Fall 2012

Analysis of DPA and DEMA Attacks

Cheuk Wong
San Jose State University

Analysis of DPA and DEMA Attacks

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Cheuk Wong

May 2012

The Designated Project Committee Approves the Project Titled

Analysis of DPA and DEMA Attacks

by

Cheuk Wong

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2012

Dr. Mark Stamp	Department of Computer Science
Dr. Robert Chun	Department of Computer Science
Jasper van Woudenberg, MSc	Riscure North America

ABSTRACT

Analysis of DPA and DEMA Attacks

by Cheuk Wong

Side channel attacks (SCA) are attacks on the implementations of cryptographic algorithms or cryptography devices that do not employ full brute force attack or exploit the weaknesses of the algorithms themselves. There are mant types of side channel attacks, and they include timing, sound, power consumptions, electromagnetic (EM) radiations, and more. A statistical side channel attack technique that uses power consumption and EM readings was developed, and they are called Differential Power Analysis (DPA) and Differential Electromagnetic Analysis respectively.

DPA takes the overall power consumption readings from the system of interest, and DEMA takes a localized EM readings from the system of interest. In this project, we will examine the effectiveness of both techniques and compare the results. We will compare the techniques based on the amount of resource and time they needed to perform a successful SCA on the same system. In addition, we will attempt to use a radio receiver to down mix the power consumption readings and the EM readings to reduce the amount of computing resources it takes to perform SCA. We will provide our test results of performing SCA with DPA and DEMA, and we will also compare the results to determine the effectiveness of the two techniques.

ACKNOWLEDGMENTS

I would like to thank Jasper van Woudenberg for giving me the chance to conduct the experiments. In addition, I would like to thank the folks at Riscure for letting me use their equipments for this project as well as helping me in completing this project.

Furthermore, I would like to thank Dr. Mark Stamp and Dr. Robert Chun at SJSU for giving me valuable insight and suggestions for this project. Lastly, I would like to thank my family for all the supports while I was in my master program.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Cryptography

In cryptography, hardware implementations of cryptographic algorithms are used to speed up the encryption and decryption processes [7, 9, 17]. Cryptographic algorithms commonly used in everyday practices are DES, 3DES, AES, and RSA. While these algorithms are mathematical secured, the hardware implementations can sometime unintentionally leak informations regarding the implementation of the algorithms, the data being encrypted, or even the secret keys. The process of attacking the algorithms from these leaked information is called side channel attack, and two of the most commonly used side channel attacks are by obtaining the power consumption readings and electromagnetic field changes during cryptographic operations on the hardware implementations [1, 2, 3]. This paper will compare the efficiencies of using the power consumption model and electromagnetic field model as side channel attacks on embedded systems. The motivation of this project is to determine the best method of performing side channel attack on embedded systems. If we can determine the best method of performing side channel attack on embedded system, then we can begin the attack with such method and not spend time trying to figure out which method is viable and which is not.

1.1 Cryptographic Algorithms

Cryptography is the practice of communicating in secrets. The basic idea of cryptography is to allow two parties to securely communicate, so no other party can read the messages between the two parties [10]. A message is referred as plain text, and an encryption algorithm transforms a plain text into cipher text. A cipher text is

unreadable unless the decryption algorithm is applied. A good encryption algorithm will produce a seemingly random cipher text by employing confusion and diffusion techniques. Confusion is to obscure the relationship between the plain text and the cipher text, and diffusion is to spread the statistical information of the plain text all around the cipher text.

A cryptographic algorithm can be thought of as a pair of mathematical functions. The first function is the encryption algorithm, and it takes two arguments, the plain text and a secret key, and produces the cipher text. The second function is the decryption function, and it takes the cipher text and the secret key as arguments and produces the plain text. If the secret key matches, the encryption and decryption functions can be thought of as inverse function of each others. According to Kerckhoff's principle, the strength of a cryptographic algorithm should only be dependent on the length of the secret key, and it is assumed that the cryptographic algorithm is known by everyone [10].

There are two types of cryptographic algorithms, and they are symmetric key cipher and asymmetric keys cipher. In a symmetric key cipher, both parties share the same secret key for encryption and decryption. Some of the most popular symmetric key ciphers are DES, 3DES, and AES. On the other hands, an asymmetric key cipher, or often referred as public key encryption, uses a key for encryption and a different key for decryption. The key for encryption is often referred as the public key since it is designed to be made public so that anyone can use it to encrypt messages. The decryption key is often referred as the private key since the encrypted messages should only be able to be decrypted by the person with the private key. Two of the most popular asymmetric key ciphers are RSA and ECC.

1.1.1 DES

Data Encryption Standard (DES) is a symmetric key block cipher that was first selected by the National Bureau of Standards, now the National Institute of Standards and Technology, as the official Federal Information Processing Standard in 1976 [12]. DES was developed by a team from IBM, and it was based on an earlier cipher called Lucifer. The National Security Agency (NSA) was also involved in the final design of DES, and it was later discovered that the NSA's involvement actually helped increase the strength of the cipher by the means of differential cryptanalysis [10].

DES is a 64 bits key block cipher with a block size of 64 bits. However, only 56 bits of the key is actually used, and every 8th bits of the key serves as a parity check of the previous 7 bits and is discarded during the cryptographic operations. Thus the effective key length of DES is only 56 bits. Since the effective key length of DES is only 56 bits, it is susceptible to brute force attacks with this key length. There are some successful brute force attacks that run in less than a day on modern machines [14]. DES has been widely adopted since its inception, and there are many hardware specifically designed to implement the DES algorithm to speed up the encryption and decryption processes. In order to extend the life time of these hardware while not adopting to other more secured ciphers, DES users adopted the use Triple DES, or sometime referred as 3DES. Triple DES can either have a set of two 56 bits keys or a set of three 56 bits keys [13]. In the two keys scheme, the plain text is first encrypted with key number one, then it is decrypted with key number two, and it is finally encrypted with key number one again. In the other scheme, the plain text is encrypted with key one, then it is decrypted with key two, and it is encrypted with key number three. The plain text is recovered from the cipher text by applying the

decryption function, encryption function, and decryption function again with the keys in reverse order in which they were applied during the encryption phase. Triple DES allows users to keep using their old hardware accelerator while increase the effective key length to either 112 bits or 168 bits depending on the Triple DES scheme used. Although DES and 3DES is a popular cipher, we will not be performing side channel attack on DES or 3DES, and we will being performing side channel attacks AES.

1.1.2 AES

Advanced Encryption Standard (AES) is a symmetric key block cipher. AES was adopted in order to replace DES. AES was originally called the Rijndael cipher and was developed by two Belgian cryptographers. The Rijndael cipher was submitted as part of the AES selection process. In November 2001, AES was announced by the National Institute of Standards and Technology as the winner of the selection process. In addition, the NSA also approved AES for protecting documents as high up in the classification level as top secret [18].

As mentioned before, AES is a block cipher, and it has a fixed block size of 128 bits, and the key length can be 128 bits, 192 bits, or 256 bits. Unlike its predecessor DES, AES is not a Feistel cipher meaning that the plain text is not divided into two halves and swapped during each round. On the other hands, AES does have many identical rounds, and the number of rounds depends on the key size: ten rounds for a 128 bits key, twelves rounds for a 192 bits key, and fourteen rounds for a 256 bits key. The S-box used in AES is called the Rijndael S-box, and it is used in both the key scheduling algorithm and the round function. The Rijndael S-box is generated by finding the multiplicative inverse for a given number in the Rijndael's finite field denoted by $GF(2^8)$, and the multiplication operation is multiplication of polynomials

modulo of $m(x) = x^8 + x^4 + x^3 + x + 1$, which is an irreducible polynomial of degree 8 [10, 18].

In AES, everything is put into a four by four matrix of bytes called the state, and there are four common operations that operate on the state throughout the entire AES algorithm. The first operation is called the AddRoundKey, and the AddRoundKey operation takes the subkey, derived from the main key using the Rijndael's key scheduling algorithm, and XOR each byte in the state. The second operation is called SubBytes, and it is simply looking up each byte in the state and replacing it with the values found in the Rijndael S-box. Similar to DES, the Rijndael's S-box introduced non-linear operations into the algorithm. The next operation is called the ShiftRows. As the name implies, each rows in the state is being shifted. The first row doesn't change; the second row is shifted to the left by one; the third row is shifted to the left by two; and forth row is shifted to the left by three. Finally, the last operation is called MixColumns, and each column in the state is being multiplied by a known matrix depending on the key size. All the operations are depicted in figure 1 This operation is closely relately to the Rijndael finite field, and the multiplication is actually a valid operation in the Rijndael finite field. For each AES encryption, there is an initial round, a final round, and the ten, or twelves, or fourteen rounds functions. The initial round consists only of a single AddRoundKey operation, and the final round consists of one of SubBytes, ShiftRows, and AddRoundKey operations. Each of the round function will have all four operations described earlier [18].

Due to its key size, it is infeasible to perform a brute force attack on AES. A known attack on AES is a related key attack on both the 192 bit and 256 bit version of AES [19]. The complexity of this attack is of 2^{100} computations. Other known successful

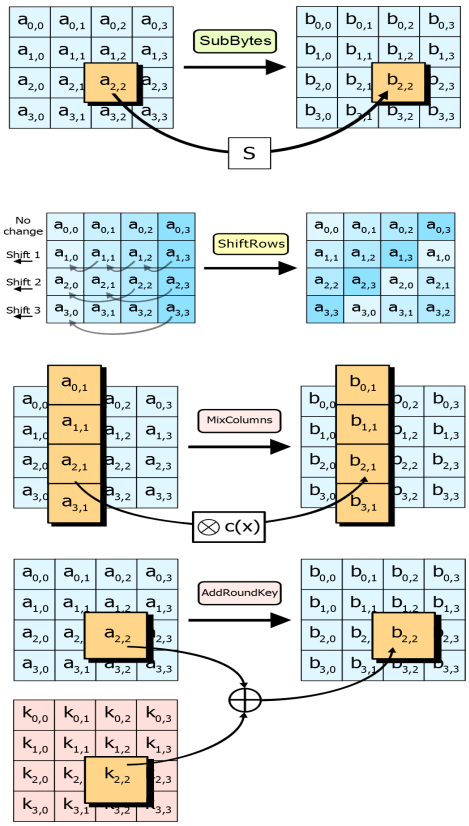

Figure 1: The operations of AES: SubBytes, ShiftRows, MixColumns, and AddRoundKey

attacks on implementations of AES are side channel attacks where information leaked from the implementation of the algorithm is used to recover the secret key [1, 3]. These side channel attacks are completely dependent of the implementation, and some of these attacks are also chosen plain text attacks. However, the attackers need to be able to have access to the hardware in order to perform the side channel attacks successfully.

CHAPTER 2

Side Channel Attacks

Side channel attacks are attacks on the hardware implementations of cryptographic algorithms or cryptography devices that do not employ full brute force attack or exploit the weaknesses of the algorithms themselves. In general, the goal of side channel attacks is to gain the secret information, such as the secret key, from the cryptography device with the information leakage due to its implementation on the system. There are numerous successful side channel attacks that can be used to recover the secrets stored within a cryptography system that uses secured algorithms such as AES and RSA [24].

2.1 Timing

One side channel attack on a cryptography system is to measure the time it takes for a cryptography system to perform a computation. The assumption in a timing attack is that the computation is data dependent, and the attacker can recover the data based on the amount of time that the system takes to performance the calculation. A simple example is a program that takes a series of bits as input and perform calculations based on each bit. The program, however, would take longer to calculate the result if the bit is one. The attackers can recover the input bit by bit from judging the timing of the calculation; if the calculation was short, then the bit is zero, otherwise the bit is one. Another example is pin number validation. A naive approach to write a pin validation program is to check the input bytes one at a time against the correct pin number, and if there is a difference between the input and the correct pin number, then the program would terminate. Since a correct first byte means that the program

will check the second byte, the calculation would take sightly longer for a correct first byte input versus an incorrect first byte input. Thus, the attack can mount a timing attack to determine the correct pin number by testing the all possible first byte until a longer calculation is measured, and repeat until all the pin numbers have been revealed. In reality, a successful timing attack was used to recover a server's private key that uses RSA [25].

2.2 Sound

Another side channel attack is the use of sound as a channel of attack. The sound that hardware produces can leak information regarding the secret information. Primitive examples are keyboard and key pads. In keyboards or key pads, each key produces a different sound, and an attacker can record the sound that the keyboard or key pad makes and determine the keys pressed. The attackers can determine users' passwords if they managed to capture the sound that the keyboards or key pads make during the time when the users entered their passwords. A more sophisticated sound based side channel attack was demonstrated by Adi Shamir by capturing the humming emission produced by the capacitors surrounding the processing unit that is performing cryptographic operations [26].

2.3 Power Consumption

A more complex form of the timing based side channel attack is taking the power consumption reading of the system of interest while performing cryptographic operations [1]. Like the temperature based attacks, the attackers simply record the power reading during cryptographic operations, and attackers can use that information to determine the secret hidden within the system. Since most modern cryptographic

devices are implemented using transistors, a charge will be applied, or removed, from the transistors when electrons flow through the transistors. This change in charge can be detected and can leak secret information. The leakage occurs due to different transistors becoming active, or inactive, based on the secret information [2]. The attackers do not necessary needed to know the details regarding the system of interests. In fact, the system of interests can be a black box, and the attack will still be successful.

There are two types of power consumption based side channel attacks, and they are Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Both of them were first introduced by Paul Kocher in 1998 [2]. The details of these two types of attacks will be discussed more in section 4.1.

2.4 Electromagnetic

Using electromagnetic field as a channel for side channel attacks is closely related to using power reading. While a power based side channel attack take the power measurements from the entire system of interests, an electromagnetic based attack localizes on the area of where the measurements will take place, such as where the cryptographic functional unit resides. Since the measurements are taken at a specific location, a specialized probe is generally needed in order to the take the measurements. Once again, the leakage of information occurs due to electrons flowing through the transistors, and the act of electrons passing through the transistors will also produce electromagnetic radiation which the probe can detect [1, 3]. In addition, the location of the measurements has to be very precises in order to get any useful information.

Similar to the power based side channel attacks, there are two types of attacks, and they are Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA) [3]. However, there is no conclusive evidences to show that whether a power based or a electromagnetic based side channel attack is more effective or vise versa [31, 32].

CHAPTER 3

SPA and SEMA

3.1 SPA

Simple Power Analysis, or SPA for short, is a power based side channel attack. In an SPA attack, attackers take the power measurements from the system of interests while it is performing cryptographic operations, or other operations of interest, and the attackers would visually inspect, or apply a template attack on, the power readings and determine the information leakage. A power measurements over a fixed period of time is usually referred as a trace [2]. It should be noted that SPA only requires a small amount of traces, and it usually takes less than a thousand traces in order to perform an SPA attack. If a successful attack only required one trace, and it is referred as a single-shot SPA attack; and if a successful attack required more than one trace, then it is called a multiple-shot SPA attack. In a multiple-shot SPA attack, the attacker can either supply the cryptographic system with either the same plain text or different plain text for each traces. The advantage of doing a multiple-shot SPA attack is to reduce the noises within the traces and get a clearer picture of what is happening in the system from the traces [1].

One of the base assumption of SPA is that the cryptographic operations of interests is running in sequential order. Under this assumption, a power trace will contain the power in terms of voltage on the Y-axis, and the X-axis will be time. For example, AES consists one or more of four general operations in each round. Recall that the four operations are AddRoundKey, SubBytes, ShiftRows, and MixColumn, and each of these operations will produce unique signatures within the power traces. Since the

algorithm will run in sequential order, and there are at least ten repetitive round of these four operations. Thus it is easy to identify the AES operations in the power trace if the attackers simply look for these ten unique signatures for software implementation of AES. In addition, attackers can also learn the information about the implementation of the algorithms solely based on the power traces [11]. For example, a system will require less clock cycles to access internal memory than accessing external memory; and the system performing input/output operations will draw more power than simple operations. Finally, if the operations of algorithms of interests is data dependent, such as the input secret key, then it is possible to obtain the data just by exampling the power traces [1]. On the other hands, the electromagnetic counter part of SPA is Simple Electromagnetic Analysis.

3.2 SEMA

Simple Electromagnetic Analysis, or SEMA for short, is a side channel attack that reads the electromagnetic field from the circuit that is being attacked. SEMA is very similar to SPA; SEMA attacks take the electromagnetic field reading from the circuit while it is performing cryptographic operations [3, 4]. The attacker will visually inspect the electromagnetic traces and determine the parts of the traces that correspond to the cryptographic operations performed, and the attacker will eventually discover the secret key based on these traces. However, SEMA attacks is localized to only a small portion of the circuits, so an extra step of finding out where the area of interested on the circuits is needed. Locating the area of interests on a circuit is done by forcing the cryptographic device to run in a loop while the electromagnetic probe takes partial snap shops of the circuit during each cycle of the loop. Once a susceptible area has been identified, the search will be refined to that

area [1].

Unlike SPA or DPA that can obtain traces by simply measuring the power consumption of a system by measuring either ends of the power source of the system, SEMA requires a specialized electromagnetic probe to take precise measurements. An inductive probe will be used for performing SEMA attacks; an inductive probe is simply a wire looped into coils, and the coils' diameter can range from 150 to 500 microns. When an electromagnetic field passes through the coils, the coils will act like an inductor and will induce current through the wire, and the strength of the electromagnetic field can be measured from the wire. When a transistor switches from a zero to a one or vice versa, a short current pulse is produced and can cause the surrounding electromagnetic field to fluctuate, and this fluctuation can be detected by using the inductive probe. This fluctuation can be a form of information leakage. This leakage of information can correlate the transition's Hamming distance [3].

CHAPTER 4

DPA and DEMA

4.1 DPA

Differential Power Analysis, or DPA for short, is a popular type of power analysis attacks on cryptographic system [2]. The main reason for DPA's popularity comes from the advantage of not requiring the detail knowledge of the system of interests, and the attacker only needs to know the algorithm that is being employed on the system. However, thousands of traces are needed in order for the attack to be successful with noisy signals. In general, DPA pre-computes a series of possible power traces and compares these power traces against the actually power traces taken. The traces with the highest correlation will most likely reveal the key [1].

DPA consists of five steps. The first step is to choose an intermediate result of the cryptographic algorithm running on the system. The intermediate result should be a function in the cryptographic algorithm, and a series of chosen plain text and possible values of the secret key should be used to applied to the function in the later steps. One example of a function is the Sbox in either DES or AES. The next step is to take the actual power traces with the cryptographic system, and the attacker should use all the chosen plain text from step one. The trace is then divided based on the operations of the cryptographic algorithm observed. From this step, the matrix T is created from the power traces and the recorded data values from applying the encryption or decryption function, and it is important that all the column should correspond to the same operations. The next step is to calculate the hypothetical intermediate values. The end result of this step is also a matrix, V. The attacker calculates all the

possible parts of the secret key. The entries in V would be applying the intermediate value function chosen and the chosen plain text from step one with these hypothetical keys. Hence, entry $v_{i,j} = f(d_i, k_j$ where $i = 1, ..., D$ and $j = 1, ..., K)$, where D are the intermediate values from step 1 and K are all the possible partial key values. The fourth step in DPA is to take the matrix V and map it to the power consumption values and produce the matrix H. In this step, the attacker uses simulation to obtain the hypothetical power consumption based on the hypothetical intermediate values. This simulation is usually done with the help of a power model, and the most commonly used power models are Hamming distance and Hamming weights. Each entry of H, $h_{i,j}$, is simulated from $v_{i,j}$ The last step of DPA is performing statistical analysis between the matrix H and the matrix T. The analysis is done column wise, and a high correlation between a column on H and T means the corresponding key hypothesis from that column is most likely be the partial key used during the encryption phase [1]. Figure 2 shows step 3 to 5 in picture form.

The more measurements that the attacker makes during step two, the more likely that the attacker will be able to the recover the key [11]. If it is the case that all the correlation values on a column are all very similar, then it is most likely the case that the attacker has not taken enough measurements to draw a strong correlation between the key hypothesis and the actual key, and the other possibility is that the key hypothesis is wrong [1]. In our experiments, we want to record the time it takes for full key recovery, and the measurement of time includes acquisitions and the analysis phase. Realistically, an attack would to be able to perform full key recovery in less than a day, and perform full key recovery in less than an hour in some cases.

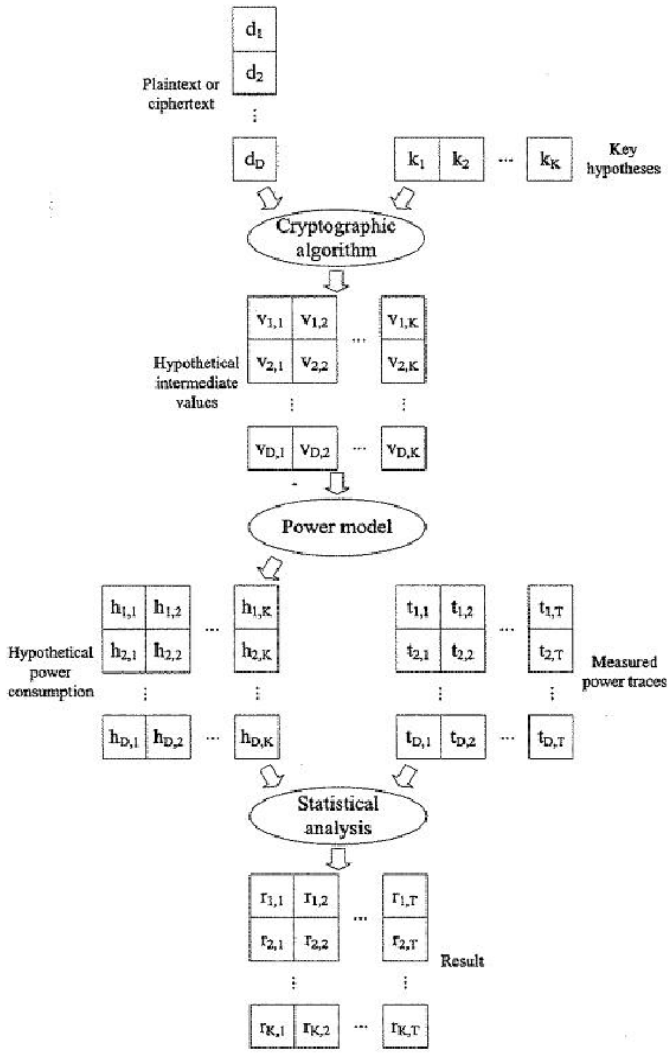

Figure 2: Depiction of the steps in DPA

4.2 DEMA

Differential Electromagnetic Analysis, or DEMA for short, is similar to DPA except with electromagnetic readings instead of power readings. Much like DPA, DEMA follows the same five steps to determine the secret key in a cryptographic system. Rather than exploiting the relationship between power consumption and the data being processed, DEMA takes a localized reading of the electromagnetic field of the circuit. There has been successful DEMA attacks against an FPGA based DES and ECC

cryptographic system with less than ten thousands traces [6, 8]. In addition, there are also successful attacks on systems that perform AES encryption [5]. Currently, it is unknown whether DPA or DEMA is more efficient in obtaining the secret key from an embedded cryptographic system [3].

