging them together before executing the correlation analysis. A Python script was developed to merge acquisitions by concatenating the data (.dat) files and modifying the header (.hdr) files to reflect the number of traces obtained by merging multiple acquisitions.

3.7 Summary

This chapter described the lab setup required to carry out the side channel analysis attacks on the SASEBO-GII platform. It began by giving an overview of the attack and the targeted architecture. Following that, sections 3.3 and 3.4 described the hardware and software components involved in the attack, respectively. In describing the hardware component, target board power distribution circuits were analyzed to determine where to collect measurements from for the power attacks. Finally, section 3.5 described the procedure of executing an attack, and in section 3.6 some of the challenges faced and their workarounds were presented.

Chapter 4

Experiments and Results

This chapter presents details of the attacks carried out in this thesis work. It begins by presenting the experimental values used in the attack on HMAC-SHA256 running on SASEBO-GII platform and defining the objective of the attack. Following that, results obtained in the four experiments are presented. Finally, a comparison is made between the results of this thesis work and a similar attack from the literature review.

4.1 Experimental Values

Throughout the experiments carried out in this research project, the same HMAC key (k) was used. The key consists of 16 32-bit words; the key is shown below in hexadecimal format:

00010203, 04050607, 08090A0B, 0C0D0E0F, 10111213, 14151617, 18191A1B, 1C1D1E1F, 20212223, 24252627, 28292A2B, 2C2D2E2F, 30313233, 34353637, 38393A3B, 3C3D3E3F

As per the HMAC algorithm, described in section 2.2.1, this key is exclusive-ored with the ipad constant defined by the standard, and the result is hashed with SHA256. The output of that operation is to be used as the starting hash value ($a - h$ working variables) for the subsequent block, which is the first block containing parts of the message (m).

Assuming the first word of the message schedule (W_1) consists of zeros, the values in the $a - h$ working variables at the end of the first round of SHA256 are precomputed to be:

$$\begin{aligned}
\mathbf{a} &= \mathbf{4D76180F}\\
b &= 27DAB238\\
c &= 1B9C94CA\\
d &= 754D32CB\\
\mathbf{e} &= \mathbf{E7BF36AD}\\
f &= D704401D\\
g &= B623BA92\\
h &= 3A50963B
\end{aligned}$$

From those 8 working variables, the attack attempts to retrieve the first byte in the a and e registers (in hex, 4D for register a and E7 for register e). Those values will be referred to as the correct guess in the subsequent sections.

4.2 Attacks

Side channel analysis experiments were executed on the single-cycle round implementation of HMAC-SHA256 on the cryptographic core of the SASEBO-GII board (the Xilinx Virtex-5 XC5VLX30). The FPGA resource utilization is shown in Table 4.1 below. In particular, the table shows that the complete design (the SHA core and the required communication blocks) uses up 14% of the FPGA's registers (87% of which are used solely by the SHA core) and 41% of the available Look Up Tables (LUTs) (96% of which are used solely by the SHA core). Those utilization metrics indicate that SHA core consumes most of the FPGA resources, hence it can be assumed that the power consumed by the FPGA chip is used primarily to run the SHA256 algorithm.

Table 4.1: FPGA Resource Utilization

	Registers			**LUTs**		
	Used	*Out of*	*Utilization*	*Used*	*Out of*	*Utilization*
Complete Design	2831	19200	14 %	7897	19200	41 %
SHA256 Core	2473	2831	87 %	7615	7897	96 %

Four attacks were carried out on the given architecture; two power attacks and two EM attacks. Each attack consists of collecting 145920 traces (with 500 samples per trace) using the Tektronix TDS7254 oscilloscope. Power measurements were made using a SMA to BNC passive voltage probe, while EM measurements were made using the Rohde and Swartz HZ-15 EM probe with the Rohde and Swartz HZ-16 preamplifier. The collected traces are then analyzed with the MATLAB scripts described in 3.4.3 to determine byte guess with the highest correlation coefficient. The four attacks will be referred to as:

- On-chip EM attack
- On-chip power attack
- Off-chip EM attack
- Off-chip power attack

The 'on-chip' attacks, in which the SHA256 intermediate variables a and e are stored in registers on the FPGA chip, were carried out first and yielded poor results. Therefore subsequently two additional attacks were carried out, in which the intermediate variables were sent off-chip via IO pins. Details about each attack and the results obtained will be presented in subsequent sections in this chapter.

It is important to note that in both on-chip and off-chip power attacks, power consumption measurements were taken from the VCC_{INT} rail, as described in section 3.3.2. In the case of the off-chip attack, the process of sending signals off-chip through I/O pins consumes some additional power from both the VCC_{INT} and VCC_{O} rails; however only the additional power through VCC_{INT} is measured.

The attacks carried out in this experiment are said to be successful if the correlation coefficient of the correct guess (Register a byte $1 = 4D_{16}$, register e byte $1 = E7_{16}$) ranks among the top 10 correlation coefficients, using up to 145920 oscilloscope traces. Ideally, the correct guess would be ranked first; however, even if it does not, it is computationally trivial to test the top 10 results to determine which of them is the correct guess.

4.2.1 On-chip EM attack

The baseline attack was chosen to be the on-chip EM attack. For this attack, the oscilloscope probes were connected as shown in Figure 4.1 below. The EM probe is placed directly above the center of the cryptographic FPGA chip, and a voltage probe is connected to the trigger pin. Before finalizing the experiment setup, different EM probe placements were investigated to determine the most optimal placement. This investigation lead to the conclusion that placing the probe directly above the center of the FPGA chip yielded the best results. A mechanical clamp was used to hold the EM probe throughout the experiment to avoid unwanted variability in the measurements due to the probe moving around.

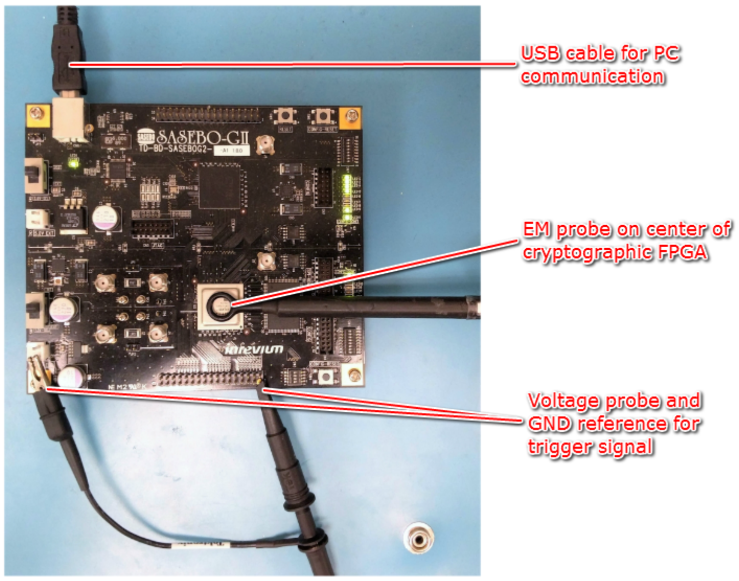

USB cable for PC communication
EM probe on center of cryptographic FPGA
Voltage probe and GND reference for trigger signal

Figure 4.1: Top View of SASEBO-GII Board Showing Scope Probe Connections for EM Attacks

Using this configuration, 145920 EM traces were collected. 10 separate oscilloscope acquisitions were needed, since the scope memory only allowed for 14592 traces per acquisition. All 10 acquisitions were collected consecutively at the same time, to avoid variability in the measurements due to the chip's temperature variation. Capturing a single acquisition on the oscilloscope took approximately 8 minutes, saving the acquisition to disk took approximately 40 seconds.

