Effectiveness of Variable Bit-Length Power Analysis Attacks on SHA-3 Based MAC

Xuan D. Tran
Communication Systems
Harris Corporation
Rochester, New York 14610
xtran@harris.com

Marcin Łukowiak
Department of Computer Engineering
Rochester Institute of Technology
Rochester, New York 14623
mxleec@rit.edu

Stanisław P. Radziszowski
Department of Computer Science
Rochester Institute of Technology
Rochester, New York 14623
spr@cs.rit.edu

***Abstract*—Keccak is the hash function selected by NIST as the new SHA-3 standard. Keccak is built on sponge construction, and it provides a new keyed Message Authentication Code (MAC) function called MAC-Keccak. The focus of this work was to apply the power analysis attacks that use Correlation Power Analysis (CPA) technique to extract the key. Our attack methodology utilizes simulated power consumption waveforms and is applied to the Keccak high-speed core hardware design from the SHA-3 competition. 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit CPA selection function key guess size attacks are performed on the waveforms to compare and analyze computational effort of successful key extraction on MAC-Keccak. Our experiments show that the larger the selection function key guess size used, the better the signal-to-noise-ratio (SNR), therefore requiring fewer traces needed to be applied to retrieve key but using more computation time.**

I. Introduction

A cryptographic hash is a one-way function used to construct a short digital fingerprint, also known as the message digest, of an arbitrary-length input data.

$$H : \{0,1\}^* \rightarrow \{0,1\}^m$$

There are many well known hash functions used by industry and government agencies. With the practical and successful attacks on the MD5 hash function to find a collision in a few seconds, and theoretical attacks on SHA-1 with 2^{63} operations versus simple birthday paradox implied bound of 2^{80} operations, NIST called for the design of a new standard cryptographic hash function, SHA-3, that will replace (or complement) the current Secure Hash Standard (SHS). On October 2, 2012, after five years of carefully narrowing down the set of candidates, reviewing the public comments, and with final internal review, NIST announced Keccak as the winner.

Side Channel Attacks (SCA) exploit weaknesses in implementation of cryptographic functions resulting from unintended side effects such as operation timing, electromagnetic radiation, thermal/acoustic emanations and power consumption, to break cryptographic schemes with no known weaknesses in their mathematical structure. Many of these attacks may also be performed using relatively cheap and easily obtained measurement equipment, making them a significant concern for both the algorithm design and implementation. The focus of this research was the implementation of a methodology for performing power analysis attacks, a type of side-channel attack in which secret information is revealed through the instantaneous power consumption of a circuit. Specifically, an attack framework was developed and applied to SHA-3 based MAC using gate-level circuit simulation. The results indicate that SHA-3 based MAC is vulnerable to power analysis attacks.

The remainder of the paper is organized as follows: Section II discusses the Keccak hash function. HMAC and MAC-Keccak are presented in Section III. Different side channel attacks and discussion of different types of power analysis attacks is presented in Section IV. Section V discusses the simulation, attack frameworks, the results and their analysis. Finally, the conclusions and some ideas for future work are presented in Section VI.

II. Keccak Hash Function

The Keccak hash function is based on a sponge construction, in which all of the input is absorbed and then the digest is squeezed out as shown in Figure 1.

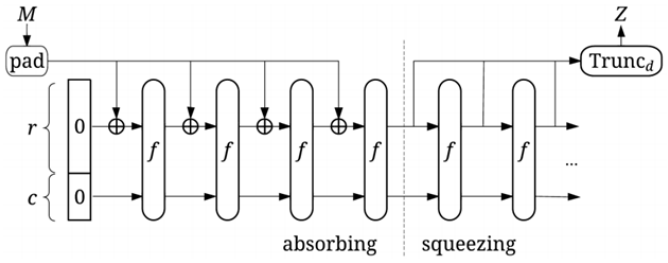

Figure 1. The sponge construction: $Z = \text{Sponge}[f, \text{pad}, r](M, d)$ [1][2].