4.3 SPA vs DPA

The major difference between SPA and DPA is the number of traces required in each attack. In general, SPA requires only a small amount of traces, and the number of traces is usually less than a thousand. On the other hands, DPA requires thousands of traces, and DPA sometime required up to the range of millions traces [11]. Furthermore, SPA usually requires the attacker to know some detail implementation of the cryptographic system whereas attacker using DPA can treat the cryptographic system as a black box [1].

4.4 Power Models

As mentioned before in section 4.1, there are two commonly used power models for DPA, and they are Hamming weight and Hamming distance. Hamming distance is the number of different symbols in the same position between two same length strings. Hamming weight is the number of different symbols in the same position between a strings and a string of the same length with all symbols being zeros. If a better the model is used during DPA, then the amount of traces required to mount a successful attack is reduced. Both Hamming weight and distance can be used for attacking microcontroller made with CMOS technology [1].

4.5 Correlation Coefficient

Correlation coefficient is the most common way to determine the linear relationship between two values. The one of most effective method of calculating the correlation coefficient between two values is the Pearson's product-moment coefficient given by: $r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i}-\bar{h_i}\times(t_{d,j}-\bar{t_j}))}{\sqrt{\sum_{d=1}^{D}(h_{d,i}-\bar{h_i})^2\times\sum_{d=1}^{D}(t_{d,j}-\bar{t_j})^2}}$ where h_i, t_i denotes the ith columns of the H and T matrix described earlier, and $\bar{h_i}$ denotes the mean values of the columns. A coefficient of either a -1 or 1 means a strong correlation between the values, and a coefficient of zero means the values are completely independent of each other. The Pearson's product-moment coefficient is used for DPA because it can be calculated quickly [1].

4.6 DPA on AES

Performing DPA on different cryptographic algorithms requires different approaches. For instance, the DES algorithm states that only 48 bits of the key are used in the first round, and the remaining 8 bits of the key are used in the second round along with other bits of the key [12]. Thus, two rounds of DPA are needed in order to achieve full key recovery for DES. As for AES, all bits of the key are used in the first round of the encryption algorithm, so only one round of DPA is needed in order to achieve full key recovery [18]. Recall that AES has four main operations within each round function, and first operation is the SubByte operation. The SubByte operation takes each byte in the state and replace it with a value from the Sbox. During DPA, the hypothetical intermediate values matrix, V, is calculated based on the SubByte operation. Depending on the implementation of the AES algorithm, the Hamming Weight or the Hamming Distance for the change in values before and after the Sbox substitution can be leaked, so the intermediate value function is the SubByte in terms of DPA. The matrix V contains all the possible key bytes based on

the chosen plain text applying to the SubByte function. The rest of the DPA follows as described in the earlier section.

4.7 Countermeasures

Countermeasures are techniques that used to prevent side channel attacks. There are many ways to set up countermeasures on a device, and countermeasures can be done on the software side as well as the hardware side. The main goal of countermeasure is to minimize the amount of information leakage. More specifically, countermeasures make the power measurements, electromagnetic measurements, timing, or any other side channels be independent of the data processed. Some of the most common techniques for countermeasures are hiding, masking, dummy instruction insertion, randomized delays, non-deterministic computations, rail logic, and many more [4, 11]. In this project, we will not be concerning about any countermeasures that may arise from the embedded system.

CHAPTER 5

Experiments

In this paper, we will run several sets of experiments. The first set of experiments will be performing DEMA on an embedded system. This set of experiment will consist of using high and low sensitivity probes with different configurations. The different configurations consist of using hardware filters and performing DEMA at different locations on the embedded system. We want to see the effects of using high and low sensitivity probes as well as the effects of the different configurations. The next set of experiments is performing DPA on the same embedded system. This set will be divided into performing DPA with a power probe and a current probe. The experiments with the power probe will be measuring the strength of resistor in relationship with effectiveness of DPA; the results will be compared to the results of experimenting with the current probe. We will then compare the best results of DEMA and DPA to see which method is better for performing side channel analysis on embedded systems.

In addition to performing known side channel attack methods, we will be performing side channel attacks with the aid of a radio receiver. The goal of performing the experiments with a radio receiver is to see if we can reduce the total time of performing side channel attacks by reducing the amount of data with the radio receiver. We will repeat the experiments with DEMA and DPA with the radio receiver and see if there are any improvements in our results.

CHAPTER 6

Embedded Cryptographic Systems

In this project, we will be performing DPA on an embedded system that performs cryptographic operations. The system of interest is the ATXmega256A3B designed by Atmel, and we will be using an evaluation board for this project.

6.1 Atmel AVR ATXmega256A3B

For our experiments, we will be using the ATXmega256A3B embedded microprocessor. The Xmega family is designed and produced by Atmel. The ATXmega256A3B is designed for large range of applications included but not limited to building, industrial, motor, board, climate control, hand-held battery applications, networking and home appliances, and medical devices. The ATXmega256A3B has a total of 64 pins with a max I/O pins of 47. The main CPU of the chip is an 8 bit AVR. In terms of memory, the ATXmega256A3B has 256 KB of in-system self-programmable Flash, 8 KB of boot code sections with lock bits, 4 KB of EEPROM, and 16 KB of internal SRAM. In addition, the chip has a set of internal 32 KHz, 2 MHz, and 32 MHz oscillators operating at 3.3 V and 150 mA. Finally, the chip has a cryptographic engine that can perform DES and AES which would be useful for this project [27]. The ATXmega256A3B microprocessor will simply be referred as Xmega from this point on.

6.2 Implementation of AES on ATXmega256A3B

The implementation of AES on the Xmega is hardware based, and the AES hardware accelerator is directly connected to the main CPU of the chip as shown in figure

3 [28]. However, the block diagram does not show us where the cryptographic accelerator is physically located for DEMA since the block diagram does not translate to physical layout, and locating the cryptographic accelerator will be the first step in DEMA. The AES and DES engine can be accessed via software from a couple of lines of code. The programmer only needs to supply the key, plain text, and a buffer to store the cipher text for encryption. The AVR code loaded onto the Xmega for our experiments is provided in the appendix. The AES accelerator can only do 128-bit key encryption/decryption with 128-bit plain text blocks. It also supports XOR data load mode to the state memory for cipher block chaining (CBC). The encryption and decryption are performed in 375 clock cycles per 16-byte blocks [28].

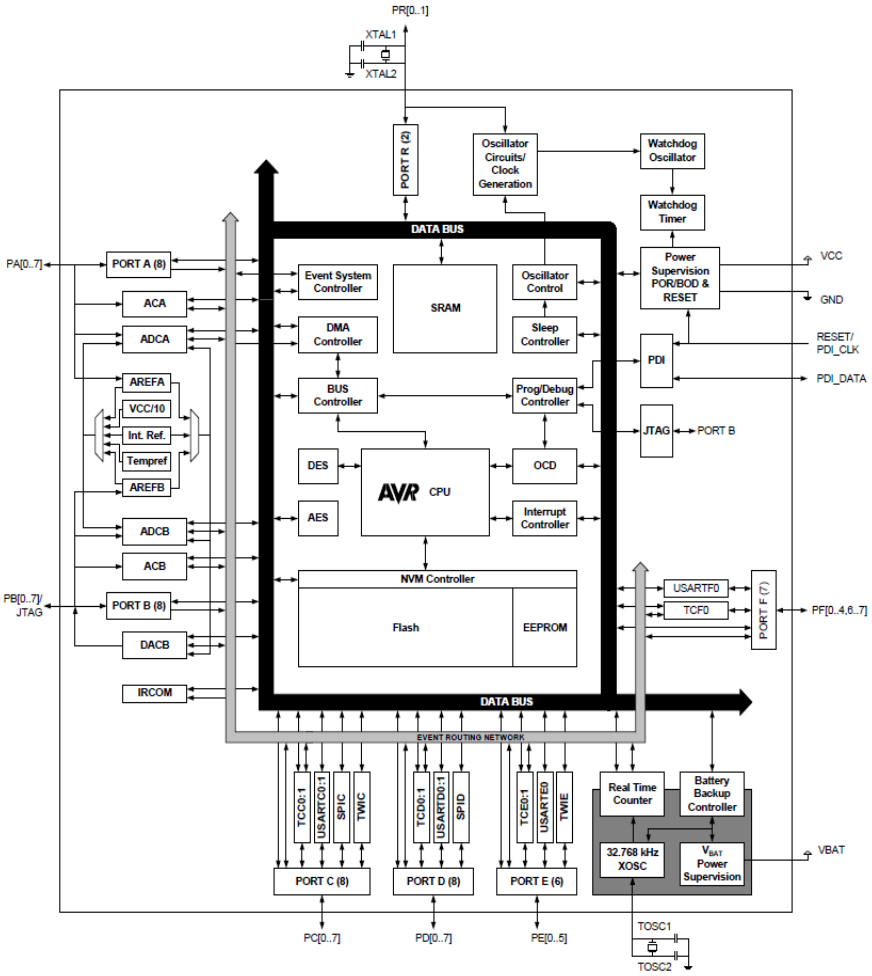

Figure 3: ATXmega256A3B Block Diagram

CHAPTER 7

General set up of the attack

The set up of the attack will be simple. First and fore most, a PC will be set up with Inspector. Inspector is a platform designed to analysis signals developed by Riscure. Inspector has many preloaded modules for signal processing for performing DEMA and DPA for DES, 3DES, AES, RSA, and ECC, and the modules can be modified to suits our needs. The PC is wired to an oscilloscope, and we will be using a LeCroy 610zi and/or Picoscope 5203 to capture signals. The oscilloscope is wired to the EM probe or power tracer directly. In addition, the PC will also be communicating with the system of interest in order to trigger cryptographic operations. Figure 4 shows a generic set up of the attack.

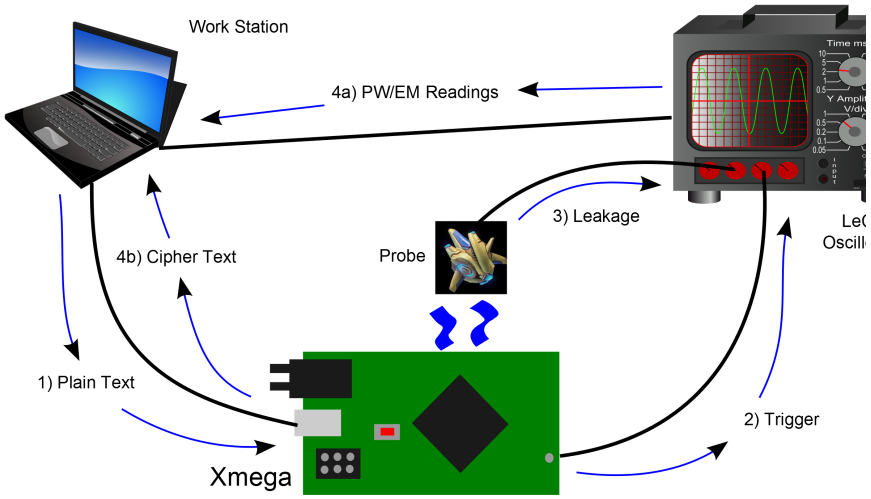

Figure 4: Generic set up diagram for SCA

In order for the PC to take a trace, it begins by sending the system of interest a command. The command contains the opcode for encryption and parameters (such as data to encrypt, mode of encryption, etc), and this will force the system of interest

into performing cryptographic operations. This step is shown in number 1 of figure 4. At the same time, the PC will send a trigger to the oscilloscope to begin recording the signals from either the power probe or the EM probe. This is shown in number 2 of figure 4. Once the cryptographic operations have finished, the oscilloscope will send the signals to the PC and a trace containing the signal, plain text and cipher text will be recorded (Number 4a/b in figure 4). Ideally, the trigger will be set to begin right before the cryptographic operations start and the signal will end right as the cryptographic operations end. However, fine tunings will needed to be made in order to produce well aligned trace set. As mentioned before in section 4.1, the more traces that the attacker can acquire the more likely that the attacker will be able to obtain the secret key.

Once we have a trace set, we will begin our analysis phase. Assuming the trace set is sampled at 1GHz, the first step in our analysis phase is to determine which frequencies show the most activities. We can determine this by running the trace set on the Spectrum module of Inspector. Further details about the Spectrum module will be described in section 9.3.1. Once we figured out which frequencies are the most active in the trace set, we will resample the trace set to these frequencies and produce new trace sets. Resampling means applying a low pass filter to the data and change the sampling rate of the signal. A low pass filter will only allow low frequencies signal to pass through, thus eliminating all the high frequency signals. The next step in the analysis phase is to align the trace set at the important section (e.g. during the first round of AES encryption). We can achieve this by the means of the Static Alignment module of Inspector. The details of this module will be discussed in section 9.3.1. We can determine which section of the trace is important by either visual inspection, running the data correlation module (details in section

9.3.1), or the KnownKeyCorrelation module if we know the key (details in section 11.2). Finally, we will perform DPA/DEMA on the trace set

7.1 Expectation

In this project, the main goal is to compare and contrast the effectiveness of DPA and DEMA on embedded systems. One of the methods of determining which analysis technique is more effective is by the number of traces required in order to the break the cryptography. The signal measurement technique that requires less traces will be the more effective technique. From previous experience, we would expect DPA to require less number of traces in order to successfully perform the attack than DEMA. This is due to the noise capture by the EM probes that causes the traces in DEMA to be more noisy. As a result, DEMA needs more traces in order to cancel out the noises.

7.2 Locating the X-Y Coordinate of the Crypto Block

Upon learning the implementation of AES of the embedded system, we can begin the attack on the embedded system by physically locating where the cryptographic operation is being performed on the chip. This step is important in order to perform DEMA on the embedded system. Recall that DEMA requires very localized electromagnetic readings on the system of interest, thus we will need to know the physical location of the cryptographic operation before we can perform DEMA on the embedded system.

To set up this step in the process, we will first force the embedded system to continuously perform cryptographic operations in a loop. This can be done by the means

of sending commands to the system, wait for a respond, and repeat. The next step is to divide the chip into 10 by 10 equal sections and begin taking raw electromagnetic measurements of each section while running the cryptographic loop. By adjusting the measurements to only filter to the frequency in which the cryptographic blocks operate in, we can see the hot spots that correspond to the area of the cryptographic engine. We can narrow the area down by performing the measurements in a smaller area around the hot spot until we can identify where exactly is the cryptographic blocks are. The full detail on how this is done on the Xmega will be described in section 9.1.

7.3 Setting up the trigger

The next step, in both DEMA and DPA, is set up a good trigger. An ideal trigger will allow the attacker to take traces that start at the exact moment when the cryptographic operations start and end at the moment when the cryptographic operations end. A hardware trigger is chosen for this project since we have full access to the hardware, and we can load any firmware we desired. The full detail of this trigger will be described in section 9.2.

CHAPTER 8

Smart Triggering

In side channel attacks, setting up a good trigger is very important, and a good trigger can reduce the time it takes to perform side channel attacks. A perfect trigger can caused all the traces to be aligned without any modifications to the trace set. By eliminating the alignment phase during the acquisition phase, the attacker can save time by not performing alignment on the trace set. However, a perfect trigger is hard to obtain unless the attacker has full control of the hardware. We will discuss how we set up our trigger for the Xmega, and we will discuss a new approach to finding a good trigger.

The icWaves is an Inspector module that is designed specifically for setting up a good trigger. Referring to figure 4, the icWaves will reside between 2 and 3. The signal from the probe will go directly to the scope as well as the icWaves. The trigger line, number 3 in the figure, will go directly to the icWaves, and the icWaves has it own trigger line that goes into the scope. The attacker can load a reference pattern as large as 256 samples onto the icWaves via Inspector. During acquisition, icWaves will compare the incoming signal from the probe to the reference pattern. The comparison that the icWaves makes is sum of absolute difference (SAD). Note the two very similar patterns will have a SAD value close to zero, so identical patterns will have a SAD value of zero. The icWaves will send a signal out of its trigger port connected to the scope if the calculated SAD value is below a certain threshold set by the attacker. The biggest problem with using the icWaves for triggering is finding a good reference pattern.

In this section, we will discuss a new Inspector module that allows us to find a good reference pattern, and we will also discuss the performance and accuracy of this module. The goal of this module is to determine the best 256 samples pattern to load into the icWaves as the reference trace given a trace set. The input of this module will be a set of aligned traces, and the output of the module will be a 256 samples pattern with a suggested threshold for the icWaves. The chosen area, by the user, from the input trace set will be used to generate patterns as candidates for triggering, and we will refer this area as inside from now on. The only additional options that the user can decide for this module is the offset. The offset will affect the pattern generation and the SAD calculations.

We will demonstrate how the offset works by describing how the pattern generation works with two different offsets. The user can choose to use the following offsets: 1, 64, 128, and 256. Let T be a trace set with 100 traces, and each $t_n \in T$ trace has 30000 samples. Let the range of samples [1000, 3000] be the inside area selected by the user. If the offset is 1, then the pattern set, P, will consist of the followings: $p_1 = [1000, 1255], p_2 = [1001, 1256], p_3 = [1002, 1257]..., p_m = [2744, 3000]$ from trace 0 of T. If the offset is 256, then the pattern set, P, will consist of the followings: $p_1 = [1000, 1255], p_2 = [1256, 1512], p_3 = [1513, 1768]..., p_m = [2536, 2792]$ from trace 0 of T. Each of these patterns will be a candidate for the final output of the module. Once the module has generated the patterns, it will begin to perform the SAD value calculations against the other traces.

The SAD calculations can be optimized from the brute force method, and we will describe the brute force method and the optimized method for offset less than or equal to 128. Let T be a trace set containing n traces, and each trace contains k samples.

From the user's selection, the algorithm was able to produce m patterns. Let SAD() denotes the sum of absolute difference function between two arrays of numbers of the same size, and let $t_n[i, j]$ denotes the samples ranged from i to j in the trace t_n.

8.1 Brute Force Method

In this section, we will discuss the brute force method of calculating the SAD values. For this discussion, we will assume the offset is 1. For each $t_n \in T$, for each $p_m \in P$, calculate $\text{SAD}(t_n[0, 255], p_m)$, $\text{SAD}(t_n[1, 256], p_m)$, ... $\text{SAD}(t_n[k - 256, k], p_m)$. The run time of the brute force algorithm is $O(n^3)$ where n is the number of samples in a trace.

8.2 Optimized SAD Calculations

In this section, we will discuss the optimized method of calculating the SAD values. For this discussion, we will assume the offset is 1. There are two optimizations that we can employ. Assuming that the user selected i to j from the trace, then the first optimization that we can employ is to calculate SAD values only up to j. For example, if the user selects the samples from 2000 to 3256, then perform SAD calculations on the followings: $\text{SAD}(t_n[0, 255], p_m)$, $\text{SAD}(t_n[1, 256], p_m)$, ... , $\text{SAD}(t_n[3000, 3256], p_m)$ for each $t_n \in T$, for each $p_m \in P$. We can do this because we are not interested in any pattern that can trigger beyond the selected area, and any triggering happens beyond the selected area is already too late for any useful effect.

The second optimization comes from an observation that a lot of the SAD calculations are being repeated. For example, given a trace t and a pattern set P, and assuming offset is 1, note the following observation: $s_0 = \text{SAD}(t[0, 255], p_0)$, $s_1 =$

$\text{SAD}(t[1,256], p_1) = s_0$ - $|t[0] - p_0[0]|$ + $|t[256] - p_1[256]|$. Thus, the following algorithm can be used for calculating the SAD values: for each $t \in T$; let $i = 0$, calculate $s_0 = \text{SAD}(t[i, i+256], p_0)$, $s_1 = s_0$ - $|t[i] - p_0[0]|$ + $|t[i+256] - p_1[256]|$, $s_2 = s_1$ - $|t[i+1] - p_1[0]|$ + $|t[i+257] - p_2[256]|$, ... etc, then increment i by 1 and repeat until done. The run time of the optimized algorithm is $O(n^2)$ where n is the number of samples in a trace.

8.3 Optimized SAD Calculations Tests

Once the module was completed, we were able to perform a performance test and a reliability test on the optimized algorithm. The test trace set is a trace set consisting of 13000 traces, and each trace consist of 640000 samples. All the traces in the trace set has been aligned before any testing was done. The main feature of this trace set is that all of the traces contain seven peaks, and this trace set simulates a trace set taken from a real embedded system. Figure 5 shows this test trace set.