Figure 4.2 below shows a single frame from a typical acquisition on the scope. The yellow signal is the trigger, and green is the EM emanation signal. The figure shows that the acquisition consists of 14592 traces (as indicated in the blue ribbon at the top of the image).

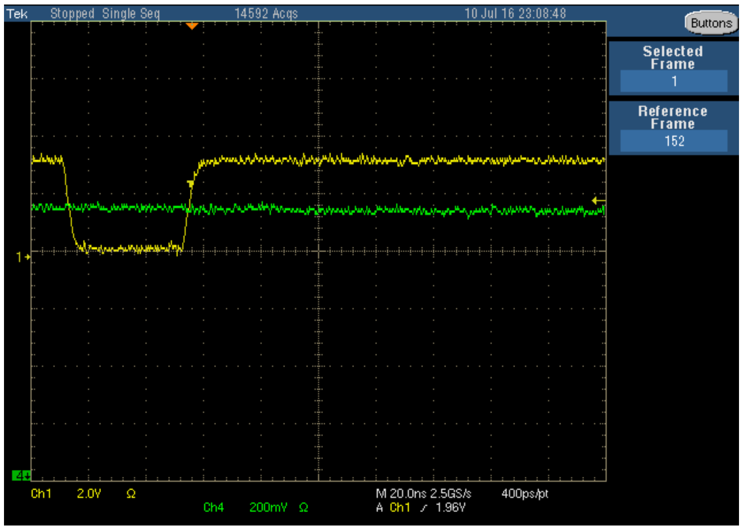


Figure 4.2: A Sample Oscilloscop Trace

The collected traces were analyzed in steps of 14592 traces. The analysis of each set of traces yields three values: The maximum correlation coefficient among all guesses, the correlation coefficient of the correct bytes guess and the rank of the correct guess coefficient among all 65536 possible guesses (256 guesses for each of register a and e byte 1). Figure 4.3 below summarizes the results of the analysis for the on-chip EM attack. On the graph, the left Y-axis is used to plot the rank of the correct guess (solid red line), while the right Y-axis is used to plot the correlation coefficient for the correct guess (solid blue line) and the maximum correlation coefficient attained in the analysis (dotted blue line).

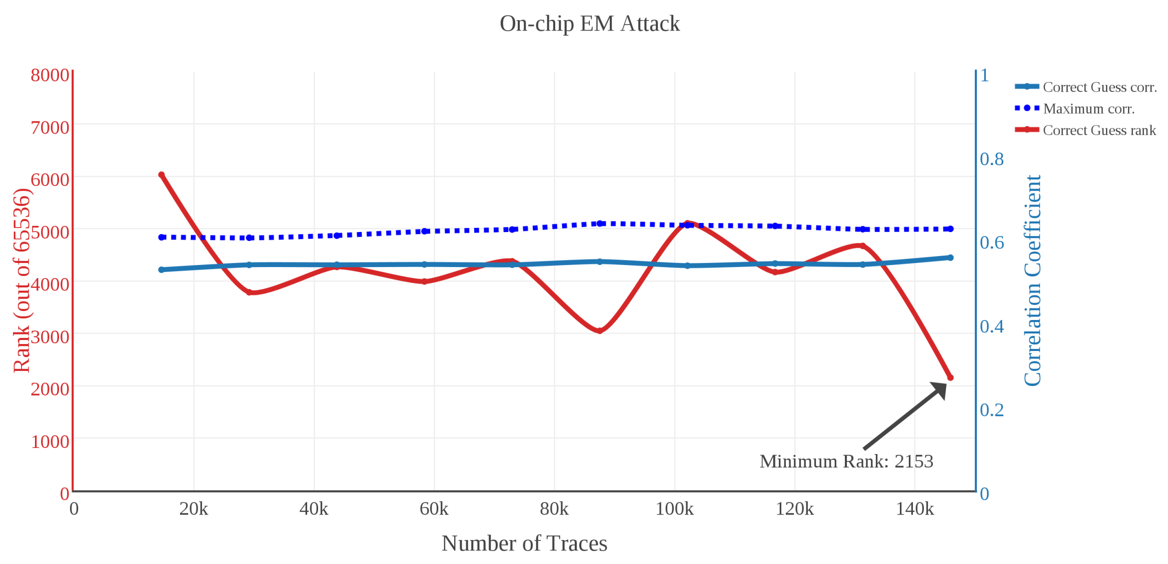


Figure 4.3: Results for On-chip EM Attack

The graph in Figure 4.3 shows that the attack failed; the minimum rank of the correct guess attained was 2153 (using all 145920 traces) meaning that 2152 byte guesses had a better correlation coefficient than the correct guess. Furthermore, the graph shows that the correlation coefficient of the correct guess is low, at approximately 0.52, and it does not increase with increasing the number of traces used in the analysis. A low correlation coefficient indicates that the side channel targeted in this experiment is weak.

4.2.2 On-chip power attack

Having discovered that the side channel from the electromagnetic emanation of the Xilinx Virtex-5 FPGA is weak, a second attack was executed to determine whether a stronger side channel exists through the FPGAs power supply rail. An SMA to BNC passive voltage probe was used to measure the power consumption of the FPGA via the J2 socket on the SASEBO-GII board. Figure 4.4 below shows where the oscilloscope probes are connected for the power attack. A typical frame of the acquisition looks similar to that in Figure 4.2.

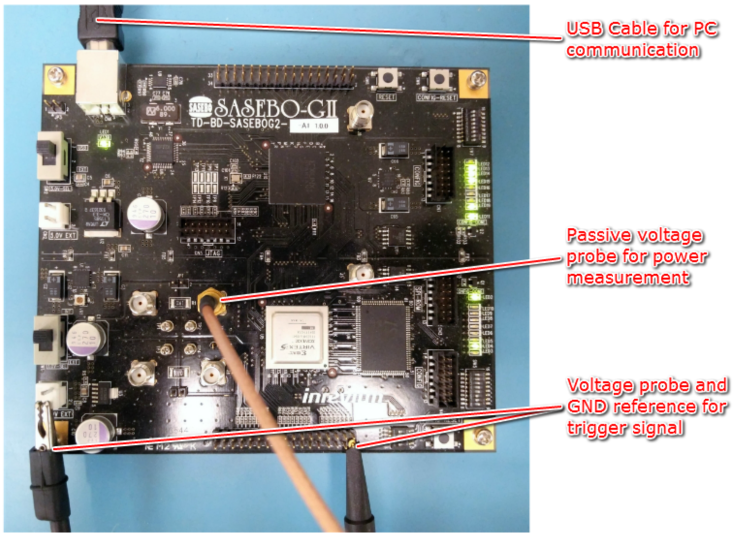

USB Cable for PC communication
Passive voltage probe for power measurement
Voltage probe and GND reference for trigger signal

Figure 4.4: Top View of SASEBO-GII Board Showing Scope Probe Connections for Power Attacks

Similar to Figure 4.3 in section 4.2.1, the results for the on-chip power attack are presented in Figure 4.5 below. Once again the graph shows that the on-chip power attack has failed, since the minimum rank attained is 109 (using 29184 traces). The graph also shows that the rank of the correct guess does not improve significantly with increasing the number of traces.

This experiment does yield better results than the on-chip EM experiment. The correlation coefficient of the correct guess is approximately 0.70, while the maximum correlation coefficient is calculated to be over 0.80. Those results indicate that the side channel through the power supply rail is better than that through the electromagnetic emanation of the FPGA chip; however a large gap exists between the maximum correlation coefficient and the coefficient of the correct guess, leading to a high rank for the correct guess.