The internal state size for the Keccak-$p[b, n_r]$ permutation is comprised of b bits, where b is called the width of the permutation and n_r is the number of rounds. The permutation is defined for any $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ and any positive integer n_r. The variable b can be considered to be a $5 \times 5 \times w$ array of bits, where $w = 2^l$, $b = 5 \times 5 \times 2^l$, and $0 \leq l \leq 6$. By default, $l = 6$ and so the internal state has $b = 1600$ bits. The messages are filled into the 3D array state from left to right beginning at the bottom plane and

then proceeding into the upper planes. The rate r defines the number of message bits that the sponge absorbs in each stage. Thus, the bit rate r determines the implementation speed. The capacity c is the number of zero bits that get appended to every r bits of the message to form the input state ($b = r + c$). The trade-off between smaller r with larger c values and larger r with smaller c values is security for speed[3]. For the arbitrary length digest output, the default rate $r = 1024$, capacity $c = 576$ and the internal state size of $b = 1600$ bits. The Keccak function consists of $n_r = 12 + 2l$ rounds of five sequential transformations, thus 24 rounds in SHA-3. A single round of Keccak is defined as

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta(Input).$$

Let $a[x][y][z]$ denote a bit in the three-dimensional array state, $a[x][y]$ a single lane and $a[x]$ a single sheet.

A. Theta Transformation, θ

The θ transformation is responsible for diffusion across the lanes of the state [3]. It is a simple XOR operation with 11 inputs (two columns of the state and one additional bit) and a single bit output.

$$\theta: \ a[x][y][z] \ \leftarrow \ a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1]$$

B. Rho Transformation, ρ

The ρ transformation is a binary rotation over each lane of the state. It is the rotation of the bits of each lane by a specific length known as the offset which depends on the fixed x and y coordinates of the state. For each bit in the lane, the z coordinate is modified by adding the offset modulo the lane size w.

$$\rho: \ a[x][y][z] \ \leftarrow \ a[x][y][z-(t+1)(t+2)/2],$$
with t satisfying $0 \le t < 24$ and
$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in GF}(5)^{2\times 2},$$
or $t = -1$ if $x = y = 0$

C. Pi Transformation, π

The π transformation is designed for disturbing the slice horizontal and vertical alignment. The transformation shuffles every row of lanes to a corresponding column. The π transformation rearranges the positions of the lanes for any slice as follows.

$$\pi: \ a[x][y] \ \leftarrow \ a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

D. Chi Transformation, χ

The χ transformation is a non-linear operation that utilizes XOR, AND, and NOT binary operations. Every bit of the output state is from the XOR between itself, the AND between one neighboring bit and the NOT of another neighboring bit in its row. The output bit is flipped if its two adjacent bits along the x-axis are 0 and 1, in that order.

$$\chi: \ a[x] \ \leftarrow \ a[x] + (a[x+1]+1)a[x+2]$$

E. Iota Transformation, ι

The ι transformation is a linear operation that XORs the bits of the first lane of the bottom plane with a round-dependent constant that depends on the round index, i_r. The other 24 lanes are not affected by the ι transformation. Without ι transformation, the round mapping would be symmetric and all rounds would be mathematically the same.

$$\iota: \ a \ \leftarrow \ a + \text{RoundConstant}[i_r]$$

The standardization of the SHA-3 functions requires the internal state of the Keccak permutation to be 1600 bits; therefore, in the restricted case of $b = 1600$, the Keccak family is denoted by Keccak$[c]$. The SHA-3 function names are nothing more than aliases for different usage instances of Keccak$[c]$, such as SHA3-512(M) = Keccak$[1024]$ $(M||01, 512)$ where $c = 1024$. The choice of c determines the r value. The technical expression of Keccak$[c]$ in terms of the sponge construction using a specific Keccak-p permutation with parameters, M, message and, d, output length is defined in the following equation:

$$\text{Keccak}[c](M, d) = \text{Sponge}[\text{Keccak-}p[1600, 24], \text{pad10*1}, 1600 - c](M, d)$$