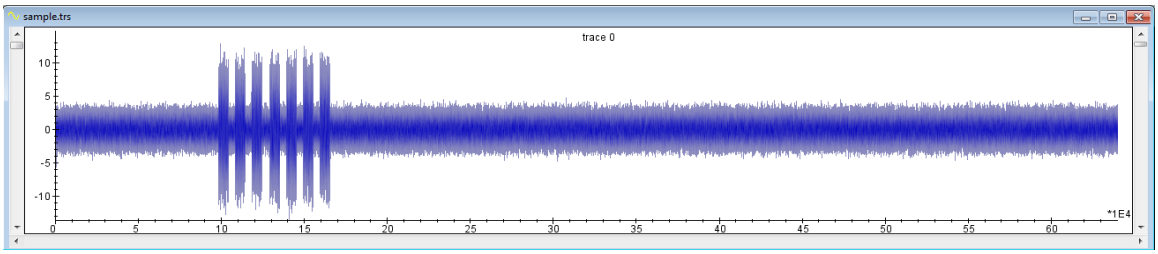

Figure 5: icWaves Test Trace Set

The first test we performed was the performance test. This test is designed to see if the optimized algorithm is faster than the brute force method. For this test, we selected samples 2027 starting at sample 107674. For comparison, a module with the brute force method with offset of 1 was able to complete all the calculations in 4 days on a machine with a 3.0GHz processor. On the other hands, the optimized module

takes about 20 hours to complete all the calculations with an offset of 1. Furthermore, if the offset is set to 64, the optimized module takes 30 minutes to complete all the calculations, and it takes 10 minutes to complete execution with offset set to 128. At offset 256, the module takes 5 minutes to complete execution. Note that a higher offset means less accurate result will be produced. Overall, the optimized module is provably faster in execution time than the brute force method.

The next test we performed was the reliability, and this test is designed to test the accuracy of the module. The offset is set to 1, and only 100 traces will be used in the following tests. For the first test (test 1), we selected 2000 samples centered at the raising edge of the second peak. Test 1 is shown in figure 6. Test 1 is designed to test the module in a high false positive environment. A false positive is when the pattern triggers outside of the selected area given a threshold, and a false negative is when the pattern failed to trigger in the selected area given a threshold. The result of test 1 is as follows: a pattern is selected at the beginning of the raising edge, and 50 false positives and 5 false negatives were reported. As explained, the high false positive was to be expected due to the design of this test.

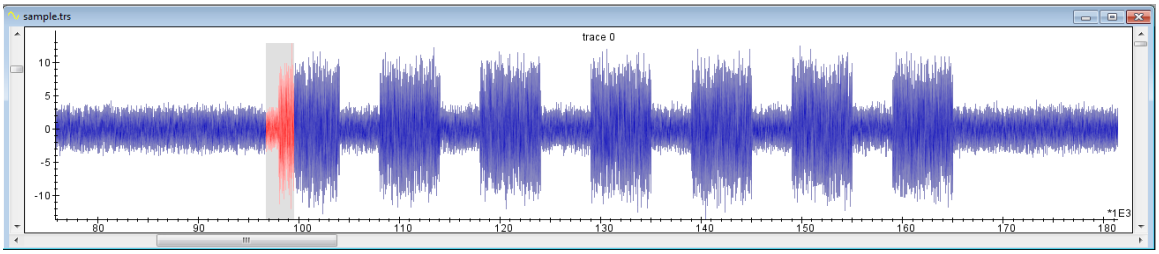

Figure 6: icWaves Test 1

Test 2 of the reliability test designed for low false positives, and it is designed to test the false negatives. For this test, we selected 2000 samples centered at the raising

edge of the first peak. Test 2 is shown in figure 7. The result of test 2 is as follows: a pattern is selected slightly after the beginning of the raising edge, and 5 false positives and 0 false negatives were reported. In other words, the module was able to find a pattern where icWaves will trigger early 5 percent of the time.

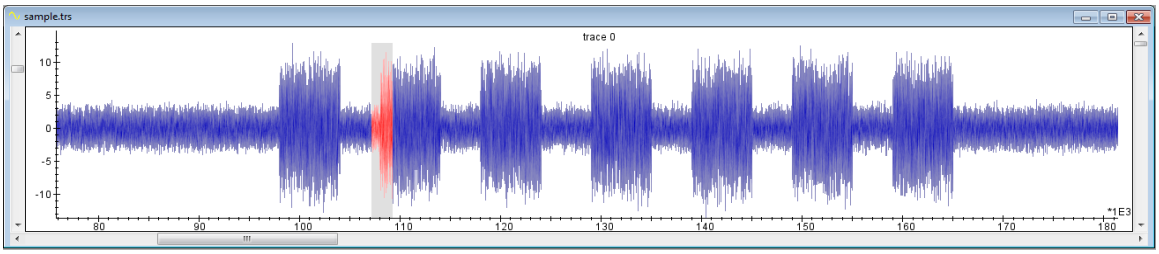

Figure 7: icWaves Test 2

The smart triggering is based on the observation that a lot of SAD calculations are being repeated. We developed an Inspector module that uses this fact to optimize the SAD calculations to find the best pattern for icWaves triggering. The result of executing the module shown that the optimization does indeed show improvement over the brute force method time wise. The optimized module is extremely accurate in finding the pattern with the least amount of false positives and false negatives.

CHAPTER 9

DEMA on ATXmega256A3B

In this chapter, we will describe in detail how we perform DEMA on the Xmega's AES cryptographic accelerator, and the results will be presented and compared to DPA in a later section. In our experiments, we would like to perform DEMA under different configurations, and we wanted to know the results of performing DEMA under these different configurations. The different configurations are the types of EM probe, the location of acquisitions, putting a hardware filter in our set up, and performing DEMA with resampled traces.

9.1 Locating X-Y on Xmega

For the Xmega, we have full control over the embedded system, so we program the Xmega to continuously perform AES cryptographic operation at 32MHz while taking measurements at different sections on the Xmega. The Xmega is divided into a 10 by 10 grid, and measurements are taken for each of these 10 by 10 blocks. The resulting traces are ran through a module for Inspector called Spectral Intensity. This module shows the amount of the average amplitude of the signal after applying a band pass filter as a grid. Blue means there are minimum activities at such frequencies at that location, and red means relatively maximum activities at such frequencies at that location. Figure 8 is the resulting grid of running the spectral intensity module on the trace set.

As we can see, there are two hot spots on the chips around the 32MHz frequency band. That means there a lot of activities on these parts of the chips around the

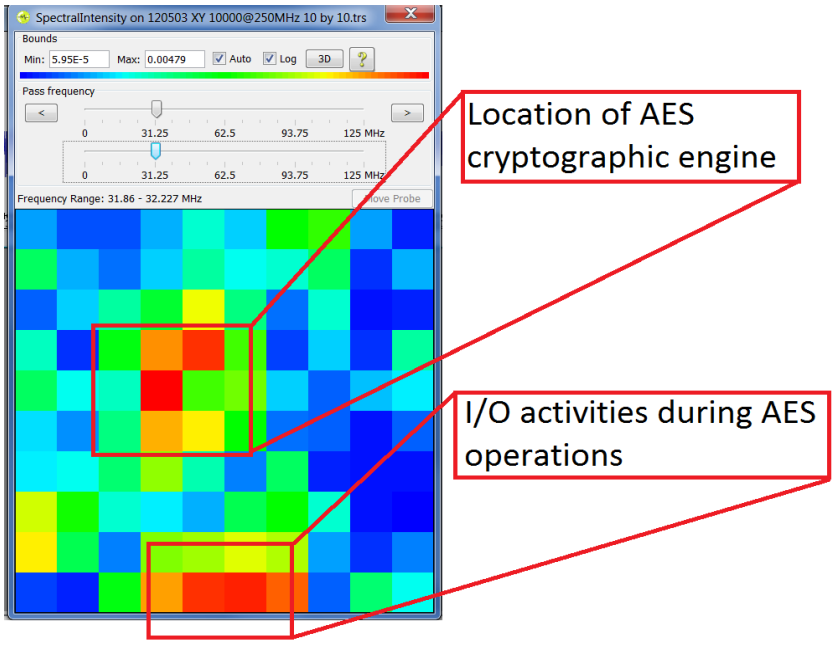

Figure 8: Spectral Intensity of Xmega during AES operations around 32MHz with +/- 0.2MHz bandwidth

32MHz frequencies band. Since the chip is programmed to only perform AES operations over and over again, we will assume that one of these hot spots is where the AES cryptographic engine is located. However, it is possible that this location can be where the clock generator is located or other components, and there is no telling what exactly is at that location without the actual circuit schematic of the Xmega. Furthermore, if we refer back to figure 3, we can see that the I/O operations are located on the edge of the chip, so we will assume that the AES cryptographic operations must be performed on the other hot spot. The I/O operations exist due to number 1 and numebr 4a/b from figure 4; the commands being sent from Inspector to the Xmega, and the responds from Xmega to Inspector are the caused of the I/O activities. From now on, we will refer the center hot spot as location 1 and the hot spot on the edge as location 2. Note that the Xmega is flipped 180 degree during the X-Y measurements, so the orientation of the spectral intensity versus the block dia-

gram is flipped. Thus, we have located the X-Y coordinate of the AES cryptographic operation on the Xmega, and we can begin taking a large amount of traces provided we have a good trigger.

9.2 Setting up the trigger for DEMA

Since we have full control over the embedded system, we could program the embedded system with a very accurate trigger. In this project, the Xmega is programmed to receive 16 bytes of randomly chosen plain text and recorded by Inspector, and the Xmega will send 16-bytes of cipher text back to Inspector. The key for encryption is programmed onto the embedded system. For this project, the secret key is chosen to be: 0x52 0x49 0x53 0x43 0x55 0x52 0x45 0x49 0x53 0x43 0x4F 0x4F 0x4C 0x21 0x31 0x00. In addition, port C1 (as shown in figure 3) will send out a 5 volts signal just before the AES encryption operation begins, and the signal will disappear as soon as the operation is completed. This port is connected to channel B of the LeCroy 610zi in order to trigger on the raising edge and begin taking traces.

9.3 DEMA at Location 1

9.3.1 DEMA with Low Sensitivity Probe at Location 1

Now that we have a good trigger, we can begin taking a large amount of traces. In order to minimize the number of samples in a trace while leave as much leakage as possible, we will begin taking traces as soon as the trigger is detected with no delays and end taking traces as soon as port C1 no longer has a 5 volts signal. As such, we will take traces at sampled at 1GHz with 26000-27000 samples in order to maximize the information we get from the traces within only the AES operations. There are two types of EM probes come with Inspector, and they are low sensitivity and high

sensitivity probes. As the name suggested, the high sensitivity probe can capture weaker signal. The trade off of using the high sensitivity is that it can also capture more noises as well. We will begin with using the low sensivity probe. Figure 9 is one of the million traces taken with the parameters described above.

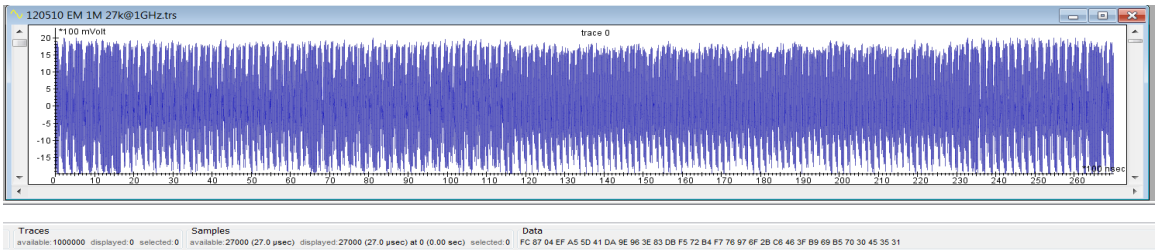

Figure 9: An EM trace taken at 1GHz with 27000 samples; LS probe

We can begin analysing the traces now that we have taken 1 million traces. Referring back to figure 9, we can take a note of a couple important details about the Xmega and its AES accelerator. First, the Xmega generates a 2 volts EM field of signal from where we positioned the EM probe. This 2 volts generation is affected by the position of the EM probe; the further away the probe is, then the voltage emission is also lowered. Second, the entire AES operation can be performed in 260 microseconds. As for performing DEMA/DPA, the plain text and cipher text are store as the first 16 bytes and last 16 bytes in the "Data" section of the trace respectively.

While we know that the Xmega is clocked at 32MHz, it is good verify this claim. We shall do so by running the Spectrum module of Inspector on the trace set. The Spectrum module will show all the frequencies that are most active given the trace set. Figure 10 is the result of running the Spectrum module on the trace set. The module is a fast Fourier transform based spectrum analyzer. Thus, we only see frequencies up to 500MHz in figure 10 instead of the sample rate of 1GHz due to the Nyquist

limit.

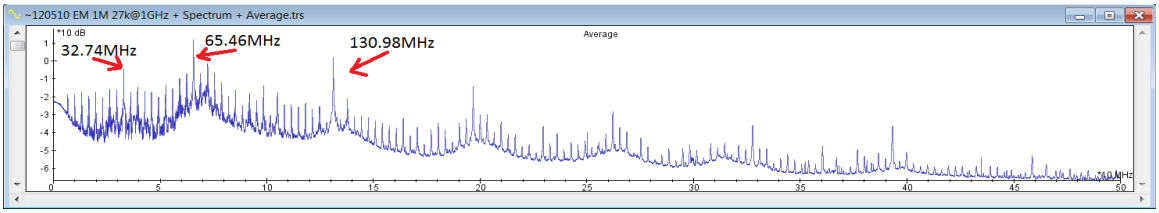

Figure 10: Spectrum of the EM trace set

As we can see, the frequency 32.74MHz stands out the most, and this tells us that the Xmega is not operating at exactly 32MHz. In addition, two upper harmonics frequencies also stands out the most: 65.46MHz and 130.98MHz. The next step in analyzing the trace set is to resample the traces at these three frequencies in order to get reduce the noises presented in the traces. The resampling is done via fast Fourier transform. Since we oversampled the traces to 1GHz, extra information are introduced to the trace set, and these extra informations will affect the success of DEMA. The results are shown in figure 11. In this figure, the example traces as the result of resampling to 32.74MHz, 65.46MHz, and 130.98MHz are shown. If we examine the trace resampled at 32.74MHz, we can see different regions. The first region is from 1.5 ms to 6.5 ms; the second region is from 6.5 ms to 11.5 ms; the third region is from 11.5 ms to 23.5 ms; and the last region is from 23.5 ms to 27 ms. For the next step, we would like to identify the exact timing of when the AES encryption occurs.

Now that we have reduced some of the noises presented in the trace set, we would like to identify the exact timing of the AES cryptographic operations. While we programmed the traces to be taken as soon as the AES operations begin, there are still operations within the AES operations that we want to exclude during DEMA such

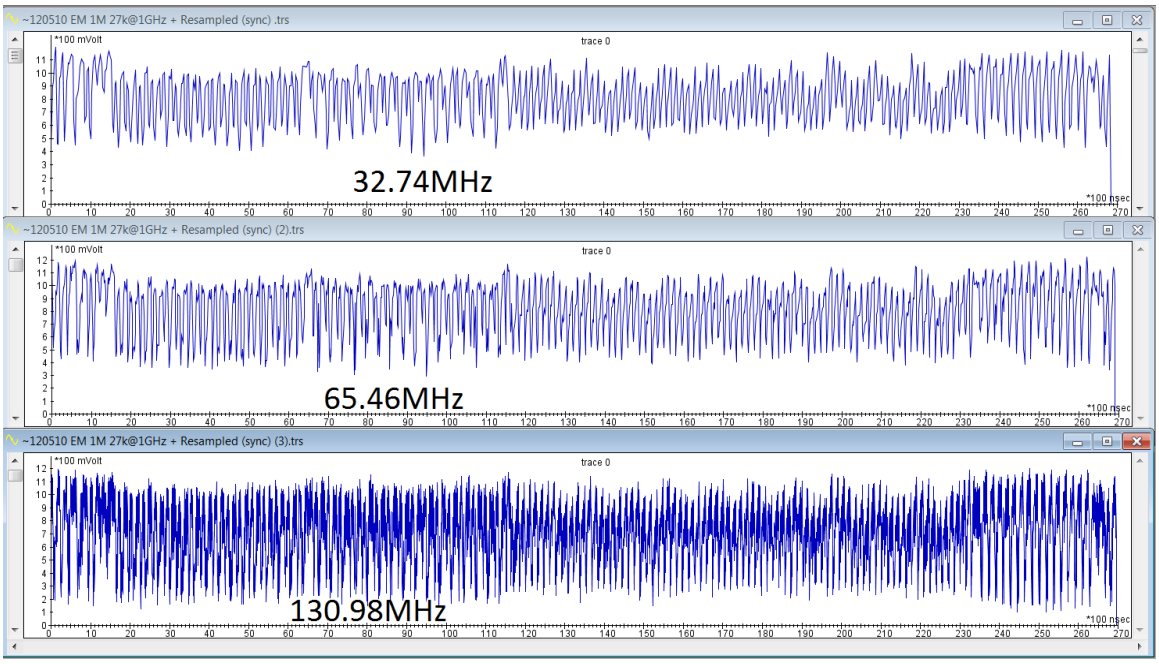

Figure 11: EM traces resampled

as input and output processing. By capturing signal with only the AES encryption operation, we can reduce the number of samples per trace, and this will reduce the time it takes to perform DEMA on the trace set. We can identify the actual AES encryption operation by identifying the AES input and output operations. We can do so by running the Correlation module. The correlation module will run a correlation on the specific bytes of the data section of each traces. Refer to section 4.5 for more detail about data correlation. A high correlation on a byte will correspond to the time when the byte is being input/output during the AES operations.

Figure 12 and 13 show the input and output data correlation on the trace set resampled at 32.74MHz respectively. Thus, it is logical to conclude that the AES encryption operations must reside in between these two sections of the trace. Figure 14 shows where the AES encryption is located within a trace.

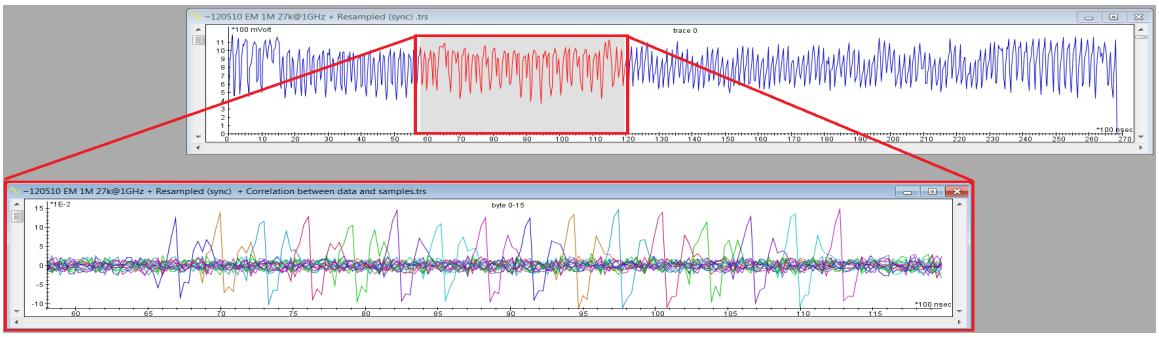

Figure 12: Correlation on the input data

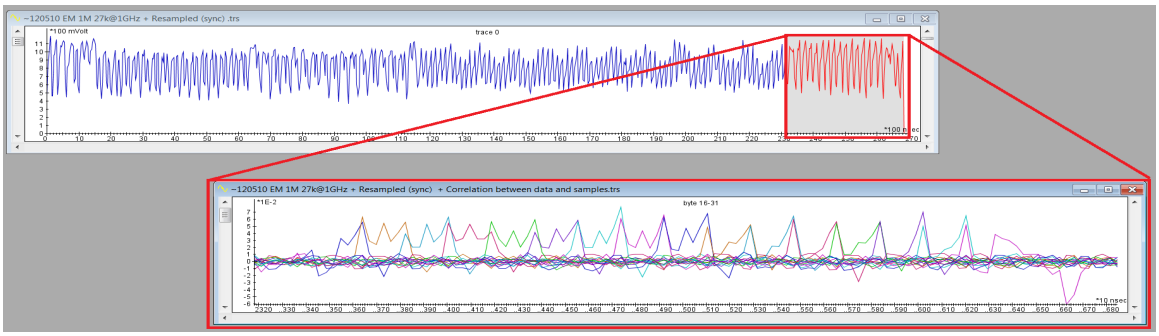

Figure 13: Correlation on the output data

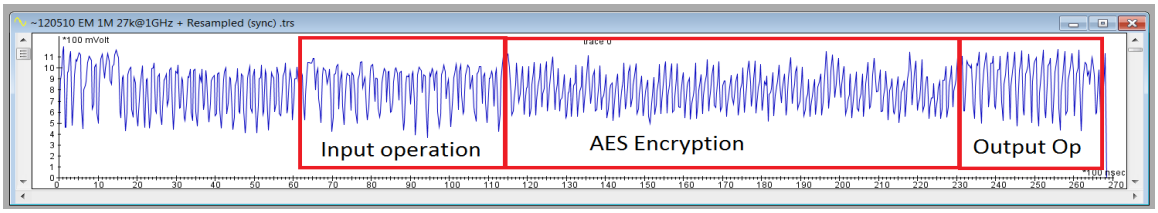

Figure 14: Location of AES encryption during the trace

The last step in this signal processing before we can perform DEMA is to align the first round of the AES encryption for all the traces in the trace set. This can be easily done with the Static Alignment module of Inspector. The Static Alignment module will allow user to select an area in the trace as well as allow user to specify a max shift range and a threshold. The module will shift the same range as the user selected area left and right up to the max shift range, and the module will calculate the correlation value for each shift. If the correlation value is above the threshold,

then the whole trace is shifted to that location. If the correlation value of each shift is below the threshold, the trace is thrown out of the trace set. Alignment is important in performing DEMA/DPA as the H matrix (Refer to section 4.1) is produced assuming all the traces are aligned. Once we have done static alignment for all three resampled trace set, we can begin perform DEMA on these trace sets. The FirstOrderAnalyst module of Inspector is ran on all these trace sets with Hamming Weight targeting the 1st round of SubByte (SBox), Hamming Distance targeting the 1st round of SubByte, and Hamming Distance targeting the 10th round of AddRoundKey as their power model. By running the different power model and target combinations, we can see leakage model of the Xmega and allow us to perform DEMA based on this leakge model. Table 1 shows the resulting keys recovering from the different trace sets, power model, and target combinations. Note that HW denotes Hamming Weight, HD denotes Hamming Distance, and "Round 10th round" denotes the AddRoundKey function of the 10th round.