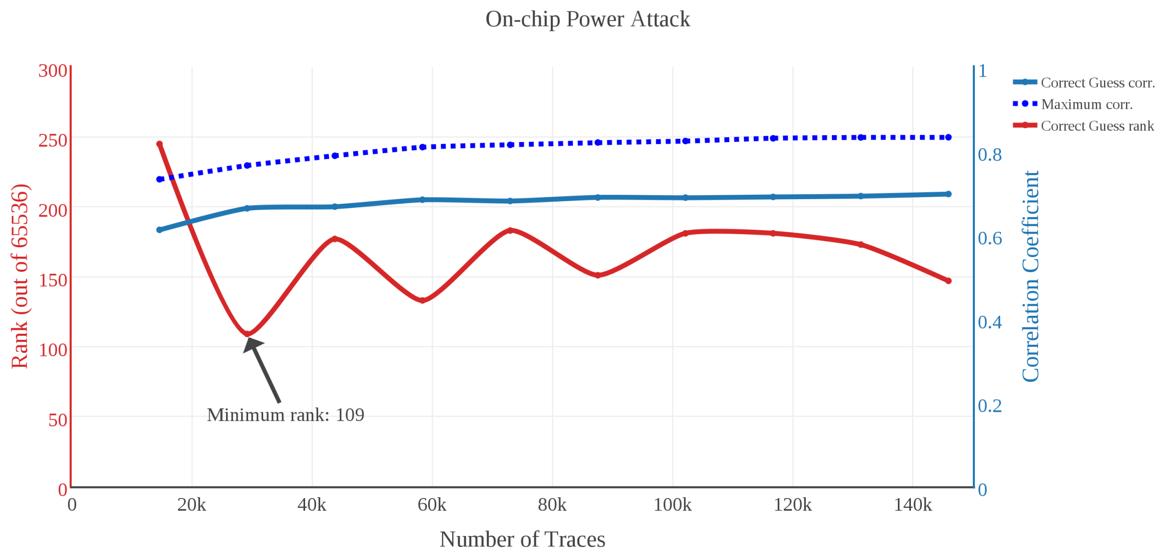


Figure 4.5: Results for On-chip Power Attack

4.2.3 Off-chip EM attack

Since both on-chip attacks were not successful, two subsequent off-chip attacks were carried out. As stated previously, the only difference between the two types of attacks is that in the off-chip attacks the values of the a and e registers were sent off-chip via I/O pins. The rationale behind that is sending signals off-chip consumes some additional power. Similar to saving the values on internal registers, the additional power consumed should be dependent on the hamming distance of the data values. Thus, by sending the signal off-chip, it is expected that the side channel becomes stronger resulting in a higher probability for the attack to be successful.

The first off-chip attack carried out was the EM attack. In this attack, the same configuration as the on-chip EM attack was used, with a minor modification to the hardware design of the cryptographic core which assigns the values of the targeted registers to I/O pins at the attack point. The oscilloscope probes were connected as shown in Figure 4.1.

The results of this attack are summarized in Figure 4.6 below. The minimum rank attained in this attack is 43 (using only 14592 traces). Interestingly, the graph shows that as the number of traces increases, the rank of the correct guess deteriorates. This characteristic is counter intuitive and was not observed in the previous two attacks. Further, the correlation coefficient of the correct guess seems to be stable at approximately 0.7, while the maximum correlation coefficient calculated improves slightly as the number of traces increases. The increased gap between the correlation coefficient of the correct guess and the maximum correlation coefficient explains why the rank deteriorates with increasing the number of traces used in the analysis.

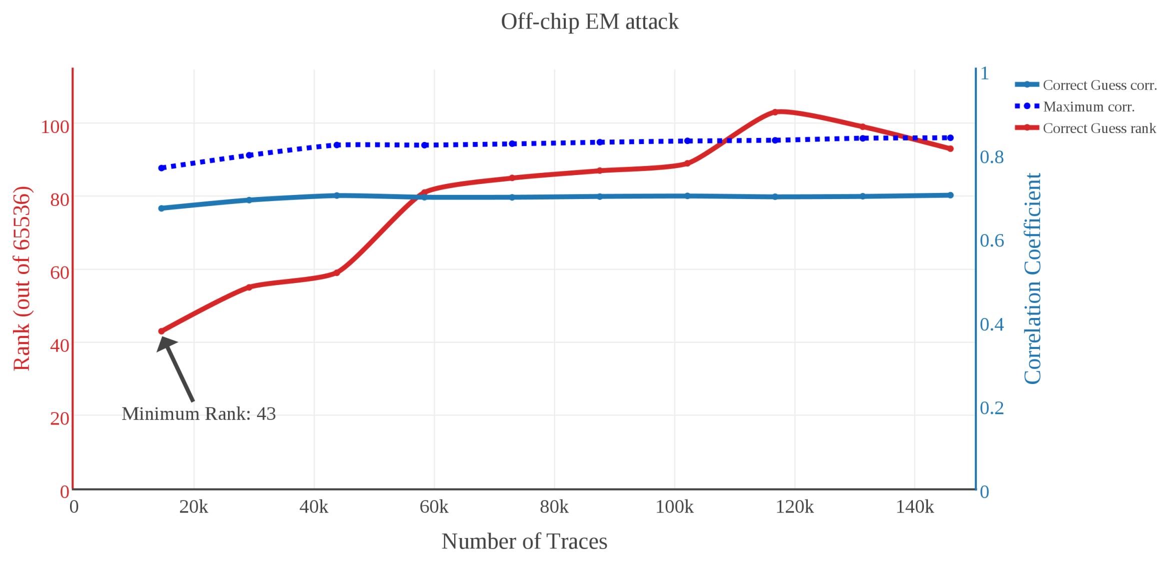


Figure 4.6: Results for Off-chip EM Attack

4.2.4 Off-chip power attack

The previous attack proved that sending the register values off-chip leads to a better side channel; however it failed due to the large gap between maximum correlation coefficient and that of the correct guess.

The final attack carried out was the off-chip power attack. It uses the same configuration as that of the on-chip power attack, including how the oscilloscope probes are connected to the board. The results of this attack are shown in Figure 4.7 below.

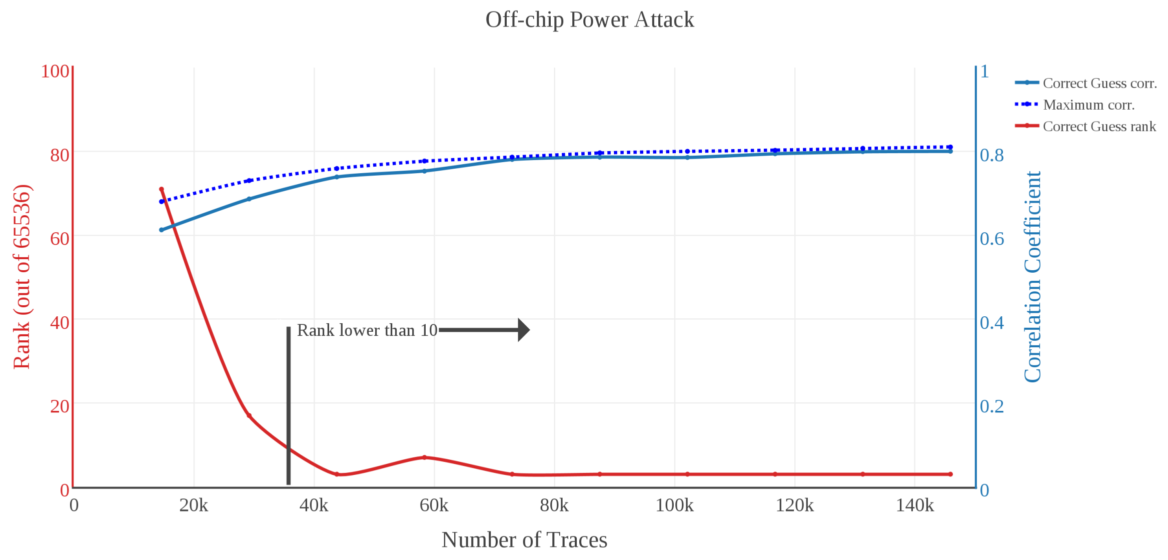


Figure 4.7: Results for Off-chip power attack

This attack was a success. As shown in the figure above, the rank of the correct guess dropped below the threshold of 10 with approximately 35000 traces. The best rank achieved was 3 (with 43776 traces), and beyond that increasing the number of traces analyzes does not seem to improve the rank.