III. HMAC and MAC-Keccak

A well known MAC construction is the keyed-hash message authentication code (HMAC) that uses the cryptographic hash function in combination with a secret key [4]. With a secure HMAC implementation, it must be computationally infeasible for an adversary to forge a valid HMAC without the secret key, nor should it be feasible for the attacker to retrieve any information about the key. The HMAC is used to verify both the origin and data integrity of a message. The popularity of HMAC is due to its provable security and efficiency, given the underlying hash function is secure [5]. The HMAC algorithm inputs an arbitrary length message M and a secret key K, and produces a fixed-length digest as follows:

$$\text{HMAC}(K, M) = H((K \oplus opad) \parallel H((K \oplus ipad) \parallel M))$$

where H is a cryptographic hash function, K is a secret key padded to the right with extra zeros until it matches the input block size of the hash function, or the hash of the original key if it's longer than the block size, M is the message to be authenticated, $ipad$ is the inner padding (0x3636...3636)

and $opad$ is the outer padding (0x5c5c...5c5c), both having the length of one block [4]. HMAC is a hash within a hash. The inner hash function takes $(K \oplus ipad)$ which is the same size as the input block, appends the message and hashes it. The outer hash function of the HMAC prepends the digest of the inner hash function with another key variant $(K \oplus opad)$ and produces the HMAC digest. With the FIPS PUB 198 standardization usage of HMACs, there have been no successful attacks on HMAC-SHA-1 because the outer application of the hash function masks the intermediate result of the internal hash. Unlike Merkle-Damgård construction, the output of the sponge constructions only reveal a small part of the final state. Hence, it is believed that Keccak is protected, by design, of such vulnerabilities. The designers of Keccak have recommended to use the direct MAC construction with the Keccak, i.e. MAC-Keccak [1].

$$\text{MAC}(K, M) = H(K \parallel M)$$

The secret key used in an HMAC and MAC is a clear target for adversaries to attack.

IV. Power Analysis Attack

One of the most powerful and best known side channel attacks is the Power Analysis Attack (PAA). It exploits traces of the power consumptions of cryptographic devices to reveal the secret data used in cryptographic computations. When performing PAA, the attacker attempts to identify a relationship between the changing internal state of the cryptographic device and its measured power consumption. In order for this relation to be meaningful to the extraction of the secret data, an intermediate state must be identified that depends upon the secret data to be extracted [6].

To correlate the sensitive cryptographic data with the power consumption different leakage models are used. The most common models used are Hamming Distance (HD), Hamming Weight (HW) and Switching Distance (SD). The Hamming Weight model is the simplest representation of a device's power consumption such that the amount of power consumed is proportional to the number of bits that are logical '1' during an operation. The Hamming Distance model assumes that the number of bit transitions during a cryptographic operation is proportional to power consumption [7]. If a bit is static during an operation, then it is assumed that it will not contribute to the power consumption. The Switching Distance model described by Peeters *et al.* [8] extends the Hamming Distance concept by considering that the transition $0 \rightarrow 1$ may not consume the same amount of power as $1 \rightarrow 0$. In our work, the Hamming Distance leakage model is utilized as part of the Correlation Power Analysis (CPA) attack.

CPA deduces the correct key by using correlation coefficients of statistics. This method was first introduced by Brier *et al.* for DES and AES [9]. In CPA, the attacker computes the correlation between two independent variables most relevant for the power consumption model and the actual device's power consumption denote by X the predicted power calculated in a power model used and by Y the equivalent real power consumption measured when processing the cryptographic operation. The Pearson's correlation coefficient $\rho_{X,Y}$ between X and Y with their expected values μ_X and μ_Y and standard deviations σ_X and σ_Y can be calculated as

$$corr(X, Y) = \rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

.

A. Hardware simulation flow

In the Keccak submission documents to the SHA-3 competition, hardware models were also provided for high-speed, mid-range, and low-area implementations of Keccak [3]. Our research simulates the execution of the Keccak high-speed core, where the simulation framework to produce the power traces of the target ASIC implementation is presented in Figure 2, and is discussed in detail in [10].

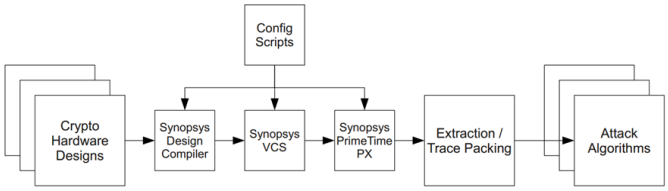

Figure 2. ASIC top-level simulation flow [10].

This design flow utilizes the Synopsys Electronic Design Automation (EDA) tools to compile, synthesize, simulate and perform power estimation of the hardware design. The simulated power extraction process is split into three main steps. First, the hardware design is compiled and synthesized using the Synopsys Design Compiler. After the design is built, the synthesized implementation and its corresponding test bench are simulated using Synopsys VCS. During the simulation, the algorithm is executed multiple times with different input vectors. Finally, the simulation output is processed by Synopsys Primetime PX tool to generate time-based power reports.