Power Model/Target	Frequency	Key Recovered	Correct Key Bytes
HW/SBox 1st round	32.74MHz	5d465397005200495c4300004cd5314f	6
HW/SBox 1st round	65.46MHz	5d4953435524549534300004c2131f4	12
HW/SBox 1st round	130.98MHz	a200534355a0b54958434f000f21c1f4	7
HD/SBox 1st round	32.74MHz	3e83e0c0e6d16c253fc06c836c4da293	0
HD/SBox 1st round	65.46MHz	00ca3fc06c836c253fc06c836c926c4c	0
HD/SBox 1st round	130.98MHz	Not ran	–
HD/Round 10th round	32.74MHz	00459c7518da4201d36300ef4917ba23	0
HD/Round 10th round	65.46MHz	e4ca1f431c871c7ffefd991a7d2e2abd	1
HD/Round 10th round	130.98MHz	8300d3e527f9ba41bbcb33420b662840	0

Table 1: DEMA with Different power models and targets with 1 million traces

9.3.2 DEMA with High Sensitivity Probe at Location 1

As we can see, the best approach for DEMA on Xmega is using Hamming Weight as power model and target the first round of SBox. We nearly recover all of the key bytes. In order to capture as much leakage as possible within our traces, we began taking more traces with the high sensitivity (HS) probe. Table 2 shows the results of analyzing these new trace sets. We were able to capture 1.5M traces within 24 hours, and another 1.6M traces the next 24 hours. In between the captures, we perform DEMA on the 1.5M trace set. While the trace set resampled to 32.71MHz and 131.1MHz show improvement over their LS trace set, the trace set resampled to 65.43MHz shows no improvement even with HS probe and move traces in the trace set. In addition, doubling the amount of traces to 3.1M traces did not improve the result as we can see in table 2.

Power Model/Target	Frequency	Traces	Key Recovered	Correct Key Bytes
HW/SBox 1st round	32.71MHz	1.5M	00490043555200005343 4f004c003100	10
HW/SBox 1st round	65.43MHz	1.5M	00490043555245490043 4f4f00213100	12
HW/SBox 1st round	131.1MHz	1.5M	00490043555200005343 4f0000218d0d	8
HW/SBox 1st round	32.71MHz	3.1M	00490043555200005343 4f004c003100	10
HW/SBox 1st round	65.43MHz	3.1M	00490043555245495343 004f002131f4	11
HW/SBox 1st round	131.1MHz	3.1M	00490000555245005343 4f4f002149f4	9

Table 2: DEMA; Hi Sensitivity Probe

9.3.3 DEMA with High Sensitivity Probe and Hardware Filter at Location 1

Since switching to HS probe shown no improvement over the LS probe, we apply a hardware filter of 48MHz. In figure 4, the filter is placed between the probe and

the LeCroy at number 3. The hardware filter will eliminate all the high frequency noises during acquisition. Figure 15 shows an example trace with filter on as well as the spectrum of the trace set. A million traces were taken in this trace set.

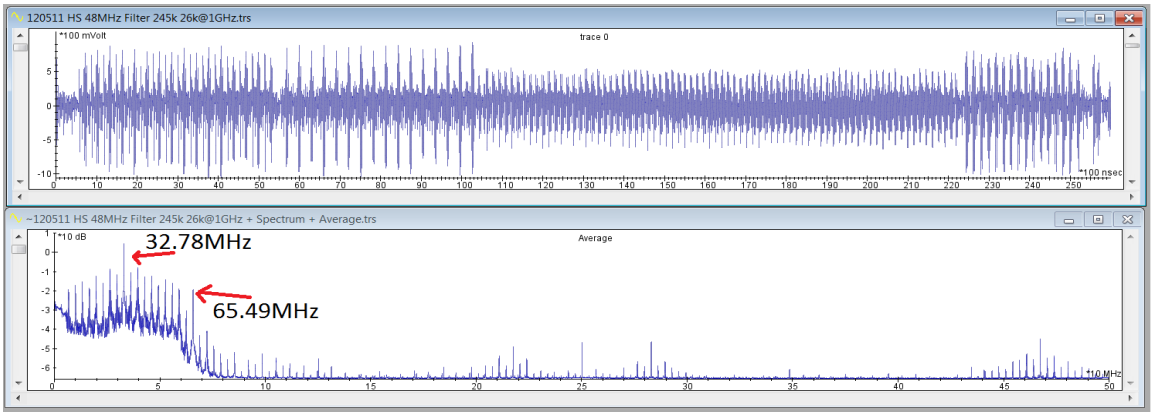

Figure 15: Sample trace and spectrum of filtered trace set

In this new trace set, we can see two dominating frequencies: 32.78MHz and 65.49MHz. We resampled the trace set to these two frequencies and the resulting traces are shown in figure 16. We began running the DEMA module on different power models and targets once again, and the results are presented in table 3. As we can see, the best approach would be attacking the first round of SBox using Hamming Weight as power model on 32.78MHz.

Table 4 shows the correlation between the number of traces and key bytes recovered on trace set resampled at 32.78MHz with Hamming Weight as power model and targeting the first round of SBox. Figure 17 shows the graph representation of table 4. As we can see, we recovered about the same number of key bytes with HS probe low pass filtered at 48 MHz as to oppose of using the low sensitivity (LS) probe, but

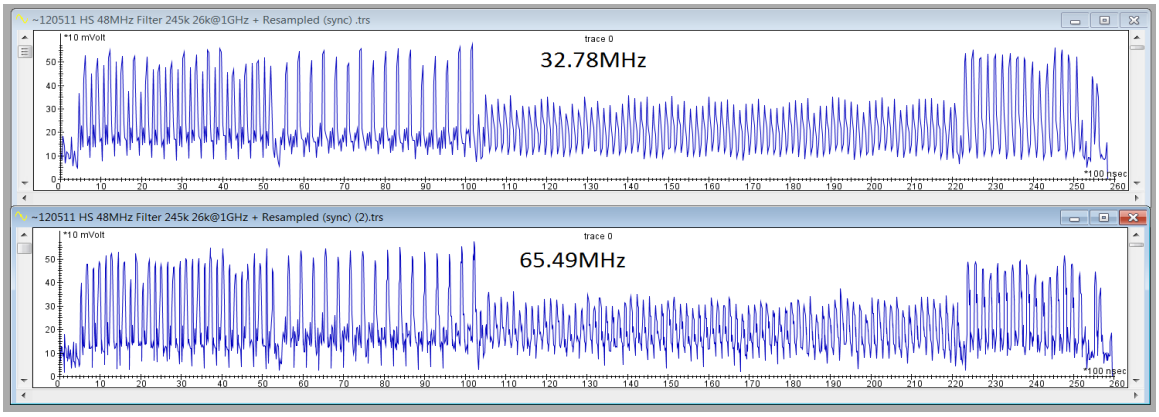

Figure 16: Filter trace set resampled

Power Model/Target	Frequency	Key Recovered	Correct Key Bytes
HW/SBox 1st round	32.78MHz	0049533d555245495343ad4f4c21000b	11
HW/SBox 1st round	65.49MHz	00b0e5f00016db91675b6cb4f63dd00f	0
HD/SBox 1st round	32.78MHz	001dc9170d286a10e02f635c7b85974c	0
HD/SBox 1st round	65.49MHz	00b1cb57f17ee289e09b861f77dca993	0
HD/SBox 10th round	32.78MHz	001dc9170d286a10e02f635c7b85974c	0
HD/Round 10th round	32.78MHz	001dc9170d286a10e02f635c7b85974c	0
HD/Round 1st round	65.49MHz	00b1cb57f17ee289e09b861f77dca993	0

Table 3: 48 MHz low pass Filtered DEMA with Different power models and targets with 100k traces

we were able to achieve such results with only 240k traces as to opposed to 1 million traces. Thus this was a huge improvement over using the LS probe since the number of traces required to achieve similar results with 4 times less traces.

Power Model/Target	Traces	Key Recovered	Correct Key Bytes
HW/SBox 1st round	25000	009e08d8a3a24d3336e1104f3d2119b8	1
HW/SBox 1st round	50000	004953d88b3c6d7288e1104fe021bd0b	2
HW/SBox 1st round	75000	00495353553c458e0043104f94214f4b	7
HW/SBox 1st round	100000	00495373551945495343104f912100b6	9
HW/SBox 1st round	125000	0049530055a245495343144f912100b6	9
HW/SBox 1st round	150000	00495300552745495343104f9121000b	9
HW/SBox 1st round	175000	004953005552454953431 44f0021004b	10
HW/SBox 1st round	200000	00495300555b454953431b4f9121004b	9
HW/SBox 1st round	225000	00495352555245495343 1b4fb621000b	10
HW/SBox 1st round	240000	004953a2555b454953431b4fb621000b	9

Table 4: DEMA on 32.78MHz; Hi Sensitivity Probe; 48 MHz Low Pass filtered

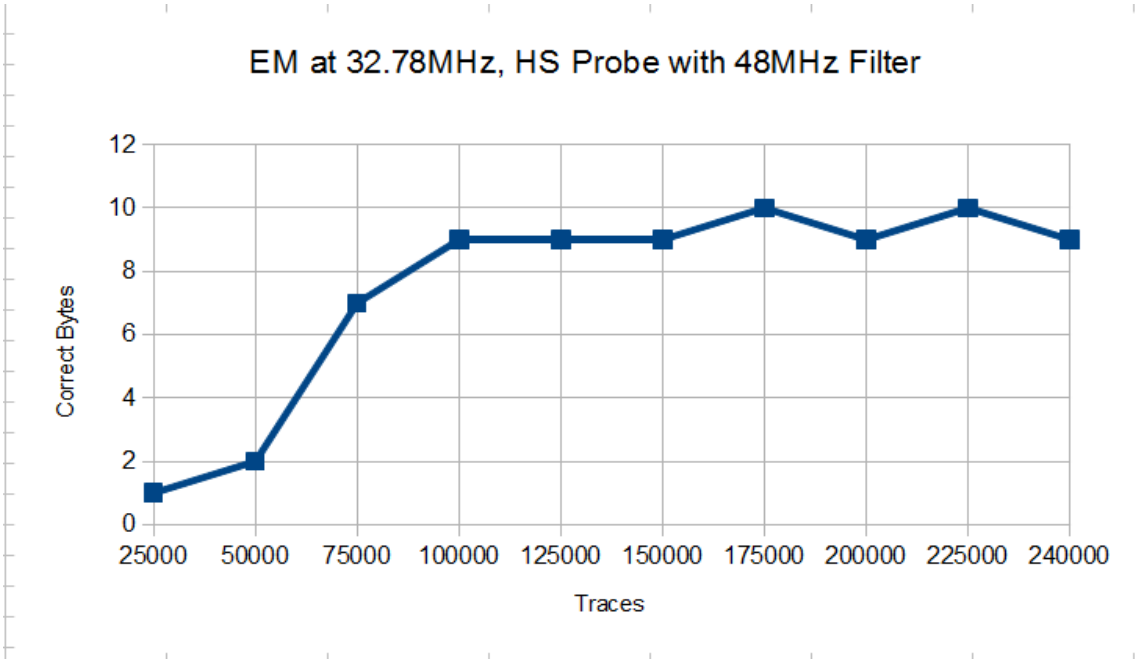

Figure 17: Traces vs Correct Key Byte; DEMA on 32.78MHz; Hi Sensitivity Probe; 48 MHz Low Pass filtered

9.4 DEMA at Location 2

9.4.1 DEMA with High Sensitivity Probe at Location 2

Since taking a large amount of traces shown no improvement, we beginning looking for alternative leakage. Recall that figure 8 displays two different hot spots on the

chip. We initially assumed that the other hot spot (this location will now be referred as location 2 from now on) was simply I/O activities. We revised our assumption and begin taking traces at location 2. The data correlation of these traces shown that there are possible leakage similar to the previous traces. Thus, we took traces at this location with HS probe and no filtering. Figure 18 shows the difference between traces from location 1 and location 2 resampled at 65.43 MHz. Table 5 shows the key bytes recovered at 32.71 MHz, 65.43 MHz, and 130.86 MHz with various trace set sizes, and figure 19 displays this data in graphical form with more data. The number of traces shown in the table and chart are after alignment, and all the unaligned traces has been thrown out. Nearly full key recovery was achieved with just 225k traces for 32.71 MHz resampling, and similar result was achieved with 125k traces for 65.43 MHz resampling. However, full key recovery was not achieved despise taking as much as 4 million traces. Note that the time to run the analysis for 32.71 MHz and 65.43 MHz are similar; and this is due to 32.71 MHz resampling takes twice as many traces as 65.43 MHz resampling, but there are roughly twice as many data in 65.43 MHz resampling to be processed.

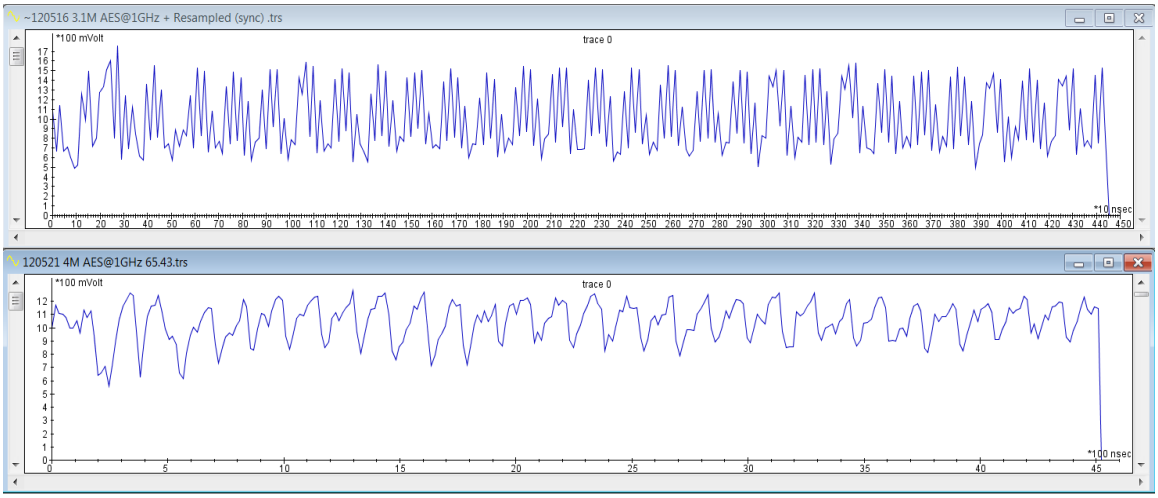

Figure 18: Traces of AES operations resampled at 65.43 MHz; Location 1 on top; Location 2 on bottom

Frequency	Traces	Key Recovered	Correct Key Bytes
32.71MHz	25k	aba673b384d566d067b50214b1f50fcf	0
32.71MHz	100k	004953439e52c449a4434fd44c21310a	10
32.71MHz	500k	00495343555245495343 4F4F4C213100	15
32.71MHz	1M	00495343555245495343 4F4F4C213100	15
32.71MHz	2M	00495343555245495343 4F4F4C213100	15
32.71MHz	3M	00495343555245495343 4F4F4C213100	15
32.71MHz	4M	00495343555245495343 4F4F4C2131f4	14
65.43MHz	25k	35498c29cb9366a7b3b59a2a3368ddcf	1
65.43MHz	100k	004940009e52454953434f4f4c21310a	11
65.43MHz	500k	00495343555245495343 4F4F4C213100	15
65.43MHz	1M	00495343555245495343 4F4F4C213100	15
65.43MHz	2M	00495343555245495343 4F4F4C213100	15
65.43MHz	3M	00495343555245495343 4F4F4C213100	15
65.43MHz	4M	00495343555245495343 4F4F4C2131b4	14
130.86MHz	25k	cbff59f7abf8c4ff3db5abec1a8a19cf	0
130.86MHz	50k	00f8e0447c4312493d5069af6222b60a	1
130.86MHz	75k	00005d5e085264923d741dc6d55b0d97	1
130.86MHz	100k	000053329e5291494b434f004c520d56	6
130.86MHz	150k	00005343a0520249e8434f4f4c21f01b	9

Table 5: DEMA at location 2; Hi Sensitivity Probe; HW/SBox 1st round

9.5 Summary of DEMA on Xmega

Three different options in configurations were introduced in our experiments: LS and HS probes, 48MHz hardware filter, locations 1 and location 2 of the Xmega, and trace set resampling. In all of our experiments, we used Hamming Weight as our power model and targeted the first round of the SubByte operation, and all the other power model and target combinations shown no results (Zero bytes of the key recovered). In sections 9.3.1 and 9.4.1, we obtained trace sets from two different locations based on the spectrum shown in figure 8. The best result we obtained from location 1 is recovering 12 bytes of the key. On the other hands, we were able to

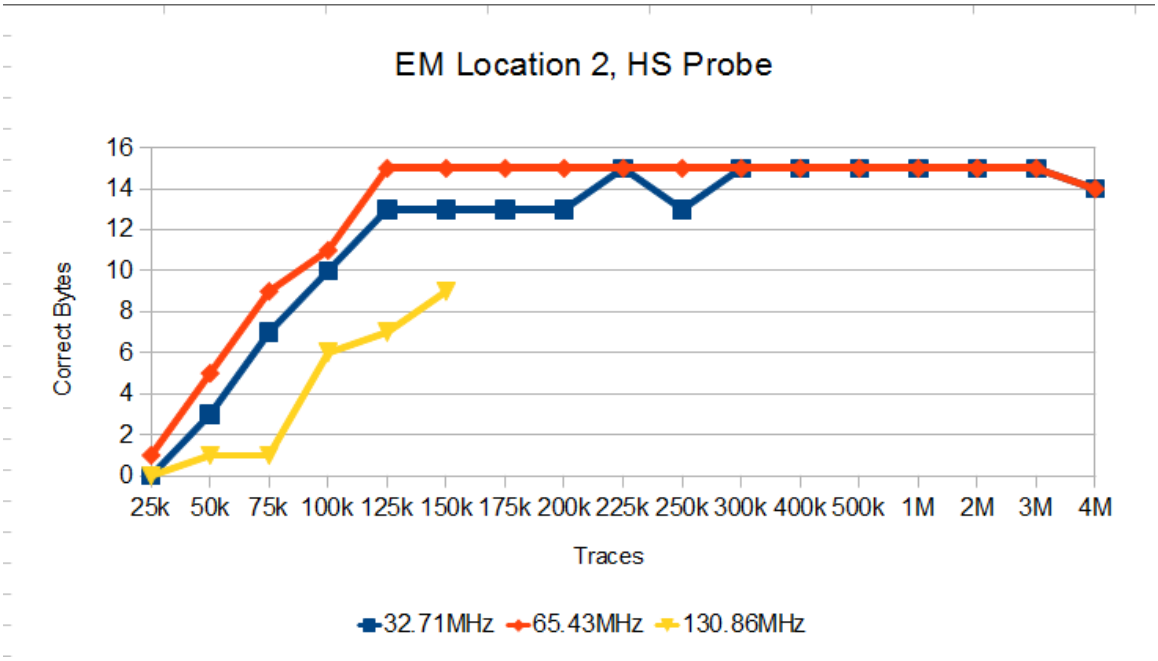

Figure 19: Traces vs Correct Key Byte; DEMA on Location 2; Hi Sensitivity Probe

recover 15 bytes of the key with location 2. Based on these results, we can conclude that there are more than one location where the leakage can occur on the Xmega, and the most leakage occurs the most at location 2 where we believed where the input/output operations of the Xmega is located. In section 9.3.3, we attempted to perform DEMA by introducing a 48MHz hardware filter into our set up to eliminate high frequencies noises. With the hardware filter, we were able to recover 11 bytes of the key with only 100k traces, while 12 bytes of the key were recovered with the unfiltered set up as shown in section 9.3.1. Both of the unfiltered and filtered experiment were done at location 1. This indicated that the hardware filter does eliminate the high frequencies noise picked up by the EM probe. In addition, we also tested the uses of LS probe and the uses of HS probe. In section 9.3.2, we performed DEMA with the HS probe, and we were able to recover 12 bytes of the key with 1.5M traces. However, we cannot draw conclusion on which probe shown better results

since the experiment done in section 9.3.1 was with 1M traces. Further research will need to be done in order draw conclusion regarding the use of LS probe and the use of HS probe. Finally, we resampled all of our trace sets to 32.71MHz (clock frequency of the Xmega) and 65.43MHz (upper harmonic) in all of our experiments. In section 9.4.1, the trace set resampled to 32.71MHz was able to recover 15 bytes of the with 225k traces; the trace set resampled at 65.43MHz was able to recover the same key with 125k traces. While the trace set resampled 65.43MHz takes less traces, it also takes twice as much data as the trace set resampled at 32.71MHz. Thus, the amount of resource (data storage and processing time) is doubled for the same key recovery for the trace set resampled at 65.43MHz

In summary, we were not able to achieve full key recovery with DEMA, but we were able to recover up to 15 bytes of the key. The location of the probe is one of the most important factor in determining if we can achieve full key recovery. For the experiments with the EM probe, we placed the EM probe in two different locations. The first location that we placed the EM probe is located at the center of the chip, and we believed location 1 is where the cryptographic engine is located on the chip. However, we were only able to recover up to 12 bytes of the key with 1.5 million traces resampled to 65.43MHz with our best attempt of DEMA at this location, and increasing the number of traces did not improve the number of bytes of the key recovered. The next location that we placed the probe is near the edge of the chip, and we believed location 2 to be the input/output lines between the chip and the serial interface. We were able to recover up to 15 bytes of the key with traces taken from this location, and 225k traces resampled to 32.71MHz or 125k traces resampled to 65.43MHz were needed to achieve the 15 bytes recovery. The calculation times between the two resampled trace sets are roughly the same due to the number of

samples per trace and the number of trace in the trace set. The best configuration we found from our experiment is to used the HS probe, unfiltered, at location 2, and perform DEMA on the trace set resampled at either 32.71MHz or 65.43MHz depending on the computational resource one might have. In the next section, we will begin performing DPA on the Xmega and discuss the results.