The correlation coefficient of the correct guess starts off slightly low (approximately 0.6 with 14592 traces); however it quickly rises and approaches the maximum coefficient (approximately 0.8 with 87552 traces). The four highest correlation coefficients found in the experiment are presented in Table 4.2 below. Notice the guesses that ranked first and second are opposites of each other (the values of the A byte guess and E byte guess are swapped between those two guesses).

Table 4.2: Results with the Highest Correlation Coefficients

Rank	**Correlation Coefficient**	**E Byte Guess**	**A Byte Guess**
1	0.80259	$E6_{16}(11100110)_2$	$4C_{16}(01001100)_2$
2	0.80259	$4C_{16}(01001100)_2$	$E6_{16}(11100110)_2$
3	0.79437	$E7_{16}(11100111)_2$	$4D_{16}(01001101)_2$
4	0.79437	$4D_{16}(01001101)_2$	$E7_{16}(11100111)_2$

4.3 Results Comparison

In this section, a comparison of the results obtained from the four experiments will be presented. The correlation coefficients and the ranks from the different experiments will be compared separately. The correlation coefficients give an indication about the strength of the side channel targeting in a particular experiment, while the rank is used as a measure to assess the success of the experiment.

Figure 4.8 below shows a comparison of the correlation coefficients obtained in the four experiments; for each experiment the coefficient of the correct guess is plotted in a solid line, and the maximum coefficient obtained is plotted in a dotted line. The four experiments may be distinguished by the color of the plotted lines.

The graph shows that the on-chip EM experiment yielded the worst results, indicating a poor side channel. The other three experiments all had a better side channel (as indicated by the maximum coefficient lines plotted in dotted lines); however a large gap exists between the maximum coefficients and the coefficients of the correct guess for the on-chip power and off-chip EM experiments. This indicates that while a strong side channel may exist, it is difficult to distinguish the correct guess from the others.

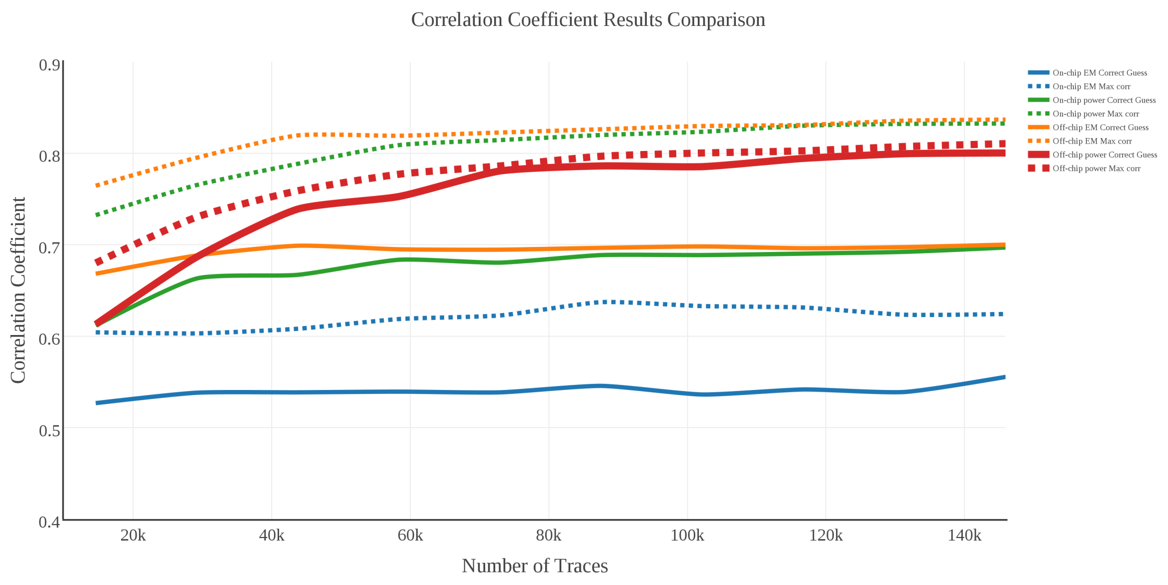


Figure 4.8: Comparison of Correlation Coefficient Results of the Four Experiments

Figure 4.9 below shows a comparison of the ranks obtained in the off-chip EM, on-chip power and off-chip power attacks. The ranks from the off-chip EM attack are omitted from this graph since they are an order of magnitude higher than the remaining ranks.

The graph indicates that the off-chip power attack is the only one that yielded a rank below the success threshold of 10. Further, the rank obtained in the other two experiments shown on the graph do not improve with increasing the number of traces analyzed up to 145920 traces.

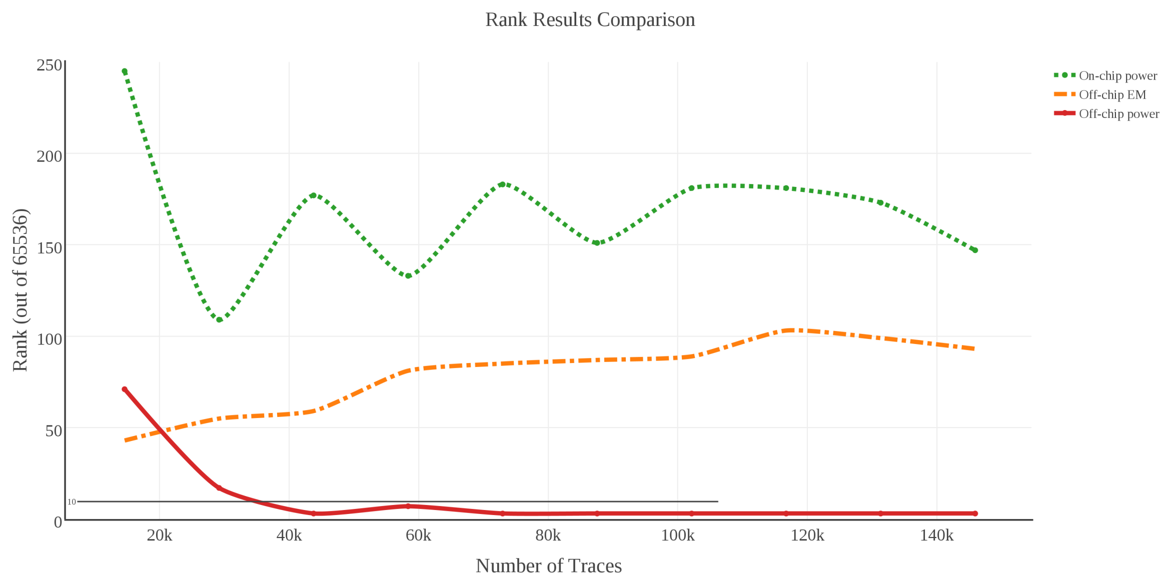


Figure 4.9: Comparison of Rank Results of the Four Experiments

4.4 Comparison to Previous Work

In recent years power analysis attacks against cryptographic devices have become fairly common and much research has been devoted to understanding the effectiveness of different types of attacks against various cryptosystems. In discussing the related work in chapter two, references were made to successful side channel attacks using Simple Power Analysis (such as attacking the DES key schedule in [23]), Differential Power Attacks (such as the DPA attack on HMAC-SHA256 in [26]) and Correlation Power Attacks (such as the CPA attack on HMAC-SHA256 in [11]).