B. Software framework

Our software analysis framework supports the CPA attacks on Keccak, using 1-bit, 2-bit, 4-bit, 8-bit and 16-bit selection functions. The central attack manager is the Side Channel Analysis (SCA) module that is responsible for the creation and distribution of work. Based on the number of physical CPU cores available on a given workstation, the SCA module will spawn one or more worker threads, up to the number of available CPU cores. The usage of multiple threads in CPA makes a big difference between completing an attack in a few hours with multiple threads and a few days for a single thread due to the intensive computation for correlation between the power traces and the power model. When launched, the attack framework software is an interactive command-line application with intuitive help menu that allows the user to select the desired attack option along with providing necessary input parameters. The software was built and tested

on Microsoft Windows 7 using Microsoft Visual Studio with .NET Framework Version 4.5.

C. Attack setup and verification

The objective of the power analysis attack on MAC-Keccak is to retrieve the prepended key of each input message. The target high-speed hardware core has an internal state of 1600 bits, rate r of 1024 bits and capacity c of 576 bits. The padded messages generated with the prepended key were chosen to fit into 1 block. Therefore, Figure 3 shows the front view of the state, where the 5 key lanes (320 bits) occupy the bottom plane and 11 lanes (704 bits) of message data with padding occupy the next planes.

With such setup one can exploit the first Keccak round operation, specifically the θ transformation, to extract the key as proposed by Taha *et al.* in [11]. The θ transformation here is computed over two successive phases. The first phase is to calculate the parity of each column of the state, which is called θ_{plane}. The second phase consists of calculating the remaining part of the θ transformation which is the XOR between every bit of the state and the two parity bits of θ_{plane}.

—-

0	0	0	0	0
0x80...01	0	0	0	0
M	M	M	M	M
M	M	M	M	M
K	K	K	K	K

Figure 3. Front view of the Keccak state with 5 key lanes and 11 lanes of message data.

Our initial CPA attack uses 200K power traces and an 8-bit selection function to identify one byte of the key. 256 correlation traces are produced, where each trace corresponds to a single key byte guess as presented in Figure 4. The correct key guess here is 0x07, which clearly stands out among the results, and confirms the feasibility of the attack.

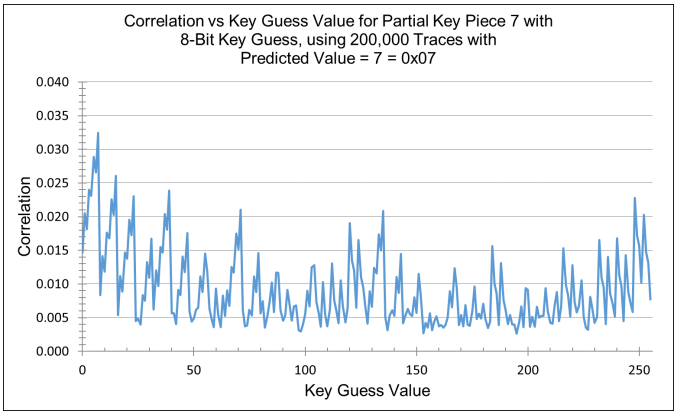

Figure 4. CPA predicted the correct key value for one byte of the whole key.

V. Results and Analysis

Figure 5 shows the success rates of n-bit CPA selection function. The legend of 'SR top n' indicates the target key piece is in the top n ranked correlation results. For example, 'SR top 8' indicates the value of the target key guess is in the top 8 correlation results, assuming the CPA attack application has access to the key and compares with the top 8 correlation results.

Obviously, in a real attack, the CPA application does not have access to the key in order to compute the success rates. The goal here is to see how much MAC-Keccak is susceptible to power analysis attacks. The SR top 1 of n-bit CPA selection function key guess sizes gives a strong clue on how to determine this part of the key. As n-bit CPA selection function key guess size increases, SR top 1 approaches the value 1 with fewer number of traces. It makes sense that 16-bit CPA selection function key guess size requires fewer number of traces to retrieve the key byte successfully because there are 65,536 key guesses to correlate compared to 4-bit CPA selection function key guess size with 16 key guesses to correlate. One can conclude that the larger the n-bit CPA selection function key guess size, the better the signal-to-noise-ratio (SNR), and that it requires fewer number of traces needed to be applied to retrieve the key.