CHAPTER 10

DPA on ATXmega256A3B

As mentioned in the previous sections, DPA is a side channel attack technique that uses the power measurements of the system of interests while it is performing cryptographic operations. For this project, we will be taking measurements in two ways, and the two ways are across the resistor on the ground end and the current on the VCC end of the chip. We will first present the result of analyzing the traces from measuring across the resistor, and we will also present the result of analyzing the traces from measuring the current.

10.1 DPA: Measuring across the resistor

The set up for measuring across the resistor is very similar to that of measuring the EM leakage for Xmega. Once again, we will have a work station running Inspector, the LeCroy oscilloscope, and the Xmega. The Xmega is connected to the work station for transferring plain text and cipher text between the two by the means of USB/serial converter. The trigger is done exactly the way as described in the DEMA section. Figure 20 is a diagram of the set up described above. The major differences are how the chip is powered and how the oscilloscope is taking measurements. The chip is no longer being powered by the work station, but it is powered by 3V DC power source. For taking measurements, we put a resistor between the ground of the power source and the Xmega. The strength of the resistor will affect the quality of the traces; a resistor with higher resistance will increase the quality of the traces. The probe will be measuring the drop in voltage across the resistor as the Xmega is performing the cryptographic operations. According to Ohm's law, $V = I \times R$, where V is voltage,

I is current, and R is resistance. Since the resistance is constant in this set up, this means the change in voltage is linearly related to the current. In addition, we noticed that the ground end of the trigger probe is drawing more current than the ground end of the power source. This causes an incomplete circuit for the Xmega and rendered it inoperable, so we put a 10M ohm resistor on the ground end of the trigger probe in order to make the Xmega operatable again.

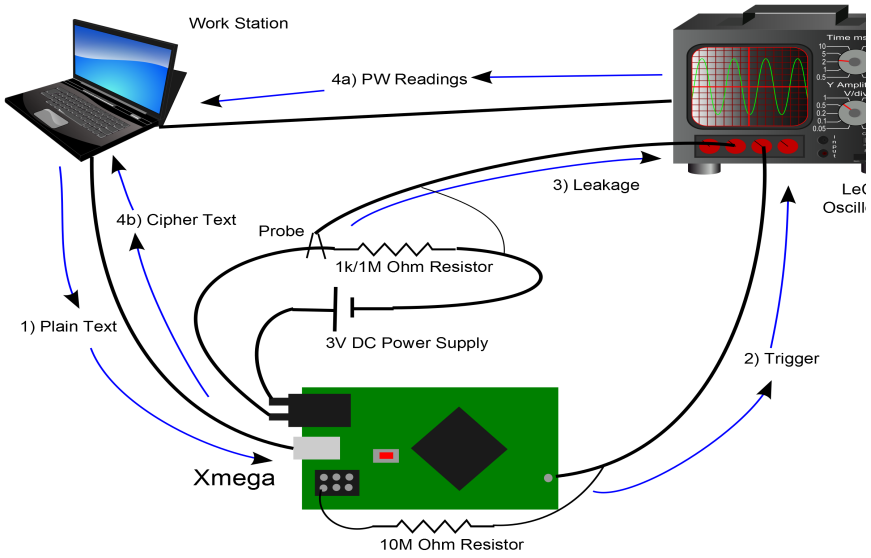

Figure 20: Set up diagram for DPA with resistor

During the acquisition of the traces, we also noticed that the power signal is very weak compared to the EM signal. The EM signal ranges from 2V to 3V, and the power signal ranges from 50mV to 100mV. As such, we increased the strength of the power signal by adding a 12V amplifier between the probe and oscilloscope. Figure 21 shows traces with and without the amplifier. Furthermore, we observed that there are periodic power spike in our traces as demonstrated in the bottom trace of figure 21. This power spike occurred due to the changes in voltage of the other active components of the chip such as LEDs and USB controller. With this additional

power spike, we need to acquire more traces and throw out the ones with power spikes occurring in the region that we are interested during our trace alignment phase.

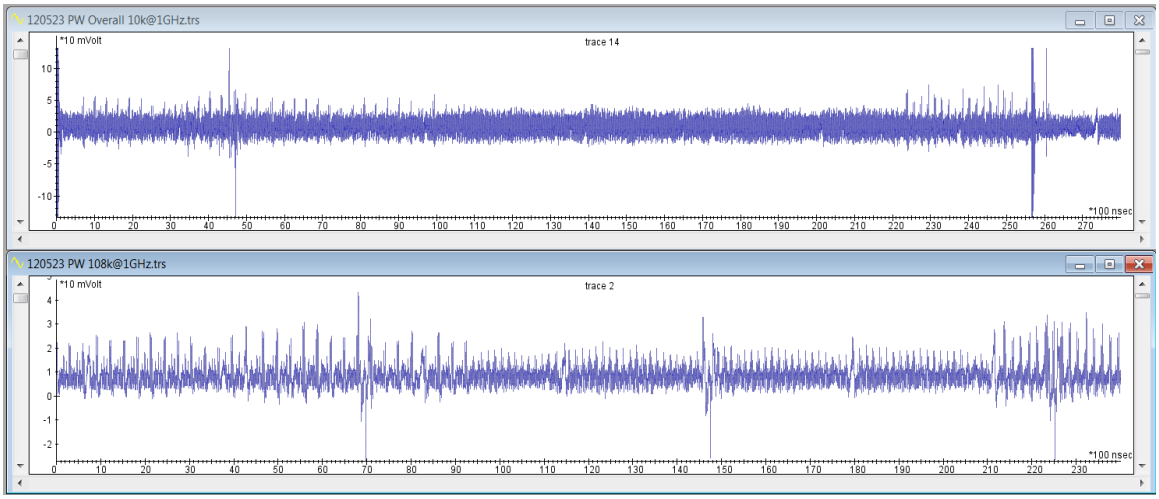

Figure 21: Power Traces of overall operations sampled at 1 GHz; With amplifier on top; Without amplifier on bottom

Since using more than 4 million traces with no notable result means that this set up will not as efficient as the set up for DEMA, we have not take more than 4 million traces with this set up. Due to time constrain, we only capture 1.5M aligned traces by measuring with the 1k ohm resistor and capture 2.5M aligned traces by measuring with the 1M ohm resistor. Figure 22 shows sample traces using the set up with the resistor, and the two traces show power signal sampled at 1 GHz and resampled at 32.96 MHz.

During the analysis phase, we resampled the traces down to 32.96MHz and 65.67MHz as suggested by running the Spectrum module on the trace set. However, the trace sets resampled down to these two frequencies yield no notable results up to 2.5 million traces. That means zero byte of the key was recovered from the resampled trace set. On the other hands, we were able to get significant results by

analyzing the raw traces sampled at 1 GHz. Table 6 shows the results by analyzing the traces. Noted that we were able to achieve a full key recovery with 2.5 million traces by measuring across an 1M ohm resistor sampled at 1 GHz. The time it takes to acquire 2.5 million traces is roughly two days, and the time for analysis is about 1 days. Thus it will take about a total of 3 days in order to achieve full key recovery. Figure 23 shows the number of traces compared against the number of key bytes recovered. Based on the figure, we can see that the 1M ohm resistor shown better results than the 1k ohm resistor, and the 1M ohm resistor was able to recover twice as many bytes of the key as the 1k ohm resistor between 400k and 1M traces. However, the 1k ohm resistor recovered 14 bytes of the key with 1M traces, which is one byte short of the trace set with 1M ohm resistor with the same number of traces. Below 1M traces, the 1M ohm resistor was able to show better results, but we cannot draw conclusion on which resistor shows better results. We can further research this question by taking more traces with the 1k resistor until we achieve full key recovery.

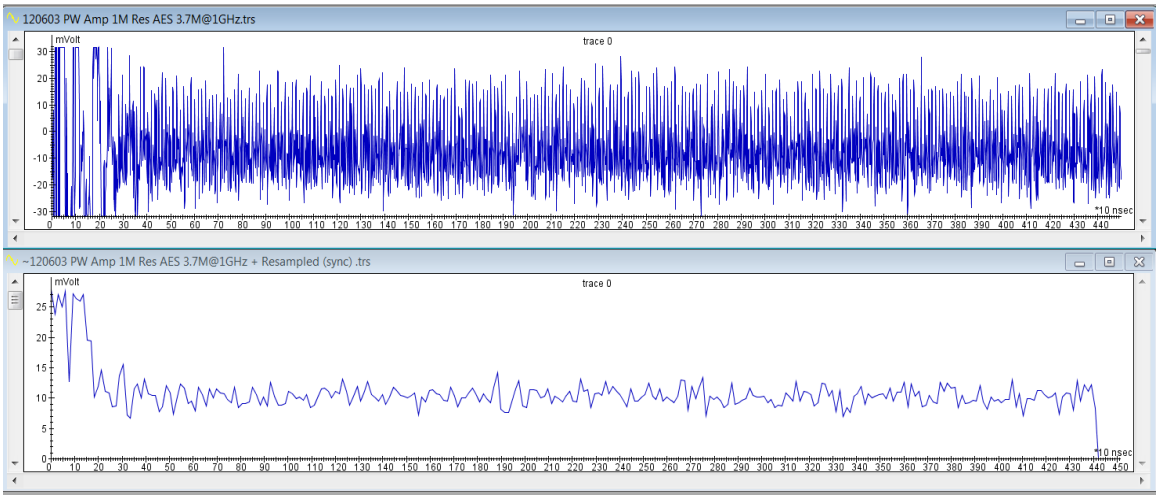

Figure 22: Power Traces of AES operation over 1M ohm resistor; Sampled at 1 GHz on top; Resampled to 32.96 MHz on bottom

Resistor	Traces	Key Recovered	Correct Key Bytes
1k ohm	25k	070eaa3011604f9556c9fbd4a48a3664	0
1k ohm	250k	33728a43554d5ad60d3d1d14676dba9b	2
1k ohm	500k	5272fd4e71771ef0539f2043660b4df2	2
1k ohm	1M	5249e443556bb04953430eb78a0b0b00	8
1k ohm	1.5M	5249534355c3454953434F434C213100	14
1M ohm	25k	d4e6257cafc99014d0a600942f87646f	0
1M ohm	250k	302a851659a774e336b58b6717cd7595	0
1M ohm	500k	5249704a55e2ae4932434f624ce55000	8
1M ohm	1M	52492643555245495343 4fb04c213200	13
1M ohm	1.5M	52495343555242495343 4f924c213100	15
1M ohm	2M	52495343555245495343 4fb04c213100	15
1M ohm	2.5M	52495343555245495343 4f4f4c213100	16

Table 6: DPA with resistors; HW/SBox 1st round; Sampled at 1 GHz

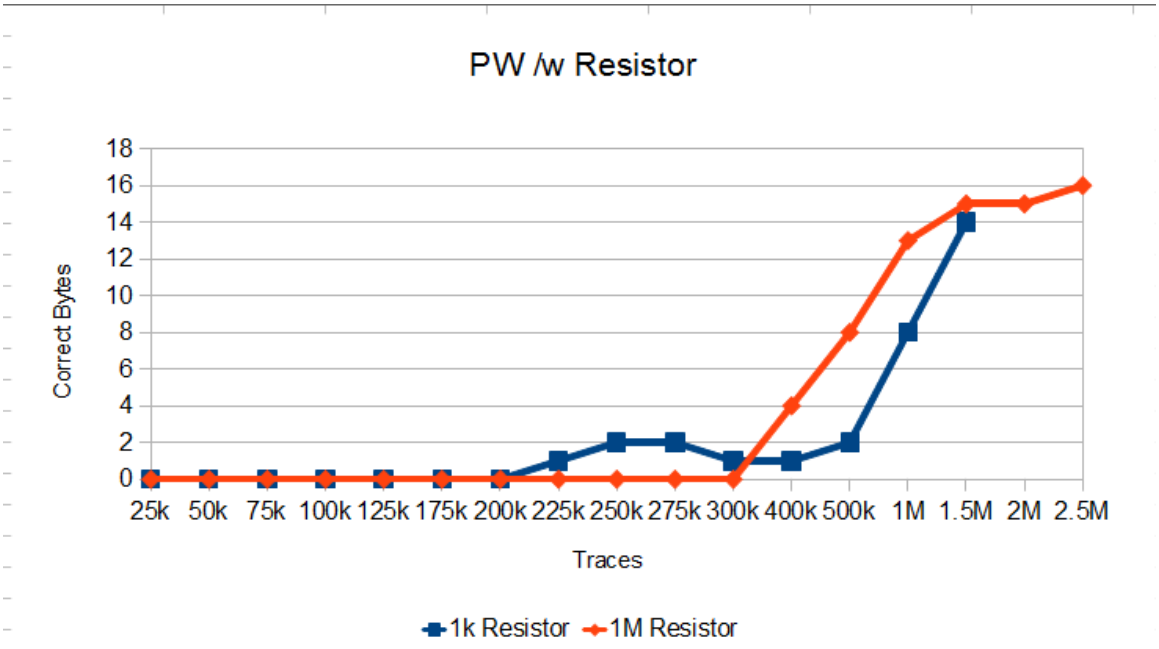

Figure 23: Traces vs Correct Key Byte; DPA with Resistor

10.2 DPA: Measuring the current

In this section, we will describe our set up for measuring the current of the Xmega while it is performing cryptographic operation, then we will discuss the results of analyzing the traces obtained with this set up. Note that this set up is the same as

the previous side channel analysis effort [30]. As expected, the result of this set up closely resemble that of the previous work.

The overall set up for measuring current is quite similar to that of the set up for measuring across the resistor. The major differences are once again the power source and the probe that we are using. Instead of using a 3V DC power supply, we used two AA batteries in series to provide a stable 3V power supply for the Xmega. The current probe is attached onto the VCC end of the batteries instead of the ground end. This was suggested by a colleague, and we would to repeat the experiments done in sections 9.3.1, 9.4.1, and 10 with the battery as power source in the near future as further research. Finally, we power the chip via the programming pin rather than from the power inlet provided on the chip. The only difference between our set up and the set up from the previous work [30] is that we did not remove the onboard LED(s). The diagram for this set up is shown on figure 24.

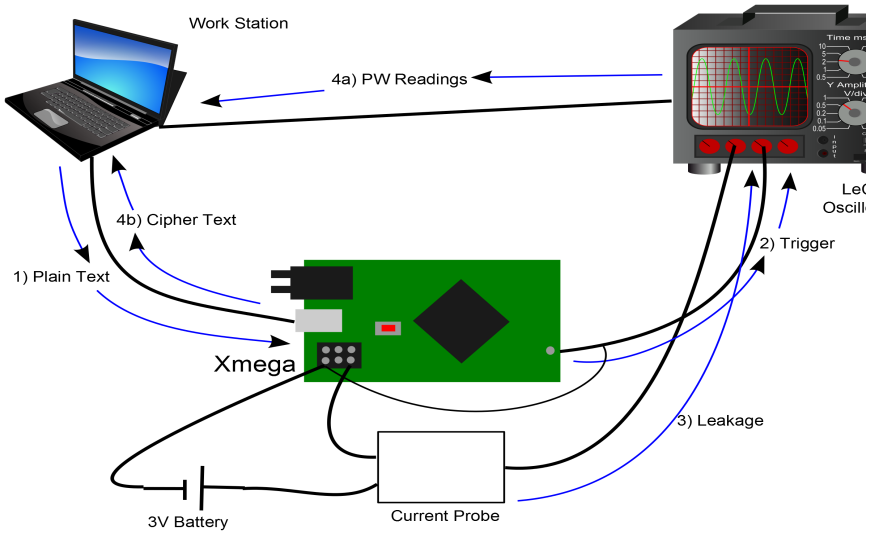

Figure 24: Set up diagram for DPA with current

Similar to measuring across the resistor, the signals for measuring the current are also very weak. The signals are in the -/+ 10 mV range. In addition, the periodic power spikes are once again present in the traces, and the process of throwing out unusable traces is once again employed with the use of the Static Alignment module (details in section 9.3.1). Figure 25 shows resulting traces with this set up. The top trace shows trace with the periodic power spike, and the bottom trace is power spike free. If we examine the traces, we can once again identify the different stages of the AES operation in the traces. The 10 ms to 21 ms range is where the AES encryptions occur, and we verified this findings by once again running the data correlation module. The major difference from the traces of this set up and from the traces of the previous set up is that a very low frequency signal, at 1MHz, is presented in the traces. We applied a software, as part of Inspector, band pass filter ranged from 1.5 MHz to 5 GHz in order to get rid of this low frequency signal while preserving the original signals. Figure 26 shows traces of the first half of the AES encryption operations. A band pass filtered is applied to these traces to remove the low frequency signal. The bottom trace is the original trace sampled at 1 GHz, and the top and middle traces are resampled to 32.71MHz and 65.43MHz from the original trace.

In the previous side channel attack effort of the Xmega, only 30k traces were needed in order to achieve full key recovery [30]. In our attempts, we were able to produce similar results. We achieve full key recovery with 45k traces resampled to 32.71 MHz. In addition, 100k traces resampled to 65.43 MHz were needed to achieve full key recovery. Note that for traces resampled at 65.43 MHz, 15 bytes of the key were recovered from analysis attempts between 60k traces and 100k traces, and the difference of correlation values for the missing byte between the real key and the key guess was no more than 0.0002. That means there is a high probability that we should

be able to achieve full key recovery with about 60k traces resampled at 65.43 MHz. We will discuss the success rate against number of traces in a later section. Finally, the analysis on the original traces was able to recover up to 15 bytes of the key with 40k traces. However, the execution time, clocked at 24 hours, of analysis of the 1 GHz traces is much higher than the analysis of the 32.71 MHz and 65.43 MHz traces. Table 7 and figure 27 show the full results discussed in this paragraph. Overall, we reproduce similar results to that of the previous side channel attack on the Xmega with only 20k more trace [30]. While we need 2.5M traces for the trace set measuring across the resistor to perform full key recovery, we only need 50k traces in order to perform full key recovery by measuring with the current probe. This result shows us that the leakage from change in current is stronger than that of measuring the change in voltage.

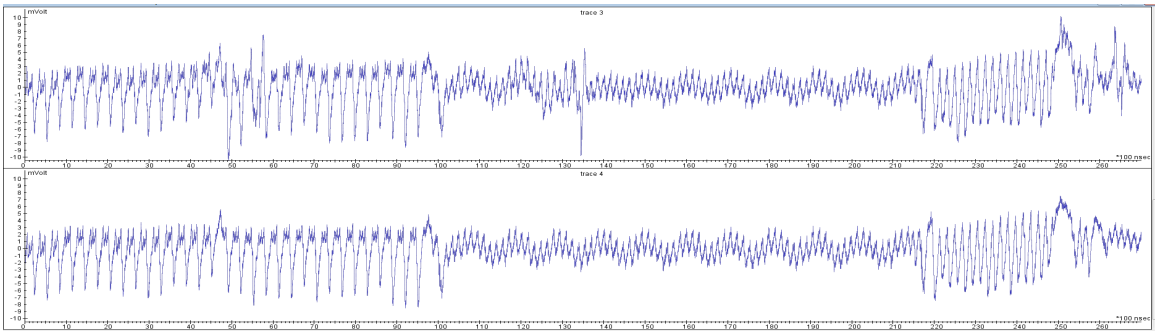

Figure 25: Power traces with current; With random spike on top; Without random spike on bottom

10.3 DPA: Success rate

In this section, we will determine the side channel attack's success rate by measuring across the resistor. There is a module in Inspector, called First Order Stats, that will

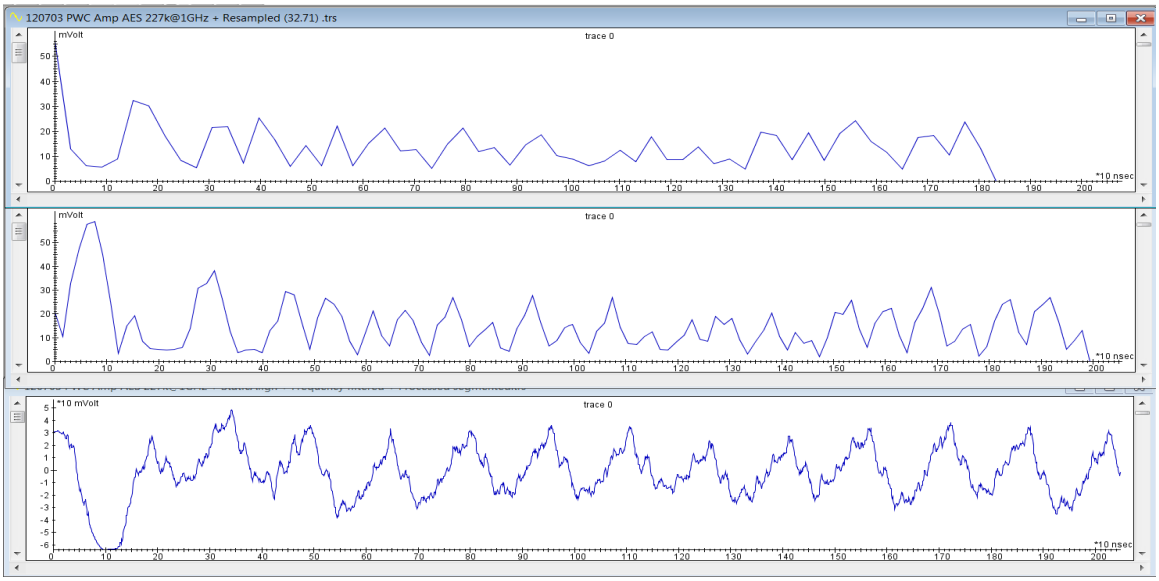

Figure 26: Power traces with current on 32.71 MHz, 65.43 MHz, and 1 GHz

Frequency	Traces	Key Recovered	Correct Key Bytes
32.71MHz	25k	5214d9bf6e52454953434f4f2d21318f	10
32.71MHz	30k	524953b4555245491e434f4f3c213100	13
32.71MHz	40k	5256534355524549534c4f4f4c213100	14
32.71MHz	45k	52495343555245495343 4f4f4c213100	16
32.71MHz	50k	52495343555245495343 4f4f4c213100	16
65.43MHz	25k	52cdd9535552d04953434f4f9e210800	10
65.43MHz	30k	527453535552568b53434f4f4c213100	12
65.43MHz	50k	52eb534355525649534 34f4f4c213100	14
65.43MHz	60k	52495343555256495343 4f4f4c213100	15
65.43MHz	75k	52495343555256495343 4f4f4c213100	15
65.43MHz	100k	52495343555245495343 4f4f4c213100	16
1GHz	25k	52c453935552dc4953434f4f4c213100	13
1GHz	30k	52e553975552dc4953434f4f4c213100	13
1GHz	40k	527b534355524549534 34f4f4c213100	15
1GHz	50k	527b534355520e4953434f4f4c213100	14
1GHz	60k	527b53435552d44953434f4f4c213100	14