The basis of this thesis work is the CPA attack on HMAC-SHA256 presented in [11]. That attack was executed on the compressor part of the single-cycle round implementation of SHA256 on the Altera DE2 board which uses the Altera Cyclone II FPGA. The Cyclone II FPGA uses a 90-nm process technology and is powered by a 1.2 V supply [12]. On that architecture the CPA attack was successful using 1024 EM traces and yielded a correlation coefficient of 0.95 for the correct byte guess.

In this thesis work, the same CPA attack was executed on the complete single-cycle round implementation of SHA256 on the SASEBO-GII board which uses the Xilinx Virtex-5 FPGA. The FPGA attacked in this experiment uses a 65-nm technology and is powered by a 1 V supply. The attack on this FPGA is theoretically more difficult since it is a lower power chip. Four different attacks were attempted on this FPGA, two EM and two power attacks. The only successful attack in this thesis work, off-chip power attack, required more than 43776 power traces, and the highest correlation coefficient achieved was 0.80.

Chapter 5

Conclusion

In this thesis work the security of the HMAC-SHA256 single-cycle round architecture implemented on the Xilinx Virtex-5 FPGA platform was analyzed. In particular, the susceptibility of the platform to various correlation power analysis attacks was evaluated.

The SASEBO-GII board was the chosen target board for the experiments. The board hosts the low power Xilinx Virtex-5 FPGA as the cryptographic FPGA, and a second Xilinx Spartan-3 FPGA as the control FPGA. A hardware design of the single-cycle round architecture of SHA256, which executes each round of the algorithm in a single clock cycle, was developed for the Virtex-5 FPGA. In addition, a board interface application was designed to execute on the host PC to provide two way communication between the PC and the cryptographic FPGA. The application was used to send plaintext messages to the hardware SHA256 core, and receive ciphertext data back from the core.

The high sample rate Tektronix TDS7254 oscilloscope was used to capture traces from the board. Some built in features on the oscilloscope were utilized to get around some of the setup challenges faced in the experiment. The *FastFrame* feature was used to collect multi-trace acquisitions, which greatly reduced the time required to capture traces. Further, the *Waveform Data Range* feature was used to crop the captured traces to the desired attack point. The Rohde &Schwarz HZ-15 EM probe was used to collect EM traces; and a SMA to BNC passive voltage probe was used to collect power consumption traces.

As discussed in chapter 4 of this thesis, four separate attacks were attempted on the target platform: On-chip EM, on-chip power, off-chip EM and off-chip power. In this naming scheme, on-chip/off-chip denotes whether the targeted register (a and e working variables in the SHA256 algorithm) are stored in registers on the FPGA chip only, or if they were sent off-chip on multipurpose IO pins. The on-chip attacks were carried out first, but they both failed to predict the correct byte guess, therefore the off-chip attacks were performed subsequently. Power/EM denotes whether the captured traces measured the power consumption of the chip or the electromagnetic emanation off the chip.

Of the four attacks performed, only the off-chip power attack was successful at ranking the correct bytes guess among the top 10 highest correlation coefficients. In that attack, the correct guess was ranked third (at a correlation coefficient of 0.79) using 43776 traces. Increasing the number of traces analyzed beyond that number did not improve the rank of the correct guess. Despite the correct guess not ranking first in the attack, it is computationally trivial to perform a trial and error process on the top 10 highest ranking guesses to determine the correct guess among them.

In performing the four attacks in this thesis work, an interesting observation was made. Both EM attacks (on-chip and off-chip) yielded poor results. This indicates that the electromagnetic emanation side channel of Virtex-5 XC5VLX30-1FFG324 is fairly poor (it does not leak information about the operation of the chip). This observation may be attributed to the fact that the FPGA chip has 12 metal layers around the silicon die to optimize routing [18]. EM interference between those metal layers may be greatly degrading the signals. Furthermore, the cryptographic FPGA chip on the SASEBO-GII chip has a metal shield covering it, as can be seen in Figure 5.1 below. The metal shielding likely contributes the most to the EM signal degradation.

Furthermore, the fact that the on-chip power attack yielded poor results indicates that the Virtex-5 FPGA chip has a poor power side channel in the case where signals remain on chip. This is likely attributed to the fact that it is a low power FPGA chip. On the other hand, the attack was successful for the off-chip case (using only power measurements of the core), since sending signals off chip draws additional power from the VCC_{INT} rail. The additional power drawn was not directly measured in this experiment.

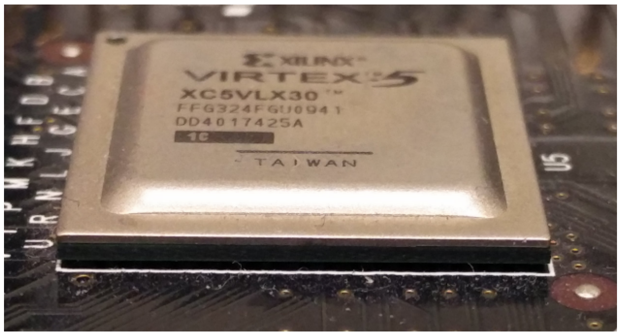


Figure 5.1: A Close Up of the Virtex-5 Cryptographic FPGA showing the Metal Shielding

5.1 Future Work

The thesis work has shown that the HMAC-SHA256 single-cycle round architecture algorithm running on the low power Virtex-5 XC5VLX30-1FFG324 FPGA is susceptible to the correlation power analysis attack, only if the state of the internal working registers of the SHA256 algorithm are sent off-chip. In the future, it would be interesting to investigate whether this cryptosystem implemented on this hardware platform may be susceptible to stronger attacks (such as the EM template attack described in [8]).

Furthermore, in the future the SASEBO-GII platform will be used to assess the security of other cryptosystems. The groundwork done in this thesis work to modify the hardware design published with the DPA Contest to transform the SASEBO-GII platform from AES-128 to SHA256, as detailed in section 3.3, will serve as a reference to future researchers tackling this problem.

APPENDICES

Appendix A

SASEBO board Cryptographic FPGA Power Supply Schematic

The figure below is extracted from the SASEBO board schematic provided in the *Side-channel Attack Standard Evaluation Board SASEBO-GII Specification (Version 1.01)* [28]. Four components are circled in the schematic, they are:

1. U1 Voltage regulator: This IC is a step down voltage regulator. It converts the 3.3V input to a 1.0V output (to be used by the cryptographic core)
2. SW1 Switch: This switch is used to select the source of the 1.0V power rail to be supplied to the cryptographic core. In it's current configuration, the output of the voltage regulator U1 is chosen to supply the cryptographic FPGA.
3. R1 shunt resistor: This is the shunt resistor used to allow for power consumption measurements on the supply rail.
4. TP2 test point: Oscilloscope probe is connected here to record power measurements.