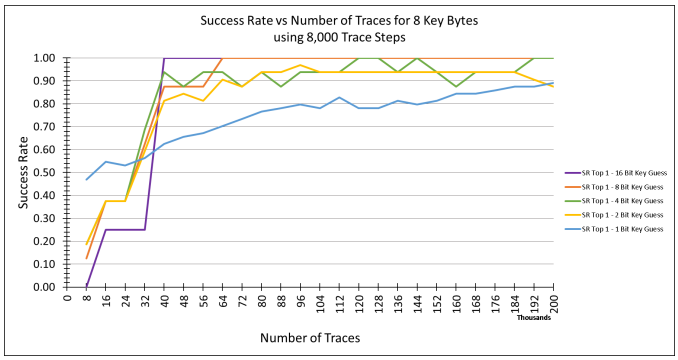

Figure 5. Success rates where the correct key is ranked first according to the correlation results for 1-bit, 2-bit, 4-bit, 8-bit and 16-bit CPA selection function key guess sizes when attacking 8 key bytes using 2 rounds of Keccak.

Figure 6 and Figure 7 show the runtime of n-bit CPA selection function key guess size. It is perhaps not surprising to see smaller n-bit CPA selection function key guess sizes such as the 4-bit CPA selection function key guess size having much better runtime than the 16-bit CPA selection function key guess size. The larger the n-bit CPA selection function key guess size, the more choices for the key guess and hence it is more computation and runtime costly for CPA attacks.

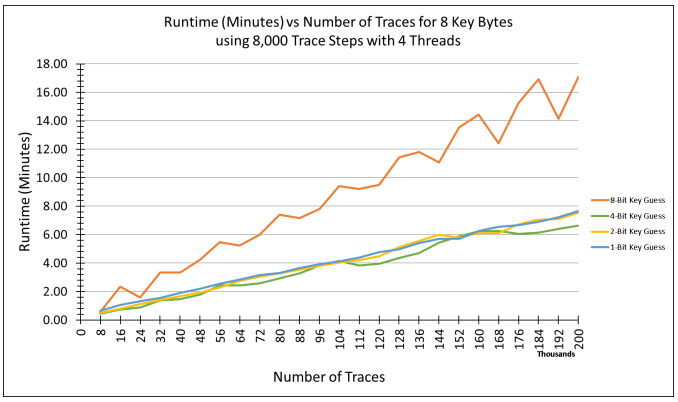

Figure 6. Runtime of 1-bit, 2-bit, 4-bit, and 8-bit CPA selection function key guess sizes using 2 rounds of Keccak.

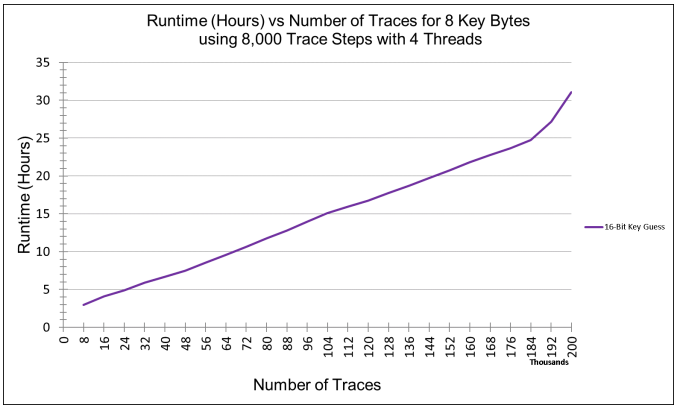

Figure 7. Runtime of 16-bit CPA selection function key guess size using 2 rounds of Keccak.

From analyzing the success rates and runtimes of the n-bit CPA selection function key guess sizes, it is clear the 8-bit CPA selection function key guess size is the best candidate for CPA attack in terms of number of required traces and computation performance time to successfully attack 8 key bytes.