Table 7: DPA with current probe; HW/SBox 1st round

perform the success rate calculation provided that we have a large amount of traces. The First Order Stats module will split the trace set into k smaller and equal in size sub trace sets. For each of these k sub trace sets, the module will perform DPA on

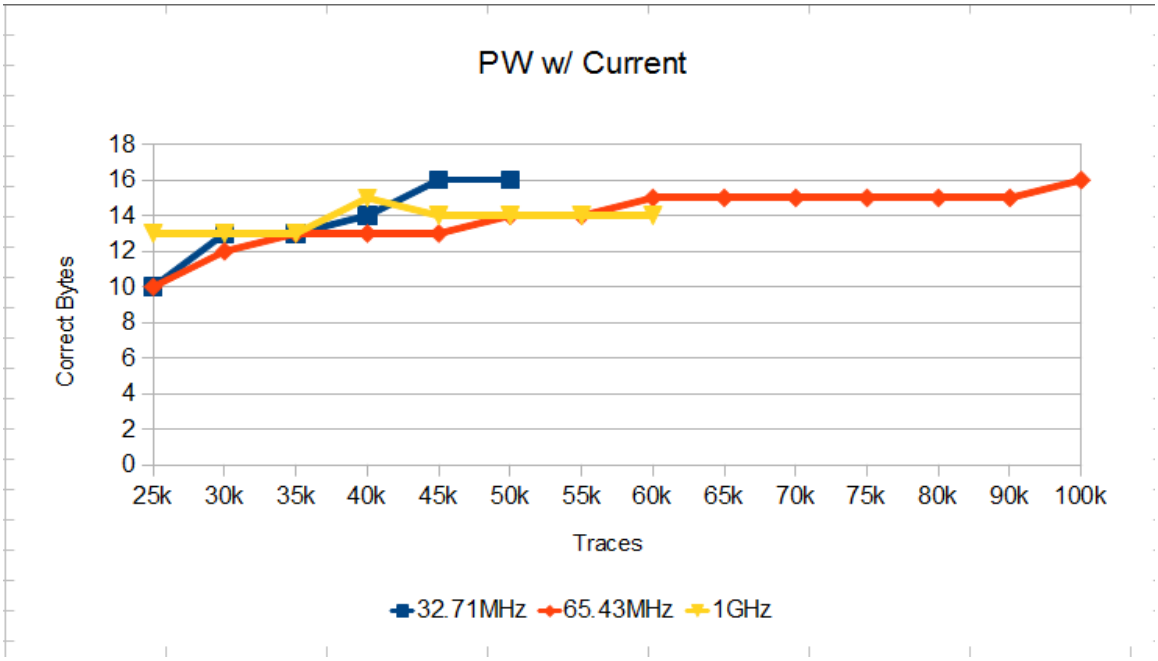

Figure 27: Traces vs Correct Key Byte; DPA with Current

a small number, i, of traces and increase i by some constant until the all traces in the sub trace sets have been used. The module will then report the number of traces needed to achieve full key recovery in each k sub trace sets for each i number of traces. We can then determine the side channel attack's success rate based on these results.

For our experiments, we will be measuring the success rate of performing DPA with current. We were able to obtain 5.2 million traces for this experiment. We will only perform the experiment on trace sets resampled to 32.71 MHz and 65.43 MHz since these are the frequencies that we were able to achieve full key recovery. Based on the results from the previous section, we know that the highest number of traces that we need to achieve full key recovery is 100k traces, so we will use this number as our limit. Hence, we have 52 sub trace sets with 100k traces each. For each of these sub trace sets, we will perform DPA at every 1000 traces (e.g. DPA on 1k traces, 2k traces, ... 100k traces). Table 28 shows the results of this experiment. Note that

both frequencies produced similar graphs. Based on these results, we can see that the minimum amount of traces needed to achieve full key recovery with some probability of success is 30k traces, and this is consistence with the previous side channel attack effort [30]. On the other hands, both frequencies show that performing DPA with 80k traces will guarantee a full key recovery. However, performing DPA on traces sampled at a lower frequency will yield faster execution time. Thus, performing DPA on trace sets resampled to 32.71 MHz execute twice as fast as trace sets resampled at 65.43 MHz. Hence, resampling traces to the operating frequency of the Xmega is suggested for DPA in terms of overall performance.

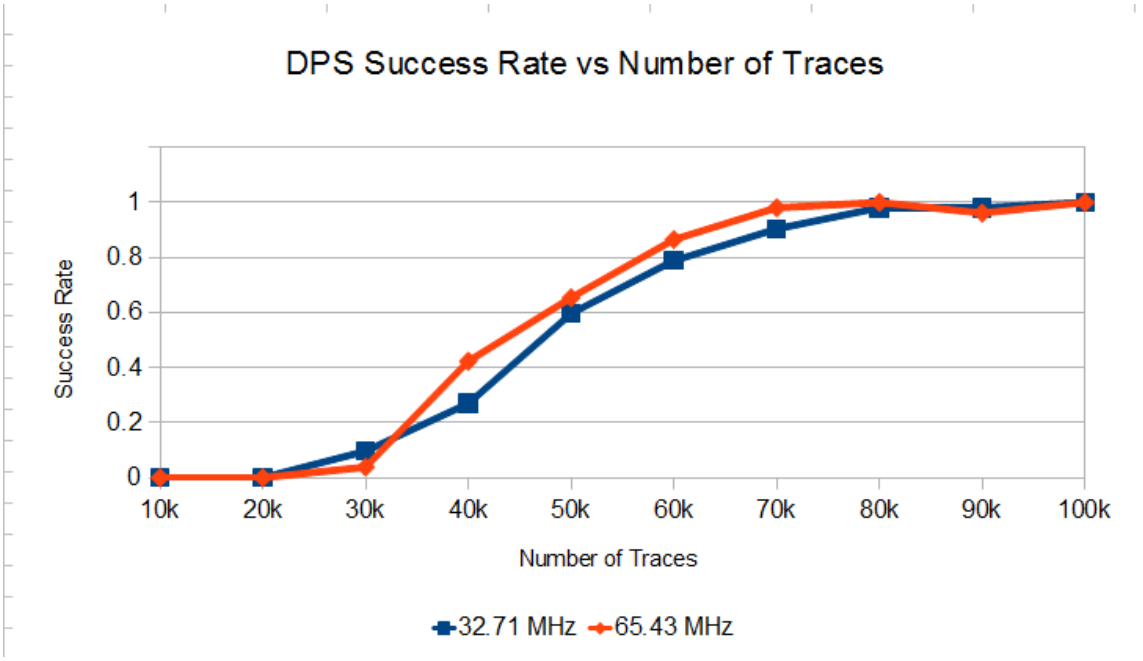

Figure 28: Success rate vs Number of traces at 32.71 MHz and 65.43 MHz

10.4 Summary of DPA on Xmega

In summary, we achieve full key recovery in multiple occasions. We performed DPA in two different ways. The first way is to measure the change in voltage by placing a resistor on the ground wire of the target, and the second way is placing a current

probe on the VCC wire of the target. Full key recovery was achieved using both ways. By measuring the voltage across the resistor, we achieve full key recovery with 2.5 million unfiltered traces. On the other hands, we achieve full key recovery using only 45k resampled traces with the current probe. In addition, the First Order Stats module shows us that only 80k traces are needed for guaranteed full key recovery. In the case of measuring across the resistors, we cannot conclude if increasing the resistance of the resistors will decrease the number of traces needed to perform full key recovery. Further experiments is required to make such claim. However, since we perform full key recovery from measuring both the change in voltage and change in current, we can conclude that measuring the change in current yield better results than measuring the change in voltage. The difference in the amount of traces needed to perform full key recovery between the two methods is 2 millions, and the total time (acquisitions and analysis) for performing DPA by measuring across the resistor is 3 days whereas the total time for performing DPA by measuring the current probe is an hour. In the next section, we will discuss the experiments and results with using a radio receiver as a downmixer and perform DPA and DEMA on the demodulated signals.

CHAPTER 11

Downshifting with Icom R7000

The Icom R7000 is a radio frequency radio receiver capable of capturing a wide range of signals. In this section, we will be using the Icom R7000 (or simply referred as Icom from now on) for downmixing. Downmixing is the process of applying a bandpass filter on the received signals down to a base band. The Icom can receive radio signal and downmix the signal to produce a single channel audio signal. The Icom can receive signals from 25MHz to 999MHz and from 1025MHz to 2000MHz in AM, AM-W, FM, FM-W, FM-N, USB, and LSB modes [29]. We chose the Icom because it can receive a wide range of frequencies where as modern radio receivers have restricted range of frequencies that they can receive. We like to expand the use of the Icom outside of the scope of this project in the near future. In this section, we will describe the set up for acquisitions with the Icom, and the results of performing DPA and DEMA on the Xmega with the Icom as a demodulator.

11.1 Set up with the Icom

We will begin by discussing about the set up for DEMA on the Xmega with the Icom. The set up with the Icom is exactly the same as the set up described in the DEMA section. The Icom is equipped with an N-type connector on the back, and this connector is used for attaching an antenna to the Icom for receiving radio signals. For this experiment, we connected the high sensitivity EM probe to the N-type connector port using a modified coax cable. There is an intermediate frequency (IF for short) port used for other audio functions of the Icom. We connect the LeCroy to the Icom using the IF port. The IF port will always produce a 10.7MHz out put signal

regardless of the frequency of the radio receiver is tuned to listen on. Finally, we put a 50 Ohms impedance matcher between the LeCroy and the Icom to prevent the Icom from overloading the LeCroy with currents. Figure 29 shows a diagram of the set up.

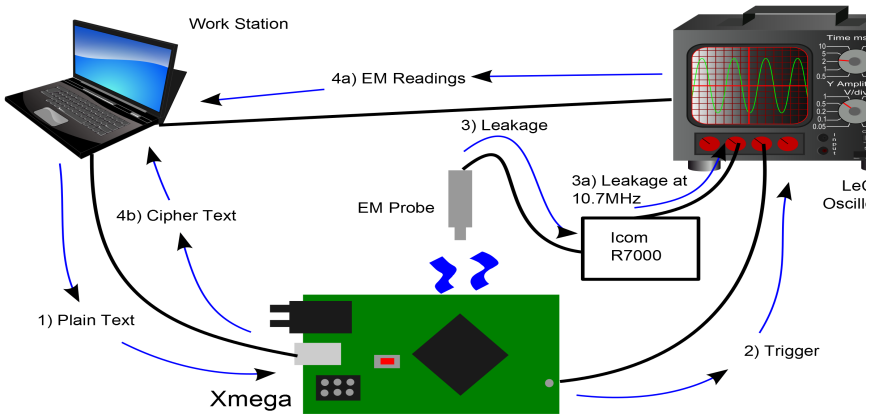

Figure 29: Set up diagram for DEMA with the Icom R7000

As for DPA, we will only perform experiment with the Icom using only the current probe since the current probe produced a much better result than measuring across the resistor. Once again, the set up for DPA with the Icom is almost exactly the same as the set up for DPA as described in the DPA section. The Icom is once again connected to the current probe on the antenna port and connected to the LeCroy on the IF port. Refer to number in figure 29 regarding where the Icom is located in the set up.

As mentioned earlier, we can tune the Icom to listen to any frequencies that we set as long as it is with in specifications, and the out put on the IF port will always be 10.7MHz. If we tune the Icom to listen to the operating frequency of the target and capture the out put signal on the IF port, we will be downshifting the operating frequency down to 10.7MHz. The point of this experiment is to see if we can downshift the operating frequency while successfully perform DPA or DEMA on the target. If

we can manage to perform DPA/DEMA at 10.7MHz, then it means we can perform DPA/DEMA more efficiently since performing DPA/DEMA at 10.7MHz takes less calculations than traces taken at the operating frequency; this is due to the clock frequency of the Xmega is 32MHz, and there are less samples per trace to perform DPA/DEMA on per trace in the trace set (re)sampled at 10.7MHz than trace sets (re)sampled at 32MHz. In our experiments, we will be downshifting the 32.71MHz of the Xmega to 10.7MHz with the Icom.

Before we discuss the results of DPA and DEMA with the Icom, we will point out a few observations about taking traces with the Icom. The first observation is that the out put signal of the IF is very weak, and it is in the -/+10mV range. We strengthened the signal by putting a 12V amplifier between the Icom and the LeCroy, and the signal with the amplifier is in the -/+50mV range. The second observation is that the Icom can, and occasionally will, pick up other radio signal then the ones we intended since the Icom is a radio receiver. The Icom can pick up signals from cell phone, EM radiations from other electronic equipments nearby, or etc. Figure 30 shows a sample trace with signals not coming from the current probe. This interference means more traces are needed in order to perform DPA/DEMA on the target, and the traces with this interference will have to be thrown out during the alignment phase of the analysis. Finally, the spectrum of the traces (shown in figure 31) taken from the IF port revealed three dominating frequencies. The three frequencies are 4.15MHz, 10.742MHz, and 17.334MHz. These frequencies are independent of the target, and they are produced by the Icom by downmixing the received signal down to these frequencies. This information will become useful when we begin analyzing the traces from measuring the IF port of the Icom.

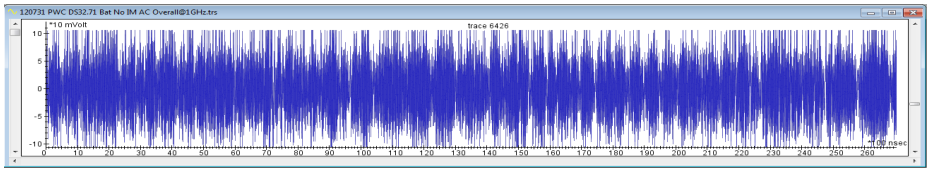

Figure 30: Sample trace of interference with Icom

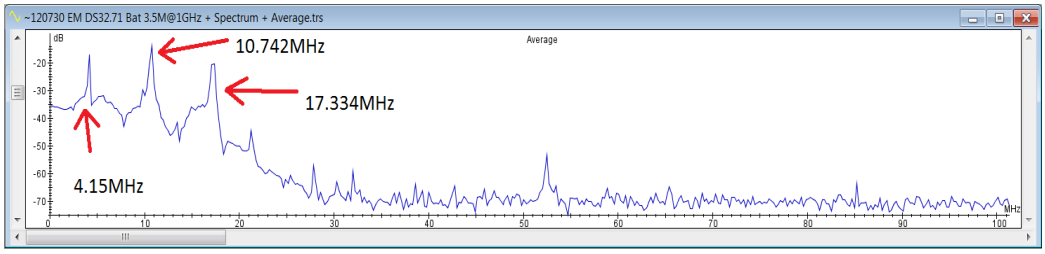

Figure 31: Spectrum of traces taken with the Icom

11.2 DPA with the Icom

In this section, we will be discussing the results of performing DPA on traces taken with the Icom. Recall that we will only be taking traces with the current probes since only 80k traces are needed in order to perform a full key recovery. Thus, if we can perform full key recovery with less than 80k traces, then using the Icom will definitely be an improvement. In this section, we like to see if downshifting with the Icom can be an improvement over traditional method of performing DPA on our target.

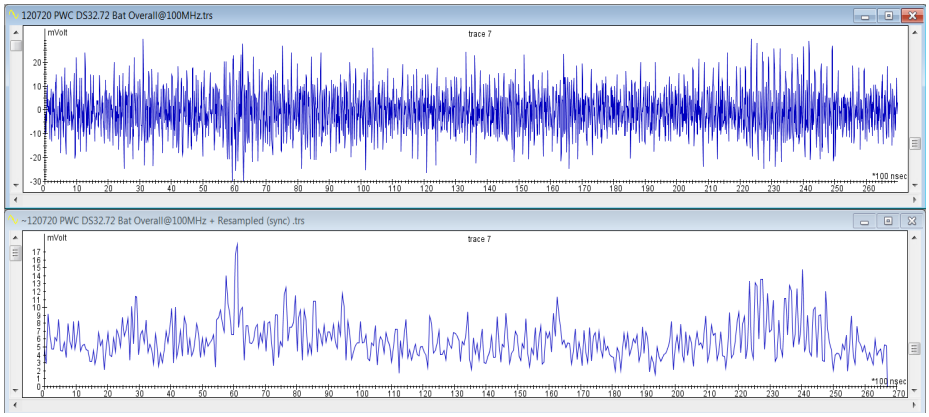

Figure 32: Sample traces of all AES operations with current probe at 100MHz; unfiltered on top; 17.334MHz resampling on bottom

We will begin by examining the traces with the over all AES operations. Figure 32 is a sample trace of the over all AES operation with the current probe. The trace on top is sampled at 100MHz and unfiltered, and the trace on the bottom is resampled to 17.334MHz as suggested by the Spectrum module. The Icom was set to receive at 32.71MHz during acquisition. The actual AES encryption operations are between 10 ms and 21 ms. We were able to perform the input and out put data correlations on the resampled trace set. However, no correlations were found between the input/out put data and the unfiltered trace set. This suggests that there are a lot of noises in the unfiltered traces and we should resample the trace sets down to one of three frequencies listed earlier.

We resampled the trace sets down to the three frequencies, 4.15MHz, 10.742MHz, and 17.334MHz, as suggested by the Spectrum module. These frequencies are independent of the target, and they are produced by the Icom by downmixing the received signal down to these frequencies. However, we were not able to recover any byte of the key with 100k traces. The next step we took in this experiment is to figure out the exact frequency in which is the leakage is happening. We reexamined the spectrum of our old trace sets taken with the current probe, and we found the following frequencies are also leaking key information: 13.184MHz, 19.531MHz, 23.926MHz, 26.123MHz, 39.06MHz, and 45.65MHz. We found these frequencies by running the Spectrum module of Inspector on traces taken in section 9.3.1 and locate the peak frequencies. We were able to perform full key recovery with only 80k traces by resampling our old trace sets to these frequencies. Once again, we were not able to recover any bytes of the key by taking traces from the IF port while tuning the Icom to these frequencies.

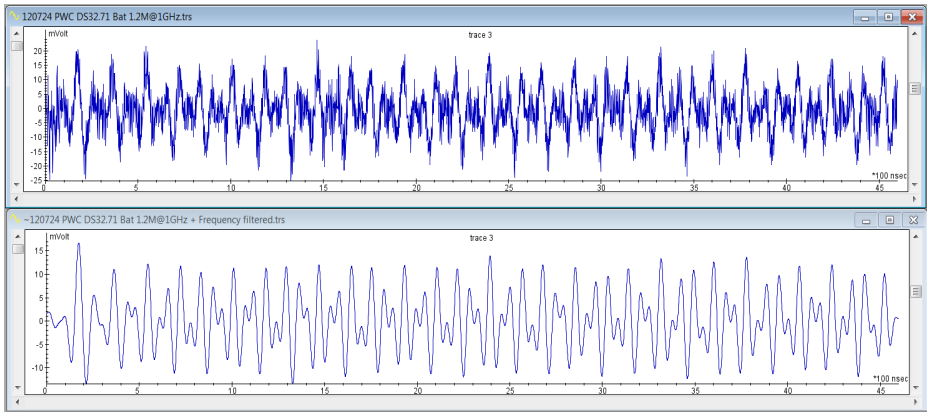

Figure 33: Sample traces with current probe and Icom; unfiltered trace on top; 5MHz to 25MHz band pass filtered on bottom

Since we were not able to recover any byte of the key by simply resampling the trace sets, band pass filtering is our next option. Band pass filtering allows us to extract samples at a certain range of frequencies from a trace. In our experiments, we applied a band pass filter from 5MHz to 25MHz to our trace set, and the purpose of applying the band pass filter is to eliminate the higher frequencies noises while preserving the 10.742MHz and 17.334MHz signals. Figure 33 shows a trace before and after applying the band pass filter. In this figure, the trace on top exhibits high frequencies noises in between each low frequency peaks, and the trace at the bottom are free of all the high frequency noises and contains only signals between the frequencies of 5MHz and 25MHz. Before we attempt to perform DPA on the filtered trace set, we ran a diagnostic tool called KnownKeyCorrelation. This tool allows us to see the correlation value between the trace set and the known key. The tool will take the known key and calculate the correlation value at each part of the trace with different power models (e.g. Hamming weight and Hamming Distance) and targets (e.g. 1st round of SubByte and 10th round of AddRoundKey). The user can compare the correlation values of all the different power models and targets combinations and determine which combination and the location in the trace are

best suited for performing DPA/DEMA. Note that is tool can only be used when the key is known by the attacker, and it is useful for attackers who have access to a copy of the system of interest before attempting to attack the actual target. For AES, the tool will test for Hamming Distance as well as Hamming Weight for both Sbox in and Sbox out. Refer to section 4.1 about the different power models, targets, and correlation calculations. Figure 34 shows the results of running the KnownKeyCorrelation module for 1.6 million traces. Each of the trace set shown in the figure is overlapped with 16 traces, and each of these traces corresponds to a byte of the key. The figure shows that there is a small correlation between the known key and the Hamming Weight of the first round of SBox out between 0.9ms and 1.3ms. Furthermore, the figure also shows a small correlation for the second round between 2.0ms and 2.7ms and a small correlation for the third round between 2.7ms and 4.0ms. The results show us a weak correlation between the known key and the trace set. Nonetheless, there are still correlation between the key and the trace set, so it is entirely possible to recover the key by performing DPA on the trace set.