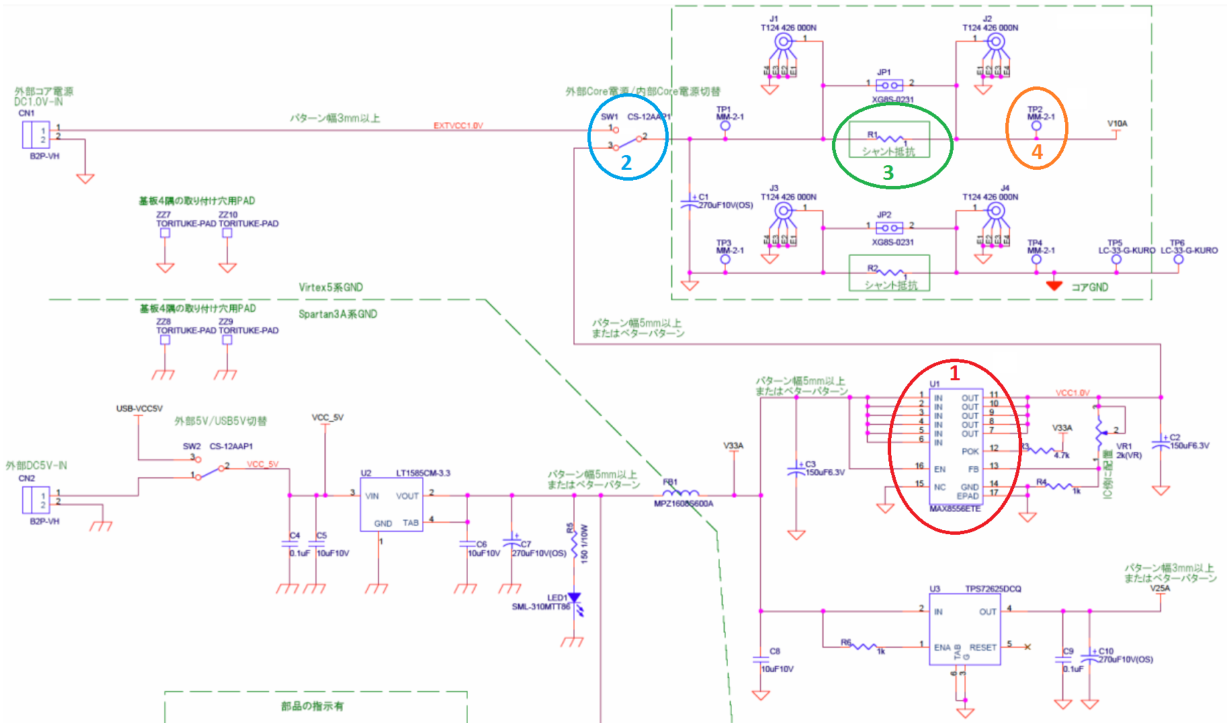


Figure A.1: SASEBO Board Schematic Extract Showing the Power Supply Network to the Cryptographic FPGA