A. Related work

Luo *et al.* in [12] present the attack targeting the θ transformation of the first round of MAC-Keccak implemented on an FPGA to retrieve the secret key, which like in our case has 320 bits on the first plane. The authors discuss the usage of the Hamming Weight power model that targets the θ_{plane}, denoted as power model 1 in their paper, to recover the whole key but with significantly larger number of power traces. In Figure 8, using the Hamming Weight power model takes about 500,000 traces to recover one byte of one key lane with a success rate around 90%, with 8 bytes in each lane. To generate the power traces and apply power model 1 to produce the results in Figure 8, the authors implement the official VHDL code of unmasked Keccak implementation on a SASEBO board which contains a Xilinx Virtex-5 FPGA running at 12MHz, and use an Agilent MSOX4104A oscilloscope to capture the measured power traces. The official Keccak VHDL code that is used is the high-speed core that has 1600 internal bits, with the rate $r = 1024$ bits and capacity $c = 576$ bits.

From the plot in Figure 8, applying 500K traces, Luo *et al.* are able to recover one key byte with SR top 1 of around 90% using the 8-bit CPA selection function key guess size. This is understandable since the traces came from a physical device that is susceptible to measurement equipment and surrounding noise. Compared to the data result of the success rate using simulated traces depicted in Figure 5, usage of 8-bit and 16-bit CPA selection function key guess size attacks are able to retrieve not only 1 key byte but the whole key lane, i.e. 8 bytes, with a success rate of 1 using only 64K traces.

Figure 5 shows that less than 40,000 traces are needed with 16 bit selection function. What this means is that using simulated traces for studying power models to target Keccak operation is more effective than using physical traces without the interference of noise such as environmental noise, measurement device noise, etc. Therefore, one can say using circuit simulation significantly reduces the complexity of mounting a power attack, provides quicker feedback during the implementation and study of a cryptographic device, and that ultimately reduces the cost of testing and experimentation.

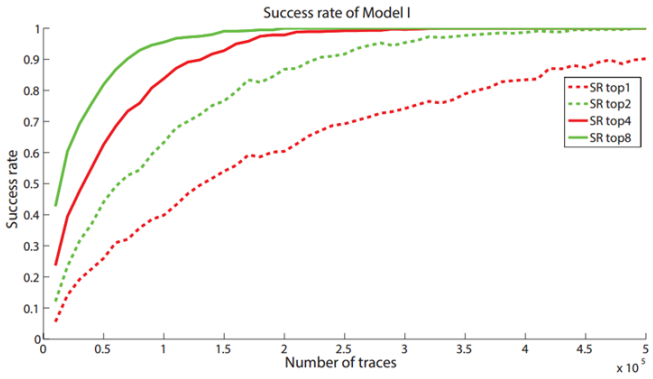

Figure 8. Success rate based on model 1 [12].

VI. Conclusion

This work took an in-depth look at the application of the power analysis attacks that use n-bit CPA selection function key guess technique to extract the key from the MAC-Keccak using simulated power traces.

From the generation and application of the simulated power traces, this research successfully attacked unprotected MAC-Keccak with keys of at most 40 bytes and extracted the keys by using 1-bit, 2-bit, 4-bit, 8-bit and 16-bit CPA selection function key guess size attacks from the exploitation and targeting of the Keccak first round θ function that absorbed the "key||message".

It was observed that the larger the selection function key guess size used, the better the signal-to-noise-ratio (SNR), therefore requiring fewer traces to retrieve the key but at the price of higher computation time. With 16-bit CPA selection function key guess size attack using 2 rounds of Keccak, 50K

traces and 7 hours of computation were required for 100% key extraction of a single key lane of 8 bytes. Compared to larger selection function key guess size, smaller selection function key guess size has lower SNR that requires higher number of applied traces for successful key extraction and utilizes less computational time due to smaller number of guesses to iterate through. With 4-bit CPA selection function key guess size attack using 2 rounds of Keccak, it requires 200K traces and 7 minutes of computation for 100% key extraction of a single key lane.