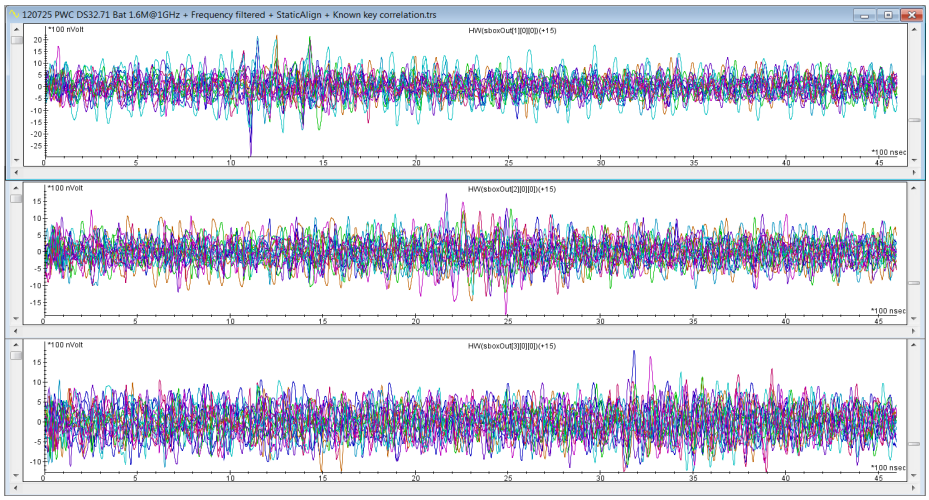

Figure 34: Known key correlation on 1.6M PWC traces with Icom

With the results of running the KnownKeyCorrelation module, we are confident that performing DPA on the band pass filtered trace set should be able to recover some bytes of the key. With 1.6 million traces in the trace set, we recover up to 10 bytes of the key, and we perform a full key recovery with an additional 1.2 million traces. Thus, we perform full key recovery at 2.8 million traces. For further experiment, we resampled the this 2.8 million traces trace set after applying the frequency filter to 4.15MHz, 10.742MHz, and 17.334MHz and perform DPA on each of these resampled trace sets. None of these new trace sets were able to recover any bytes of the key. Table 8 displays this said results. This suggests that the some of the information leakage is hidden in the 10.742MHz frequency while some of the other information leakage is hidden in the 17.334MHz frequency.

Frequency	Traces	Correct Key Bytes
5MHz to 25MHz	1.6M	10
5MHz to 25MHz	2.8M	16
4.15MHz	2.8M	0
10.742MHz	2.8M	0
17.334MHz	2.8M	0

Table 8: DPA with current probe; HW/SBox 1st round; Downshifting with Icom

In summary, we perform full key recovery at 2.8 million traces with the set up involving the Icom. However, recall that only 80k traces are needed to guarantee a full key recovery without the Icom, so frequency downshifting with the Icom is not an improvement over the traditional set up with the current probe. The signal that the IF port of the Icom produced contains high frequency noises that need to be filtered out. In addition, the Icom also receives external noises from other sources such as cell

phones and other nearby electronic equipments, and traces containing these noises will have to be thrown out. The total execution time of performing DPA with the Icom on 2.8M traces is 3 days with acquisitions and analysis whereas traditional method required only an hour of execution time. Thus, using the Icom as a method of performing DPA is not viable replacement, or an improvement, for DPA. In the next section, we will explore how downshifting affects DEMA.

11.3 DEMA with the Icom

In this section, we will be discussing the results of performing DEMA with the set up with the Icom. We have already described the set up in section 11.2, and figure 29 shows a diagram of the set up. The only difference is replace the current probe with the HS EM probe. Once again, an impedance matcher is placed between the Icom and the LeCroy in order prevent the Icom from overloading the scope with current, and we will show results of removing the impedance matcher in a later section.

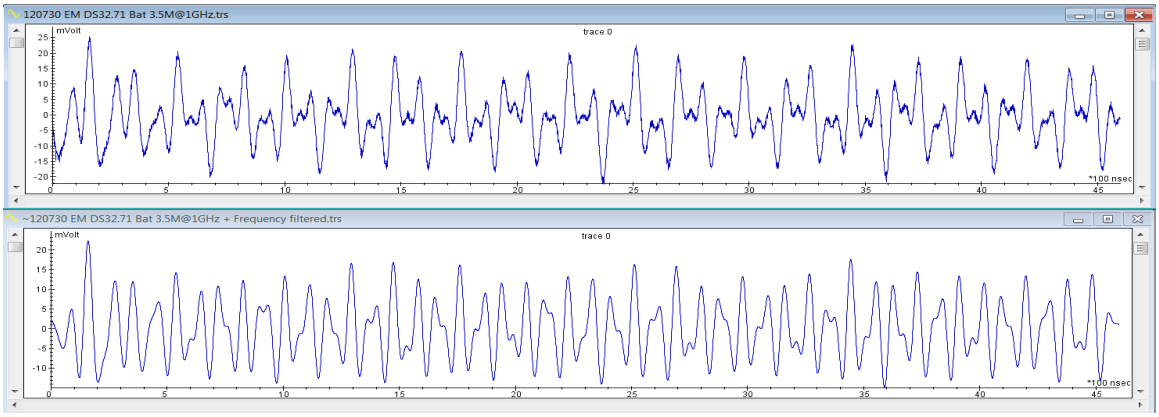

Figure 35: Sample trace of EM during AES with Icom; unfiltered trace on top; 5MHz to 25MHz band pass filtered on bottom

For the experiment with DEMA, we obtain 3.5 million traces. We once again attempt to perform DEMA by simply resampling the trace set to 4.15MHz, 10.742MHz,

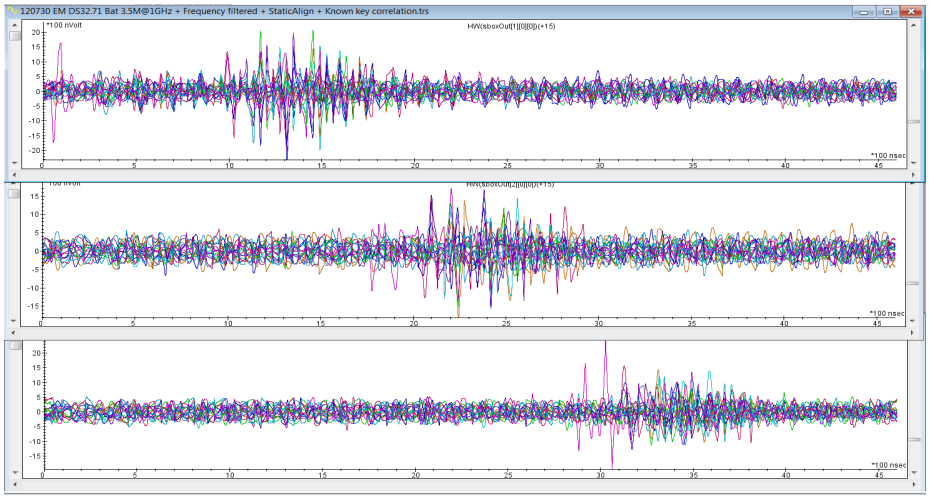

Figure 36: Known key correlation on 3.5M EM traces with Icom

and 17.334MHz. Again, none of these new trace sets were able produce any correct key bytes, so we applied the band pass filter to the trace set once again. Since the IF port of the Icom always produce signals in the same frequencies, we simply apply the same band pass filter from 5MHz to 25MHz to the trace set. Figure 35 shows sample trace before and after applying the band pass filter. Note that the filtered trace from the EM probe looks very similar to the filtered trace from the trace taken with the current probe; this is most likely due to the signals coming from the same source, which is the IF port.

Before we perform DEMA on the filtered trace set, we ran the KnownKeyCorrelation module on the trace set. Figure 36 shows the result of running the KnownKeyCorrelation on the filtered trace set. The result shown is once again Hamming Weight of SBox out, and each trace shows the correlation between the key bytes and the trace set. We can see that the key bytes correlate to the first round's Hamming Weight of the SBox out between 0.9ms and 1.5ms; the correlation between the key bytes and the second round's Hamming Weight of the SBox out is between 2.0ms and

3.0ms, and the third round is between 3.0ms and 4.0ms. If we compare the result of the KnownKeyCorrelation between the EM trace set and the trace set with the current probe shown in figure 36 and figure 34 respectively, we can see that the EM trace set have a much stronger correlation between the key bytes and the trace sets. This is merely a demonstration of how a stronger correlation looks like as there are 0.7M more traces to perform the known key correlation in this trace set than the one presented in figure 34.

Since the result of the KnownKeyCorrelation shows a strong correlation between the key bytes and the trace set, we expect a full key recovery if not near full key recovery. The key we recovered from the performing DEMA on the trace set is the following: 524953435552454953f74f4f4c2131f0. Examining the recovered key bytes shown that only 2 key bytes are incorrect. Recall that we recovered 15 bytes of the key with 4 million traces on the normal set up, so the number of traces needed to recover most of the key is consistence with previous results.

In summary, we recover 15 bytes of the key bytes with the set up with the Icom up to 3.5 million traces, and this is comparable to the previous results without the Icom. None of key bytes was recovered from performing DEMA on trace sets resampled to 4.15MHz, 10.742MHz, and 17.334MHz. This is consistence with the previous experiment of performing DPA with the Icom by resampling to these frequencies, and the two results highly suggests the information leakage is hidden in combination of the 10.742MHz and 17.334MHz frequencies. Overall, the set up with the Icom is a possible alternative set up for the normal set up. Since the signal has been downshifted from a higher frequency down to below 20MHz, performing a DEMA will require less samples per traces. This implies less calculation is needed in order to

perform DEMA on a target. It is possible that we can use the set up with the Icom for target operating at a higher frequency for downshifting in order to improve the efficiency of performing DEMA, but further research will needed to be done in order to confirm such claim. In the next section, we will discuss the results of using the set up with the Icom in different configurations.

11.4 DPA with the Icom with other configurations

In this section, we will show the results of applying different configurations to the set up with the Icom. The set ups are current probe with no impedance matcher, current probe with AC coupling and no impedance matcher, and finally, current probe with AC coupling and no impedance matcher and Icom tuned to 65.43MHz. The reason for applying these configurations is to see if we can apply different configurations to improve the performance of performing DPA/DEMA with the Icom.

The first configuration is simply remove the impedance matcher. Recall that we place a 50 Ohms impedance matcher between the Icom and the LeCroy in order to prevent the Icom from overloading the scope with current. However, the impedance matcher will cause some signal loss between the Icom and the scope, and this can potentially distort the captured signal enough that the information leakage is no longer within the signal [33]. In order to verify that the impedance matcher is not interfering with our experiment, we used a multimeter to check that the current coming out of the IF will not overload the LeCroy. Hence, it is safe to connect the IF port directly to the scope without the impedance matcher. We took 150k traces with the Icom directly connected to the impedance matcher since only 80k traces are needed to perform full key recovery on the normal set up. We performed DPA on the aligned trace set sampled at 1GHz, but zero key byte was recovery from the trace set.

In addition, we resampled the trace set to 10.742MHz and 17.334MHz and performed DPA on the resampled trace sets with no success. We once again applied a band pass filter of 5MHz to 25MHz to the trace set, then we ran the KnownKeyCorrelation module on the filtered trace set, and figure 37 shows the results of running the module. From the figure, we can see that there is little to no correlation between the key bytes and the trace set on the first round, and absolutely no correlation between the key bytes and the trace set on the second round. Thus, the 50 Ohms impedance matcher is not a negative factor in our experiments.

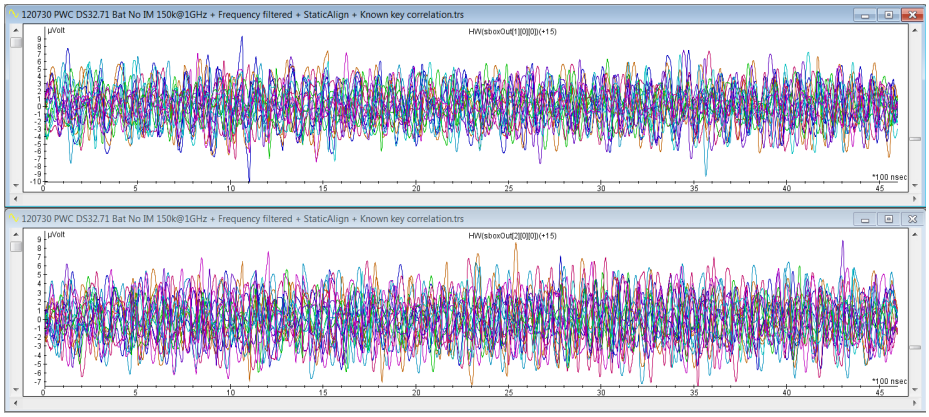

Figure 37: Known key correlation on 150k PWC traces with Icom and no impedance matcher

The next configuration is to use the current probe with no impedance matcher and set the LeCroy scope to AC coupling with 1M Ohms impedance. If the scope is set to DC coupling, the actual signal is measured; if the scope is set to AC coupling, then the DC component of the signal is removed from the trace after a high pass filter is applied. By removing the DC component of the signal, we increase the resolution of the signal [34]. Figure 38 shows sample traces of both DC coupling and AC coupling. Note that the signal range for the trace with DC coupling is at -/+20mV, and the signal range for the trace with AC coupling is at -/+50mV. Figure 39 shows sample

trace captured during the entire AES operations; the top trace is unfiltered sampled at 1GHz, and the bottom trace is resampled to 17.334MHz. Note that the resampled trace shown more distinguishable AES operations (e.g. input/output operations and AES encryption) as we have seen in section 9.3.1 in the trace compared to the traces in figure 32. The input and put operations can easily be identified, and the AES encryption operations are between 10ms and 22ms. However, zero byte of the key was recovered from performing DPA on the aligned trace set resampled at 1GHz. In addition, none of the key bytes was recovered from trace sets with 150k traces resampled to 10.742MHz and 17.334MHz. The trace set applied with a 5MHz to 25MHz band pass filter also recovered zero bytes of the key. Thus, the DC coupling setting is also not a negative factor in our experiments.

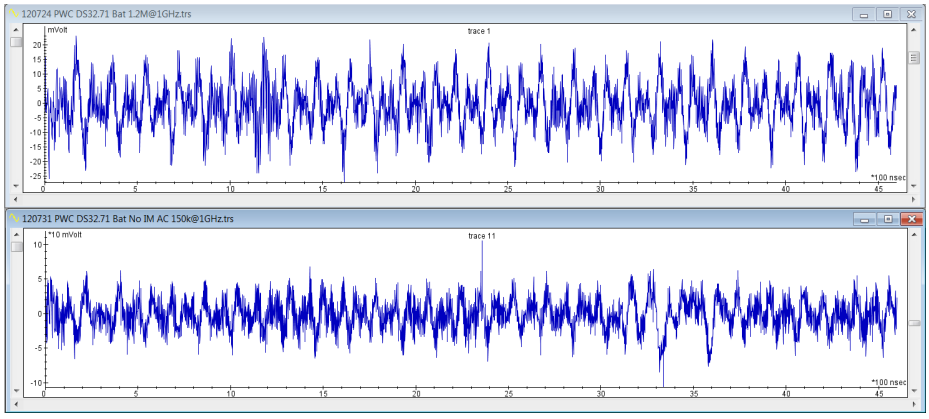

Figure 38: Sample traces with current probe and Icom; DC coupling on top; AC coupling on bottom

The last configuration we tried is to set the Icom to tune to 65.43MHz. Recall that this 65.43MHz frequency is the upper harmonic frequency of the operating frequency of the Xmega. In the experiments we conduct earlier, we see that the trace set resampled to 65.43MHz shown similar, if not better, DPA results than trace set resampled to 32.71MHz. Thus, it is worth while to attempt to take traces with

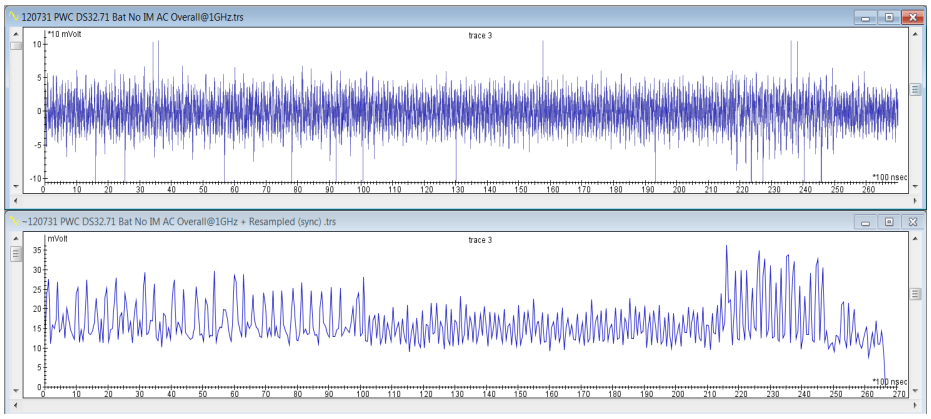

Figure 39: Sample traces of all AES operations with current probe and AC coupling at 1GHz; unfiltered on top; 17.334MHz resampling on bottom

the Icom tuned to 65.43MHz. This set up also has no impedance matcher and the scope is set for AC coupling. Figure 40 shows a sample trace taken with this set up. Unfortunately, zero byte of the key was recovered from the aligned trace set sampled at 1GHz. In addition, performing DPA on trace set resampled at 10.742MHz and 17.334MHz and trace set applied with band pass filtered ranged from 5MHz to 25MHz recovered zero bytes of the key up to 100k traces. Therefore, tuning the Icom to the upper harmonic did not improve the efficiency on DPA.

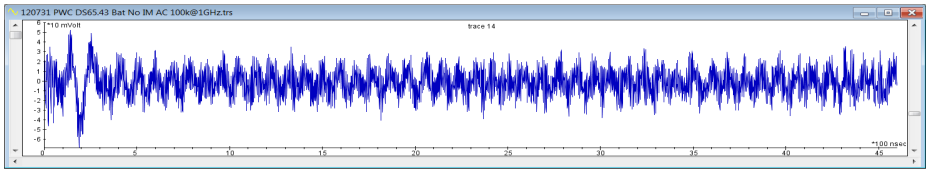

Figure 40: Sample trace with current probe and Icom tuned for 65.43MHz

Although this is not a configuration, this last experiment we ran was to cover all of our basis to make sure we did not miss anything. Recall that we perform the band pass filter from 5MHz to 25MHz in all of our experiments in this section, so the 4.15MHz frequency in the signals was eliminated from the trace sets. For all the trace set that we obtained in this section, we run the band pass filter and perform

DPA on all of them again, but the band pass filter is from 1.5MHz to 25MHz instead. The DPA performed on these filtered trace sets did not produce any byte of the key. Therefore, the 4.15MHz frequency did not contribute to the information leakage.

11.5 Summary of the Downshifting with Icom

In this section, we experimented DPA and DEMA with a slightly different set up. We place an Icom R7000 radio frequency receiver between the probes and the LeCroy oscilloscope. The Icom served as a signal downmixer, and it shifts high frequency signals down to a mere 10.742MHz. By shifting the signal down to 10.742MHz, we can reduce the number of calculation during DPA and DEMA and thus improve their efficiencies.

The results of our experiment show that the downshifting does not improve the efficiency of DPA with the current probe. The number of traces needed for full key recovery increased from 80k to 2.8 million traces. On the other hands, we produced similar results for DEMA with the downshifting experiments. Fourteen bytes of key were recovered from performing DEMA on 3.5 millions traces taken with the Icom, and fifteen bytes of the key were recovered from performing DEMA on 4 millions traces taken with the normal set up. Thus, it is possible to increase the efficiency of DEMA using this downshifting technique.

CHAPTER 12

Comparison between DPA and DEMA

Several experiments were perform on the Xmega in regards to side channel attacks. In sections 9.3.1 and 9.4.1, we began with measuring the Xmega with the EM probe, and we began taking measurments of the Xmega's change in voltage while it is performing AES encryption in section 10.2. Finally, we took measurements of the Xmega with the current probe while it is performing AES encryption in section 10. In addition, we performed experiments where we used a radio receiver as a demodulator in section 11.2. Table 9 shows the best results of each experiments.

Model	Frequency	Traces	Downshift	Correct Key Bytes
EM@L1	65.43MHz	3.1M	No	11
EM@L2	32.71MHz	225k	No	15
EM@L2	65.43MHz	125k	No	15
EM@L2	130.86MHz	150k	No	9
PWR	32.71MHz	0.5M	No	0
PWR	65.43MHz	0.5M	No	0
PWR	1GHz	2.5M	No	16
PWC	32.71MHz	45k	No	16
PWC	65.43MHz	100k	No	16
PWC	1GHz	40k	No	15
EM@L1	1GHz*	3.5M	Yes	14
PWC	1GHz*	2.8M	Yes	16

Table 9: Summary of all experiments with best results; PWR denotes power/resistor, and PWC denotes power/current; * = band pass filtered from 5MHz to 25MHz

12.1 Summary of DEMA

We began by examining the results of the experiments conducted with the EM probe. In our experiments, we placed the EM probe at two different locations. Location 1, or L1 for short, is located near the center of the chip, and location 2, or

L2 for short, is located on the edge of the chip. We chose these two locations based on the Spectral Intensity graph shown in figure 8. The figure shows that L1 and L2 have the highest amount of activities at the 32MHz frequency, with a bandwidth of -/+0.2MHz, while performing AES encryption operations.