Appendix B

Hardware Design of SHA256 Module

This appendix contains the code listings for the SHA256 hardware module. The version listed below contains the code modification to output the internal values of the attacked registers onto I/O pins on the board.

```
module SHA_256(Kin, Din, Dout_SHA, Krdy, Drdy, EncDec, RSTn, EN, CLK, BSY,
   Kvld, Dvld, Ftrig, RegIO);
  input   [127:0] Kin;   // Key input
  input   [127:0] Din;   // Data input
  output  [127:0] Dout_SHA; // Data output
  input   Krdy;          // Key input ready
  input   Drdy;          // Data input ready
  input   EncDec;        // 0:Encryption 1:Decryption
  input   RSTn;          // Reset (Low active)
  input   EN;            // AES circuit enable
  input   CLK;           // System clock
  output  BSY;           // Busy signal
  output  Kvld;          // Data output valid
  output  Dvld;          // Data output valid
  output  Ftrig;     // Flexiable trigger output
  output  [15:0] RegIO; // Register output to IO pins

  wire [127:0] Dout_SHA;
  wire BSY_E, BSY_D;
```

```verilog
  wire Dvld_E, Dvld_D;
  wire Kvld_E, Kvld_D;

  wire Dvld_tmp, Kvld_tmp;
  reg  Dvld_reg, Kvld_reg;

  assign Dvld_tmp = Dvld_E;
  assign Kvld_tmp = Kvld_E;

  assign Dvld = ( (Dvld_reg == 1'b0) && (Dvld_tmp == 1'b1) ) ? 1'b1: 1'b0;
  assign Kvld = ( (Kvld_reg == 1'b0) && (Kvld_tmp == 1'b1) ) ? 1'b1: 1'b0;

  SHA256_main SHA256_main(Kin, Din, Dout_SHA, Krdy, Drdy, RSTn, EN, CLK, BSY
    , Kvld_E, Dvld_E, Ftrig, RegIO);

 // Behavior for Dvld_reg and Kvld_reg.
  always @(posedge CLK) begin
    if (RSTn == 0) begin
      Dvld_reg <= 1'b0;
      Kvld_reg <= 1'b0;
    end
    else if (EN == 1) begin
      Dvld_reg <= Dvld_tmp;
      Kvld_reg <= Kvld_tmp;
    end
  end
endmodule

////////////////////////////
//      SHA Core      //
////////////////////////////
module SHA256Core(a,b,c,d,e,f,g,h,New_a,New_b,New_c,New_d,New_e,New_f,New_g,
    New_h,W,RoundCountN, CLK, selectKin,check_it);
  input  [31:0] a,b,c,d,e,f,g,h,W;
  input  [9:0]   RoundCountN;
  input  CLK;

  output [31:0] New_a,New_b,New_c,New_d,New_e,New_f,New_g,New_h,check_it;
```

```
input   [3:0]   selectKin;
wire    [31:0]  T1, T2;
wire    [31:0]  temp_BS0, temp_BS1, temp_Ch, temp_Maj,const_Kout ;

Ch      CH_INST    (e,f,g,temp_Ch);
Maj       MAJ_INST   (a,b,c,temp_Maj);
SIGMA0   SIGMA0_INST(a,temp_BS0);
SIGMA1   SIGMA1_INST(e,temp_BS1);
const_k   K_inst(RoundCountN,const_Kout);

assign New_h=g;
assign New_g=f;
assign New_f=e;
assign New_e=d+h + temp_BS1 + temp_Ch + const_Kout + W;
assign New_d=c;
assign New_c=b;
assign New_b=a;
assign New_a=h + temp_BS1 + temp_Ch + const_Kout + W+temp_BS0 + temp_Maj;

  //Output multiplexer. Controlled by byte 14 of KEY field in interface
 application
   function [31:0] MUX_a;
   input [3:0] selectKin;
       case (selectKin)
         4'b0000 : MUX_a = h;
         4'b0001 : MUX_a = temp_BS1;
         4'b0010 : MUX_a = temp_Ch;
         4'b0011 : MUX_a = const_Kout;
         4'b0100 : MUX_a = W;
         4'b0101 : MUX_a = temp_BS0;
         4'b0110 : MUX_a = temp_Maj;
         4'b0111 : MUX_a = {22'b0,RoundCountN};
         4'b1000 : MUX_a = a;
         4'b1001 : MUX_a = b;
         4'b1010 : MUX_a = c;
         4'b1011 : MUX_a = d;
         4'b1100 : MUX_a = e;
         4'b1101 : MUX_a = f;
```

```
          4'b1110  : MUX_a = g;
          default: MUX_a = h + temp_BS1 + temp_Ch + const_Kout + W+temp_BS0
  + temp_Maj;
        endcase
      endfunction
      assign check_it=MUX_a(selectKin);

endmodule
////////////////////////////
//   SHA 256 main          //
////////////////////////////
module SHA256_main(Kin, Din, Dout, Krdy, Drdy, RSTn, EN, CLK, BSY, Kvld,
   Dvld, Ftrig, RegIO);
  input   [127:0] Kin;   // Key input
  input   [127:0] Din;   // Data input
  output  [127:0] Dout;  // Data output
  input   Krdy;          // Key input ready
  input   Drdy;          // Data input ready
  input   RSTn;          // Reset (Low active)
  input   EN;            // AES circuit enable
  input   CLK;           // System clock
  output  BSY;           // Busy signal
  output  Kvld;          // Key valid
  output  Dvld;          // Data output valid
  output  Ftrig;       // Flex Trig output
  output  [15:0] RegIO;  //A and E register to IO pins

  reg  [15:0] IOrg;
  reg  TRIGrg;
  reg   [9:0]    RoundCount;     // Round counter
  reg   Kvldrg, Dvldrg, BSYrg;
  reg  [127:0] Dout;
  wire    [31:0] temp_s0, temp_s1;
  wire   [3:0] selectKin;

// Original values from the standar
//  reg    [31:0]  H0=32'h6a09e667;
//  reg    [31:0]  H1=32'hbb67ae85;
//   reg    [31:0]   H2=32'h3c6ef372;
//   reg    [31:0]   H3=32'ha54ff53a;
//   reg    [31:0]   H4=32'h510e527f;
//   reg    [31:0]   H5=32'h9b05688c;
//   reg    [31:0]   H6=32'h1f83d9ab;
//   reg    [31:0]   H7=32'h5be0cd19;

//Values after initial key round
  reg    [31:0]   H0=32'h27DAB238;
  reg    [31:0]   H1=32'h1B9C94CA;
  reg    [31:0]   H2=32'h754D32CB;
  reg    [31:0]   H3=32'hC525442C;
  reg    [31:0]   H4=32'hD704401D;
  reg    [31:0]   H5=32'hB623BA92;
  reg    [31:0]   H6=32'h3A50963B;
  reg    [31:0]   H7=32'h2AF82D54;

  reg    [31:0]   a,b,c,d,e,f,g,h;
  wire  [31:0]   New_a,New_b,New_c,New_d,New_e,New_f,New_g,New_h,check_it;
  reg     [31:0] W[0:64];
  assign selectKin = Kin[19:16];
  SHA256Core SHAC (a,b,c,d,e,f,g,h,New_a,New_b,New_c,New_d,New_e,New_f,New_g
    ,New_h,W[RoundCount],(RoundCount), CLK,selectKin ,check_it);

  assign Kvld = Kvldrg;
  assign Dvld = Dvldrg;
  assign BSY  = BSYrg;
  assign Ftrig = TRIGrg;
  assign RegIO = IOrg;

  small_sigma1 SS1 (W[RoundCount-2],temp_s1);
  small_sigma0 SS0 (W[RoundCount-15],temp_s0);

  always @(posedge CLK) begin
  TRIGrg <= 0;
  if (RSTn == 0) begin
    RoundCount     <= 10'b0000000000;
    Kvldrg <= 0;
    Dvldrg <= 0;
    BSYrg  <= 0;
  end
  else if (EN == 1) begin
   if (BSYrg == 0) begin
     if (Krdy == 1) begin
        Kvldrg <= 1;
        Dvldrg <= 0;
     end
     else if (Drdy == 1) begin
       TRIGrg = 1;
       a <= H0;
       b <= H1;
       c <= H2;
       d <= H3;
       e <= H4;
       f <= H5;
       g <= H6;
       h <= H7;
       RoundCount <= 10'b0000000000;
       Dvldrg <= 0;
       BSYrg  <= 1;
     end
   end
   else begin
     a<=New_a;
     b<=New_b;
     c<=New_c;
     d<=New_d;
     e<=New_e;
     f<=New_f;
     g<=New_g;
     h<=New_h;
      if (RoundCount == Kin[7:0]) begin
        Dvldrg <= 1;
        BSYrg  <= 0;
      end
      else begin
```

```
          if (RoundCount==0) begin
              W[0]=Din[127:96];
          end
          else if (RoundCount==1) begin
              W[1]=Din[95:64];
          end
          else if (RoundCount==2) begin
              W[2]=Din[63:32];
          end
          else if (RoundCount==3) begin
              W[3]=Din[31:0];
          end
          else if (RoundCount<14) begin
            W[RoundCount]=32'h0;
          end
          else if (RoundCount==15) begin
            W[RoundCount]=32'h00000018;
          end
          else begin
            W[RoundCount]=temp_s1+W[RoundCount-7]+temp_s0+W[RoundCount-16];
          end
          Dout[127:32]<= {check_it,a+H0,W[RoundCount]};
          Dout[31:10]<= 21'b0;
          Dout[9:0]<=RoundCount;
          RoundCount <= RoundCount+1;
          IOrg <= {a[31:24], e[31:24]}; //[a7 a6 ..a0 e7 e6 ..e0]
        end
      end
    end
  end
endmodule

module const_k(k_input,k_output);
  input [9:0] k_input;
  output [31:0] k_output;
  reg[31:0] regk[0:63];
  initial begin
      regk[0]=32'h428a2f98;
```

```
        regk[1]=32'h71374491;
        regk[2]=32'hb5c0fbcf;
        regk[3]=32'he9b5dba5;
        regk[4]=32'h3956c25b;
        regk[5]=32'h59f111f1;
        regk[6]=32'h923f82a4;
        regk[7]=32'hab1c5ed5;
        regk[8]=32'hd807aa98;
        regk[9]=32'h12835b01;
        regk[10]=32'h243185be;
        regk[11]=32'h550c7dc3;
        regk[12]=32'h72be5d74;
        regk[13]=32'h80deb1fe;
        regk[14]=32'h9bdc06a7;
        regk[15]=32'hc19bf174;
        regk[16]=32'he49b69c1;
        regk[17]=32'hefbe4786;
        regk[18]=32'h0fc19dc6;
        regk[19]=32'h240ca1cc;
        regk[20]=32'h2de92c6f;
        regk[21]=32'h4a7484aa;
        regk[22]=32'h5cb0a9dc;
        regk[23]=32'h76f988da;
        regk[24]=32'h983e5152;
        regk[25]=32'ha831c66d;
        regk[26]=32'hb00327c8;
        regk[27]=32'hbf597fc7;
        regk[28]=32'hc6e00bf3;
        regk[29]=32'hd5a79147;
        regk[30]=32'h06ca6351;
        regk[31]=32'h14292967;
        regk[32]=32'h27b70a85;
        regk[33]=32'h2e1b2138;
        regk[34]=32'h4d2c6dfc;
        regk[35]=32'h53380d13;
        regk[36]=32'h650a7354;
        regk[37]=32'h766a0abb;
        regk[38]=32'h81c2c92e;
        regk[39]=32'h92722c85;
        regk[40]=32'ha2bfe8a1;
        regk[41]=32'ha81a664b;
        regk[42]=32'hc24b8b70;
        regk[43]=32'hc76c51a3;
        regk[44]=32'hd192e819;
        regk[45]=32'hd6990624;
        regk[46]=32'hf40e3585;
        regk[47]=32'h106aa070;
        regk[48]=32'h19a4c116;
        regk[49]=32'h1e376c08;
        regk[50]=32'h2748774c;
        regk[51]=32'h34b0bcb5;
        regk[52]=32'h391c0cb3;
        regk[53]=32'h4ed8aa4a;
        regk[54]=32'h5b9cca4f;
        regk[55]=32'h682e6ff3;
        regk[56]=32'h748f82ee;
        regk[57]=32'h78a5636f;
        regk[58]=32'h84c87814;
        regk[59]=32'h8cc70208;
        regk[60]=32'h90befffa;
        regk[61]=32'ha4506ceb;
        regk[62]=32'hbef9a3f7;
        regk[63]=32'hc67178f2;
   end
   assign  k_output=regk[k_input];
endmodule

module Maj(x, y, z, Maj_out );
   input   [31:0] x;
   input    [31:0] y;
   input    [31:0] z;
   output  [31:0] Maj_out;

   assign Maj_out = ((x & y )^ (x & z ) )^(y & z);
endmodule