There are several avenues to extend this research work. The current work applies the side channel vulnerability assessment of the Keccak high-speed core hardware implementation using gate-level circuit-simulated power consumption waveforms from Synopsys EDA tools with a 130-nm CMOS standard cell library. The success of the CPA attack depends on how accurate the power model, i.e. using Hamming weight versus Hamming distance, is for the target operation to correlate with the power traces. As mentioned earlier the total power consumption of the CMOS circuit is the sum of the static power and dynamic power. Potential future work is to investigate the effect of using smaller technology sizes such as 45nm, 16nm or 7nm CMOS standard cell library and see how that will impact the CPA attack. As the technology gets smaller in size, the static power is exponentially increasing whereas the dynamic power is linearly decreasing. It will be interesting to see if the dynamic power for these smaller size technologies is overshadowed by the constant larger static power, that is using and realizing them in existing power leakage models like Hamming weight or Hamming distance in CPA attacks to be in any way effective as with larger technology sizes.

Another possibility for future work is to investigate other potential targets of the Keccak operations for better power correlations. This thesis only focused on the θ_{plane} calculations of the first round θ operation that performed column XOR operations. Since the θ operation is a linear function, it does not produce high correlation values such that this thesis is only able to extract the key from MAC-Keccak on the first plane, i.e. < 40 bytes. From past experience with attacking the non-linear operation such as the S-Box of AES that gave the strongest correlation values, one would want to target the non-linear operation of Keccak. In this case, future work should look at the first round χ non-linear operation for high correlation values. Besides targeting the first round χ non-linear operation, another target is to examine the output of the first round as a result of the five Keccak operations. This way the output of the first round is known to have gone through the non-linear operation and, hopefully, will provide better correlation values to attack key bytes beyond the first plane.

Since this work shows MAC-Keccak is susceptible to power analysis attack using simulated power traces, one possible future work is to implement software countermeasure such as using blinding technique e.g. masking the inputs and outputs of elementary operations.

Finally, a potential future work is to investigate and to take advantage of the SR top 4 of 8-bit CPA selection function key guess size and SR top 1 of 16-bit CPA selection function key guess to narrow down the correct key effectively. The usage of the SR top 4 of 8-bit CPA selection function key guess size is to narrow down the key choices to take advantage of the fast runtime, and the usaage of the SR top 1 of 16-bit CPA selection function key guess is to take advantage of using the least number of power traces with respect to other n-bit CPA selection function key guess.

REFERENCES

[1] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Cryptographic sponge functions." [Online]. Available: http://sponge.noekeon.org/CSF-0.1.pdf, 2011.

[2] NIST, "DRAFT FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," *Federal Information Processing Standards*, 2014.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The Keccak reference, Version 3.0." [Online]. Available: http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January 2011.

[4] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Advances in Cryptology CRYPTO 96*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 1–15, Springer Berlin, Heidelberg, 1996.

[5] O. Benoit and T. Peyrin, "Side-channel Analysis of Six SHA-3 Candidates," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, vol. 6225 of *Lecture Notes in Computer Science*, pp. 140–157, Springer Berlin, Heidelberg, 2010.

[6] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'99, pp. 388–397, London, UK: Springer-Verlag, 1999.

[7] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards," in *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, pp. 17–27, Berkeley, CA, USA: USENIX Association, 1999.

[8] E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons," in *Integration, the VLSI Journal*, vol. 40, pp. 52–60, Spring 2007.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 135–152, 2004.

[10] K. S. Jr. and M. Lukowiak, "Methodology for Simulated Power Analysis Attacks on AES," in *Military Communications Conference, 2010-MILCOM 2010*, pp. 1292–1297, IEEE, 2010.

[11] M. Taha and P. Schaumont, "Differential Power Analysis of MAC-Keccak at Any Key-Length," in *8th International Workshop on Security (IWSEC2013)*, pp. 68–82, 2013.

[12] P. Luo, Y. Fei, X. Fang, A. A. Ding, M. Leeser, and D. R. Kaeli, "Power Analysis Attack on Hardware Implementation of MAC-Keccak on FPGAs," in *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–7, 2014.

[13] M. Taha and P. Schaumont, "Side-Channel Analysis of MAC-Keccak," in *Signal Processing Workshop on Higher-Order Statistics (SPWHOS)*, pp. 125–130, 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2013.

[14] J. P. Uyemura, *Introduction to VLSI Circuits and Systems*. John Wiley & Sons, second ed., 2002.

[15] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, and R. V. Keer, "The Keccak implementation overview, Version 3.2." [Online]. Available: http://keccak.noekeon.org/Keccak-implementation-3.2.pdf, May 2012.