The first experiment we conducted with the EM probe is at location 1. This location was chosen first since it is most likely where the cryptographic engine is located based on the schematic diagrams. We set the LeCroy oscilloscope to sample at 1 GHz. The signals from the EM probe can ranged from -/+500mV to -/+3V depending on how far the probe is away from the Xmega; these signals is much stronger compared to the current probe and the signals from measuring the change in voltage. Note that only 27000 samples per trace is needed in order to capture the entire AES encryption operation sampled at 1 GHz, and this holds true for all the experiments conducted with the Xmega. While we measured the Xmega with both the high sensitivity and low sensitivity probes, the results discussed in this section will only consists of traces taken with the high sensitivity probe.

At location 1, we were able to captured up to 3.1 million traces with the high sensitivity probe. We resampled these 3.1 million traces to 32.71MHz and 65.43MHz, which are the operating frequency of the Xmega and the upper harmonic, and we performed DEMA on these resampled trace sets. The trace sets resampled at 32.71MHz recovered up to 10 bytes of the key with 1.5 million traces and recovered up to 10 bytes with 3.1 million traces. On the other hands, the trace sets resampled at 65.43MHz recovered up to 12 bytes with 1.5 million traces and recovered up to 11 bytes with 3.1 million traces. Since doubling the amount of traces shows no improvement in number of key bytes recovered, we looked for an alternative source of leakage, and that is

location 2.

Location 2 is located on the edge of the chip, and we believed this location is where the input/output operations are being conducted for the chip. Regardless of what we believed which functions are located on the chip, the leakage information of the key exists in location 2 as well. We were able to capture up to 4 millions traces at location 2. By resampled to the trace set down to 32.71MHz, we were able to recover up to 15 bytes of the key with 225k traces. Furthermore, we were able to recover up to 15 bytes of the key with 125k traces with the trace set resampled down to 65.43MHz. However, we were not able to recover the last byte of the key even using up all 4 million traces. Note that there are twice as many samples per trace in the trace set resampled to 65.43MHz compared to the samples per trace in the trace set resampled to 32.71MHz, so the amount of time to perform DEMA on 125k traces resampled to 65.43MHz is about the same as the amount of time to perform DEMA on 225k traces resampled to 32.71MHz.

12.2 Summary of DPA

In terms of DPA, we conducted the experiments in two ways. The first way is to place a resistor on the ground wire of the chip, and we measure the change in voltage across the resistor as the Xmega performs AES encryption. The second way is simply place a current probe on the VCC wire of the Xmega, and we measure the signal from the current probe. Once again, the LeCroy was set to sample at 1 GHz, and the signals ranged within the -/+ 100mV with a 12V amplifier placed between the scope and the probes in both cases. The signal strength of the current and power probe is much weaker compared to the signals from the EM probe. The signal strength of the EM is based on the distance between the probe and the target. In our case, we

placed the probe about 1mm away from the target. However, the signals from the current probe and measured across the resistor shown much better results than the signals from the EM probe regardless of the strength of the signals.

We began DPA with measuring the change in voltage across the resistor placed on the ground wire of the Xmega. This method of doing DPA will depend on the resistance of the resistor. A stronger resistor will give a stronger signal to perform DPA on, but the stronger resistor has a higher chance of causing failure on the Xmega. Luckily, we did not encounter any calculation error from the Xmega cause by placing resistor as strong as 1M Ohm. We began with measuring across a 1k Ohm resistor, and we managed to capture 1.5 million traces with this set up. Initial attempts at performing DPA with trace set resampled to 32.71MHz and 65.43MHz showed no result up to 0.5 million traces, so we perform DPA on the unfiltered traces sampled at 1 GHz. We recovered 8 bytes of the key with 1 million traces, and we recovered 14 bytes of the key with 1.5 million traces. Afterward, we tested the effects of using a stronger resistor, and we replaced the 1k Ohm resistor with a 1M Ohm resistor. We were able to capture 4 million traces with the 1M Ohm resistor for our experiments. At 1M traces, this trace set taken with the 1M Ohm resistor was able to recover 13 bytes of the key and was able to recover 15 bytes of the key with 1.5M traces. This was a sign of improvement over the 1k Ohm resistor. At the end, we did not need all 4 million traces to perform a full key recovery, and we only needed 2.5 million traces. This result was already an improvement over DEMA since we never managed to perform a full key recovery with DEMA.

The next set of experiments was measuring the current on the VCC wire of the Xmega while it is performing AES encryptions. In addition, a battery power source

was used in place of the DC lab power supply that we were using in the previous experiments. The signals captured with this set up is even weaker than the set up with the resistor, and the range of the signal with this set up is in the -/+10mV without an amplifier. On the other hands, this set up proved to be most effective among all the other experiment that we have conducted. Trace set resampled from 1 GHz down to 32.71MHz is used to perform DPA, and only 45k traces are needed to perform full key recovery. In addition, trace set resampled to 65.43MHz only required 100k traces to perform full key recovery. The First Order Stats module shown that only 80k traces are needed to guarantee full key recovery. Over all, using the current probe seems to be the best method in performing side channel attack on an embedded system.

Lastly, we will briefly mention about the experiments with using a radio receiver as a demodulator. In this experiment, we performed the experiments with the EM probe and the current probe with an addition component to the set up. We placed an Icom R7000 radio receiver between the probes and the LeCroy. The radio receiver was tuned to the operation frequency of the Xmega at 32.71MHz, and the LeCroy was connected to the Icom via the IF port. The IF port always produces a 10.74MHz signal. In summary, we were able to recover 14 bytes of the key with 3.5 million EM traces taken at L1, and we were able to recover all key bytes with 2.8 million traces taken with the current probe. In terms of performance, the set up with the Icom is comparable for EM, but the set up with the Icom is no improvement for the current probe. It is a possible alternative set up for EM.

12.3 DEMA vs DPA

We will compare the perform of DEMA and DPA with and without the Icom. We will begin with the case without the Icom. In section 12.1, we concluded that the best result for performing DEMA on the Xmega is to acquire traces at location 2 presented in figure 8. We shown that the time it takes to perform DEMA on 125k traces resampled to 65.43MHz is the same amount of time it takes to perform DEMA on 225k traces resampled to 32.71MHz, and this will allow us to recover all bytes of the key. The said time is 1 hour on a 3.0GHz machine. However, it takes 45 minutes to acquire 125k traces, and it takes 1.5 hour to acquire 225k traces. Thus, the best strategy, time wise, for performing DEMA on the Xmega is to acquire 125k traces on location 2 with the HS probe, and perform analysis on trace set resampled to 65.43MHz. The total time for DEMA with this strategy will be 1 hour and 45 minutes on a 3.0GHz machine. On the other hands, we mentioned that the best strategy for performing DPA is to acquire 80k traces using the current probe in section 12.2. The total time for performing DPA with this strategy is 1 hour on a 3.0GHz machine. Thus, DPA has a 45 minutes gain in full key recovery over DEMA in term of time performance. The difference in number of traces is 45k in favor of DPA. According to our data, DPA 50 percent faster than DEMA in the case with the traditional methods. Thus, an attack should perform DPA with the current probe for the best perform time wise with the least amount of traces.

In section 11.2, we performed DEMA/DPA with the Icom as a downmixer. In section 12.2, we mentioned that 3.5 millions EM traces were taken at location 1, but only 14 bytes of the key were recovered. Time wise, it takes 4 days to acquire 3.5 millions traces, and it takes 1 day to analysis the aligned trace set sampled at 1GHz

to produce the 14 bytes key recovery on a 3.0GHz machine. Thus, the total time it takes to perform DEMA with only 14 bytes of key recovered is 5 days. On the other hands, full key recover was performed with DPA by acquiring traces with the current probe. This was done with 2.8 millions aligned traces sampled at 1GHz, and it takes 3 days for the acquisition to complete. Furthermore, the analysis phase takes 1 day to complete. Thus, the total time to perform full key analysis with the current probe and the Icom is 4 days on a 3.0GHz machine. The total gain in time for DPA over DEMA with the Icom is 1 day, and the DPA need 0.7 million less traces than DEMA with the Icom. With the Icom, DPA is still faster than DEMA, but DPA is only faster by 20 percent according to our data. In both case with and without the Icom, DPA with the current probe is best method in performing full key recovery on the Xmega.

CHAPTER 13

Conclusion

Side channel attack is an emerging field of studies in computer hardware designs. Side channel attacks can allow attackers to obtain the secrets hidden inside hardware without attacking the logic behind the cryptographic algorithms. There are many forms of side channel attacks; temperature and sound based side channel attacks are some examples. Some of the more complex side channel attacks are Differential Power Analysis (DPA) and Differential Electromagnetic Analysis (DEMA). They are statistical attacks that require the use of power or electromagnetic traces of the target while it is perform cryptographic operations. A power or electromagnetic trace consists of the voltage or current readings of the target in respected to time. Given enough traces, an attacker can determine the secret key that is hidden in the hardware.

In this paper, we performed DPA and DEMA on an ATXmega256A3B microcontroller, and it is a popular series of microcontroller that is used in many places. We chose the ATXmega256A3B microcontroller as our target because it is capable of performing AES encryption and decryption on the hardware level. The goal of this paper is to compare the effectiveness of DPA and DEMA on embedded system such as the ATXmega256A3B microcontroller. In terms of raw strength, the EM probe was able to produce a much stronger signal than the current probe. For DEMA, we used a high sensitivity EM probe to obtain traces, and we were one byte short of full key recovery with 125k traces. On the other hands, only 45k traces were needed to perform full key recovery using a current probe. Thus, while the EM probe can produce stronger signals, the signals capture by the EM probe can also be very noisy

compared to the current probe. Overall, DPA with the current probe is more effective than DEMA with the results that we produced.

In addition to the comparing the effectiveness of DPA and DEMA on embedded system, we also performed experiment with using a radio receiver as a signal demodulator. The radio receiver will take any signal and shifts it down to a mere 10.7MHz signal. This downshifting can reduce the number of calculations needed to perform DPA and DEMA. Once again, the number of traces needed to perform DPA was recorded to be less than that of number of traces needed for DEMA. Thus, the downshifting experiments reinforced the claim that DPA is more effective than DEMA. On the other hands, we also found that the set up with the radio receiver serving as a downmixer could potentially be used as an alternative to the traditional DEMA set up, but further research would need be conducted to verify this claim.

LIST OF REFERENCES

[1] Mangard, Stefan; Oswald, Elisabeth; Popp, Thomas. "Power Analysis Attacks: Revealing the Secrets of Smart Cards". *Springer Science and Business Media*. New York, New York, 2007.

[2] Kocher, Paul; Jaffe, Joshua; Jun, Benjamin. "Differential Power Analysis". Cryptography Research, Inc. San Francisco. 1998.

[3] Gandolfi, Karine; Mourtel, Christophe; Olivier, Francis. "Electromagnetic Analysis: Concrete Results". *Cryptographic Hardware and Embedded Systems*. 2001.

[4] Lomne, Victor; Maurine, Philippe; Torres, Lionel; Robert, Michel; Soares, Rafael; Calazans, Ney. "Evaluation on FPGA of Triple Rail Logic Robustness against DPA and DEMA". *Design, Automation and Test in Europe Conference and Exhibition*. 2009

[5] Ding, Guo-liang; Li, Zhi-xiang; Chang, Xiao-long; Zhao, Qiang. "Differential Electromagnetic Analysis On AES Cryptographic System". *Web Mining and Web-based Application*. 2009

[6] Sauvage, Laurent; Guilley, Sylvain; Danger, Jean-Luc; Mathieu, Yves; Nassar, Maxime. "Successful Attack on an FPGA-based WDDL DES Cryptoprocessor Without Place and Route Constraints". *Design, Automation and Test in Europe Conference and Exhibition*. 2009.

[7] Nambiar, Vishnu; Khalil-Hani, Mohamed; Zabidi, Mun'im. "Accelerating the AES encryption function in OpenSSL for Embedded Systems". *Electronic Design*. 2008

[8] De Mulder, E.; Buysschaert, P.; Delmotte, P.; Preneel, B.; Vandebosch, G.; Verbauwhede, L. "Electromagnetic Analysis Attack on an FPGA Implementation of an Elliptic Curve Cryptosystem". *Computer as a Tool*. 2005.

[9] Krishnamurthy, R.; Mathew, S.; Sheikh, F. "High-performance energy-efficient encryption in sub-45nm CMOS era". *Design Automation Conference*. 2011

[10] Stamp, M. "Information Security: Principles and Practice". John Wiley and Sons, Inc. Hoboken, New Jersey. 2006.

[11] Ambrose, J.; Ignjatovic, A.; Parameswaran, S. "Power Analysis Side Channel Attack: The Processor Design-level Context". VDM Verlag Dr. Muller Aktiengesellshaft and Co. Saarbrucken, Germany. 2010.

[12] "Data Encryption Standard (DES)". National Institute of Standards and Technology. Gaithersburg, Maryland. 1988.

[13] "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher". National Institute of Standards and Technology. Gaithersburg, Maryland. 2008.

[14] "Break DES in less than a single day". SciEngines. Kiel, Germany. 2009

[15] Koblitz, N. "Elliptic curve cryptosystems". *Mathematics of Computation*. 1987.

[16] "Recommendation for Key Management - Part 1: General (Revise)". National Institute of Standards and Technology. Gaithersburg, Maryland. 2007.

[17] Bertoni, G.; Breveglieri, L.; Fragneto, P.; Macchetti, M.; Marchesin, S. "Efficient Software Implementation of AES on 32-Bit Platforms". *Cryptographic Hardware and Embedded Systems*. 2003

[18] "Announcing the Advanced Encryption Standard (AES)". National Institute of Standards and Technology. Gaithersburg, Maryland. November 26, 2001.

[19] Biryukov, A.; Khovratovich, D. "Related-key Cryptanalysis of Full AES-192 and AES-256". University of Luxembourg. 2009

[20] Rivest, R.; Shamir, A.; Adleman, L. "A Method for Obtaining Digital Signature and Public-Key Cryptosystems". *Communications of the ACM*. 1978.

[21] Augier, M.; Bos, J.W.; Kleinjung, T.; Wachter, C. "Ron was wrong, Whit is right". EPFL. Lausanna, Switzerland. 2011.

[22] Brumley, D.; Boneh, D. "Remote Timing Attacks are Practical". SSYM'03 Proceedings of the 12th conference on *USENIX Security Symposium*. 2003.

[23] ISO/IEC 7810:2003 "Identification cards - Physical characteristics"

[24] Chen, S.; Wang, R.; Wang, X.F.; Zhang, K. "Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow". *IEEE Symposium on Security and Privacy*. 2010.

[25] Kocher, P. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". Cryptography Research, Inc. San Francisco, USA.

[26] Shamir, A.; Tromer, E. "Acoustic cryptanalysis". Blavantnik School of Computer Science, Tel Aviv University. November, 2011.

[27] "Atmel AVR XMEGA A Manual Preliminary", Atmel. 2009.

[28] "Atmel 8/16-bit AVR XMEGA A3B Microcontroller", Atmel. 2010.

[29] "Icom Instruction Manual: Communications Receiver IC-R7000", Icom Inc. 1988.

[30] Kizhvatov, I. "Side Channel Analysis of AVR XMEGA Crypto Engine". University of Luxembourg.

[31] J. R. Rao, P. Rohatgi. "EMpowering Side-Channel Attacks." Cryptology ePrint Archive, http://eprint.iacr.org/, Report 2001/037, 2001.

[32] E. Prouff. "DPA attacks and S-boxes." In proceedings of FSE 2005, LNCS 3557, pp. 424 - 441, Springer, 2005.

[33] "LeCroy Application Note 016", LeCroy Corporation.

[34] "Application Note 3768: To AC-Couple or Not to AC-Couple? That is the Question!". Maxim Integrated. 2006.

APPENDIX A

AES driver

```
// \brief Polled function that does an AES encryption on one 128-bit data
    block.
// \note This code is blocking and will dead lock if no interrupt flags
    are set.
// \param plaintext Pointer to the plaintext that shall be encrypted
// \param ciphertext Pointer to where in memory the ciphertext (answer)
    shall be stored.
// \param key       Pointer to the AES key
// \retval true  If the AES encryption was successful.
// \retval false If the AES encryption was not successful.
bool AES_encrypt(uint8_t * plaintext, uint8_t * ciphertext, uint8_t * key)
{
        bool encrypt_ok;

        /* Load key into AES key memory. */
        uint8_t * temp_key = key;
        for(uint8_t i = 0; i < AES_BLOCK_LENGTH; i++){
                AES.KEY = *(temp_key++);
        }
        /* Load data into AES state memory. */
        uint8_t * temp_plaintext = plaintext;
        for(uint8_t i = 0; i < AES_BLOCK_LENGTH; i++){
                AES.STATE = *(temp_plaintext++);
```

```
        }
        /* Set AES in encryption mode and start AES. */
        AES.CTRL = (AES.CTRL & (~AES_DECRYPT_bm)) | AES_START_bm;
        PORTC.OUT |= (1<<1) | 0x01; //Begin HW trigger
        do{
                /* Wait until AES is finished or an error occurs. */
        }while((AES.STATUS & (AES_SRIF_bm|AES_ERROR_bm) ) == 0);
        PORTC.OUTCLR |= (1<<1); //Clear HW trigger
        /* If not error. */
        if((AES.STATUS & AES_ERROR_bm) == 0){
                /* Store the result. */
                uint8_t * temp_ciphertext = ciphertext;
                for(uint8_t i = 0; i < AES_BLOCK_LENGTH; i++){
                        *(temp_ciphertext++) = AES.STATE;
                }
                encrypt_ok = true;
        }else{
                encrypt_ok = false;
        }
        return encrypt_ok;
}
```

APPENDIX B

AES run

```
/* Key used when AES encryption is done operations. */
uint8_t key[BLOCK_LENGTH] = {0x52, 0x49, 0x53, 0x43, 0x55, 0x52, 0x45,
    0x49,
                             0x53, 0x43, 0x4F, 0x4F, 0x4C, 0x21, 0x31, 0x00};

uint8_t lastsubkey[BLOCK_LENGTH];
uint8_t read_key[BLOCK_LENGTH];

/* Variable used to check if decrypted answer is equal original data. */
bool success;

int main(void) {
        int data;
        int index=0;
        char buffer[BLOCK_LENGTH];
        Config32MHzClock();
        CLK.PSCTRL = 0x00; // no division on peripheral clock
        PORTC.DIR |= (1<<1);
        PORTCFG.CLKEVOUT = PORTCFG_CLKOUT_PE7_gc;
        PORTE.DIR = (1<<7); // clkout

        // configure PORTF, USARTF0 (PORTF:3=Tx, PORTF:2=Rx) as asynch
            serial port
```

```
        // This will connect to the USB-Serial chip on EVAL-USB boards
        // For other boards rewrite all occurences of USARTF0 below with
           USARTE0
        // then you can use PORTE:2,3 as asynch serial port (EVAL-01,
           EVAL-04 boards)
        PORTF.DIR |= (1<<3) | (1<<0); // set PORTF:3 transmit pin as output
        PORTF.OUT |= (1<<3);       // set PORTF:3 hi
        USARTF0.BAUDCTRLA = 207; // 9600b (BSCALE=207,BSEL=0)
        USARTF0.CTRLB = USART_TXEN_bm | USART_RXEN_bm; // enable tx and rx
           on USART
        while(1) {
                data=UsartReadChar(); // read char
                if(index==sizeof(buffer)-1) {
                        buffer[index]=data;       // null terminate
                        index=0;                // reset buffer index
                DoAES(buffer);
                } else {
                buffer[index++]=data;
                };
        };
};

void DoAES(char plaintext[]) {
        /* Variables used to store the result from a single AES
           encryption/decryption .*/
        uint8_t single_ans1[BLOCK_LENGTH];
        /* Assume that everything is ok*/
```

```
        success = true;
        /* Before using the AES it is recommended to do an AES software
           reset to put
         * the module in known state, in case other parts of your code has
            accessed
         * the AES module. */
        AES_software_reset();
        AES_interruptlevel_set(1);
        /* Generate last subkey. */
        AES_lastsubkey_generate(key, lastsubkey);
        success = AES_encrypt(plaintext, single_ans1, key); //Call
           AES_driver.c for encryption
        if(success) {
                UsartWriteChar(0x00); //For Inspector's protocal
                UsartWriteChar(0x10);
                UsartWriteDatabytes(single_ans1);
        } else {
                UsartWriteString("Failed to encrypt\n");
        }
}

void UsartWriteDatabytes(char *string) {
        int i;
        for(i=0;i<BLOCK_LENGTH;i++) {
                UsartWriteChar(*string++);
        }
};
```

```
void UsartWriteChar(unsigned char data) {
        USARTF0.DATA = data; // transmit ascii 3 over and over
        if(!(USARTF0.STATUS&USART_DREIF_bm))
                while(!(USARTF0.STATUS & USART_TXCIF_bm)); // wait for TX
                    complete
        USARTF0.STATUS |= USART_TXCIF_bm; // clear TX interrupt flag
};

unsigned char UsartReadChar(void) {
        while(!(USARTF0.STATUS&USART_RXCIF_bm)); // wait for RX complete
        return USARTF0.DATA;
};

// write out a simple '\0' terminated string
void UsartWriteString(char *string) {
        while(*string != 0) UsartWriteChar(*string++);
};

void Config32MHzClock(void) {
        CCP = CCP_IOREG_gc; //Security Signature to modify clock
        // initialize clock source to be 32MHz internal oscillator (no PLL)
        OSC.CTRL = OSC_RC32MEN_bm; // enable internal 32MHz oscillator
        while(!(OSC.STATUS & OSC_RC32MRDY_bm)); // wait for oscillator ready
        CCP = CCP_IOREG_gc; //Security Signature to modify clock
        CLK.CTRL = CLK_SCLKSEL_RC32M_gc; //select sysclock 32MHz osc
        // update baud rate control to match new clk
```

```
        USARTF0.BAUDCTRLA = 207; // 9600b (BSCALE=207,BSEL=0)
};
```