```

```
module Ch(x, y, z, Ch_out);
  input   [31:0] x;
  input    [31:0] y;
  input    [31:0] z;
  output [31:0] Ch_out;

  assign Ch_out = (x & y )^ (~x & z );
endmodule

module SIGMA0(x , SIGMA0_out);
  input   [31:0] x;
  output [31:0] SIGMA0_out;

  assign SIGMA0_out = {x[1:0],x[31:2]}^ {x[12:0],x[31:13]} ^{x[21:0],x
    [31:22]};
endmodule

module SIGMA1(x , SIGMA1_out);
  input   [31:0] x;
  output [31:0] SIGMA1_out;

  assign SIGMA1_out = {x[5:0],x[31:6]}^ {x[10:0],x[31:11]} ^{x[24:0],x
    [31:25]};
endmodule

module small_sigma0(x , small_sigma0_out);
  input   [31:0] x;
  output [31:0] small_sigma0_out;

  assign small_sigma0_out[31:29] = x[6:4]^ x[17:15];
  assign small_sigma0_out[28:0] = {x[3:0],x[31:7]}^ {x[14:0],x[31:18]} ^x
    [31:3];
endmodule

module small_sigma1(x, small_sigma1_out);
  input   [31:0] x;
  output [31:0] small_sigma1_out;
  assign small_sigma1_out[31:22] = x[16:7]^ x[18:9];
```

```verilog
  assign small_sigma1_out[21:0] = {x[6:0],x[31:17]}^ {x[8:0],x[31:19]} ^x
    [31:10];
endmodule
```

References

[1] Dpa contest v2 introduction. http://www.dpacontest.org/v2/, 2010. [Acccessed: 2015-7-13].

[2] Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Comparative evaluation of rank correlation based dpa on an aes prototype chip. In *International Conference on Information Security*, pages 341–354. Springer, 2008.

[3] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Annual International Cryptology Conference*, pages 1–15. Springer, 1996.

[4] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[5] Vincent Carlier, Hervé Chabanne, Emmanuelle Dottax, and Hervé Pelletier. Electromagnetic side channels of an fpga implementation of aes. In *CRYPTOLOGY EPRINT ARCHIVE, REPORT 2004/145*. Citeseer, 2004.

[6] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 252–263. Springer, 2000.

[7] Jean-Sébasticn Coron, Paul Kocher, and David Naccache. Statistics and secret leakage. In *International Conference on Financial Cryptography*, pages 157–173. Springer, 2000.

[8] Pierre-Alain Fouque, Gaëtan Leurent, Denis Réal, and Frédéric Valette. Practical electromagnetic template attack on hmac. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 66–80. Springer, 2009.

[9] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 251–261. Springer, 2001.

[10] Catherine H Gebotys. *Security in embedded devices.* Springer Science & Business Media, 2009.

[11] Catherine H Gebotys, Brian A White, and Edgar Mateos. Preaveraging and carry propagate approaches to side-channel analysis of hmac-sha256. *ACM Transactions on Embedded Computing Systems (TECS)*, 15(1):4, 2016.

[12] Altera Inc. Cyclone ii device handbook, volume 1. https://www.altera.com/en_US/pdfs/literature/hb/cyc2/cyc2_cii5v1.pdf, 2008. [Acccessed: 2016-2-7].

[13] Koa Speer Electronics Inc. General purpose 2%, 5% tolerance thick film chip resistor. http://pdf.datasheet.cloud/datasheets-1/koa_speer_electronics/RK73BW2HTTD1R0J.pdf, 2011. [Acccessed: 2015-12-11].

[14] Tektronix Inc. Tds7404, tds7254 & tds7154 user manual, 2002.

[15] Tektronix Inc. P6139b p5050b 500 mhz passive probe, datasheet, 2016.

[16] Xilinx Inc. Spartan-3a fpga family: Data sheet (v2.0). http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf, 2010. [Acccessed: 2016-2-6].

[17] Xilinx Inc. Platform cable usb ii. http://www.xilinx.com/support/documentation/data_sheets/ds593.pdf, 2015. [Accessed: 2015-10-8].

[18] Xilinx Inc. Virtex-5 family overview (v5.1). http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2015. [Acccessed: 2015-8-17].

[19] Marcio Juliato and Catherine Gebotys. Fpga implementation of an hmac processor based on the sha-2 family of hash functions. *University of Waterloo, Tech. Rep*, 2011.

[20] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2, 3):141–158, 2000.

[21] Paul Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology Crypto '96, Lecture Notes in Computer Science, vol 1109*, pages 104–113. Springer, 1996.

[22] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Introduction to differential power analysis and related attacks. https://42xtjqm0qj0382ac91ye9exr-wpengine.netdna-ssl.com/wp-content/uploads/2015/08/DPATechInfo.pdf, 1998. [Accessed: 2015-8-5].

[23] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science, vol 1666*, pages 388–397. Springer, 1999.

[24] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1:5–27, 2011.

[25] Roman Korkikian. Power analysis basics. https://cybermashup.com/2014/07/16/power-analysis-basics/, 2014. [Accessed: 2016-3-3].

[26] Robert McEvoy, Michael Tunstall, Colin C Murphy, and William P Marnane. Differential power analysis of hmac based on sha-2, and countermeasures. In *International Workshop on Information Security Applications*, pages 317–332. Springer, 2007.

[27] Thomas S Messerges, Ezzy A Dabbish, and Robert H Sloan. Investigations of power analysis attacks on smartcards. *Smartcard*, 99:151–161, 1999.

[28] National Institute of Advanced Industrial Science and Technology. Side-channel attack standard evaluation board sasebo-gii specification (version 1.01). http://www.rcis.aist.go.jp/files/special/SASEBO/SASEBO-GII-en/SASEBO-GII_Spec_Ver1.01_English.pdf, 2009. [Accessed: 2015-8-10].

[29] Katsuyuki Okeya. Side channel attacks against hmacs based on block-cipher based hash functions. In *Australasian Conference on Information Security and Privacy*, pages 432–443. Springer, 2006.

[30] FIPS Pub. 198, the keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, 2002.

[31] FIPS Pub. 180-3, secure hash standard (sha). *Federal Information Processing Standards Publications*, 2008.

[32] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.

[33] Rohde Schwarz. Probe set r& s for e and h near-field emission measurements with test receivers and spectrum analyzers. https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/HZ-15_en.pdf, 2006. [Acccessed: 2015-7-13].

[34] Rohde Schwarz. R& s hz-16 preamplifier 3ghz, 20 db, power adapter (100v to 230v), for r& s hz-15. https://www.rohde-schwarz.com/us/product/hz16-productstartpage_63493-8986.html, 2006. [Acccessed: 2015-7-13].

[35] Kenneth C Sedra, Adel Smith. *Microelectronic circuits*. Oxford University Press, 2010.

[36] Neil HE Weste and Kamran Eshraghian. *Principles of CMOS VLSI design*, volume 188. Addison-Wesley New York, 1985.