International Journal of Computing and Network Technology
@ 2014: Scientific Publishing Center, University of Bahrain

Design and Implementation of Hiding Techniques to Obfuscate Against Side-Channel Attacks on AES

Todd R. Andel[1,*], Austin Fritzke[2], Jeffrey W. Humphries[3] and J. Todd McDonald[1]

[1] *School of Computing, University of South Alabama, Mobile, AL 36688 (USA)*
[2] *Department of Electrical and Computer Engineering, Air Force Institute of Technology, Dayton, OH 45433 (USA)*
[3] *Department of Computer Science, Covenant College, Lookout Mountain, GA 30750 (USA)*

Received 26 Dec. 2013, Revised 22 Feb. 2014, Accepted 25 Feb. 2014, Published 1 May 2014

Abstract: While AES is computationally secure, it is not without weakness. Side-channel attacks on AES hardware implementations can reveal its secret key. Because of this vulnerability, countermeasures to side-channel attacks are crucial to data security. This paper assesses a current design and proposes a new design for securing AES, along with evaluating their implementation onto field programmable gate arrays. Both countermeasures were successfully implemented, and the data remained secure against common side channel attacks. Results indicate successful obfuscation of the secret key over the original AES algorithm.

Keywords: Side-Channels, FPGA, DPA, AES, Countermeasure

1. Introduction

While the Advanced Encryption Standard (AES) is known to be computationally secure, it is vulnerable to side-channel attacks against hardware implementations. One of the more common hardware attacks is Differential Power Analysis (DPA) introduced by Kocher et al. [1]. DPA is performed by sending the AES algorithm known plaintext or receiving known ciphertext. During the process, power and/or electromagnetic traces from the encryption device are recorded and are correlated to the known data and key guesses. To counteract this vulnerability, DPA countermeasures attempt to create independence between the data and the power traces. The two main countermeasure types are hiding and masking [2]. Hiding attempts to cover the message with noise from the circuit or use other means such as timing variance to create independence between the data and the traces. Masking conceals the data by adding or multiplying random numbers to the data during the encryption process, and then removing the mask(s) before the output is determined.

This paper focuses on two hiding techniques to obfuscate the data from an attacker. The first countermeasure focuses on preventing traces from being aligned by randomizing the clock frequency of the circuit, therefore hindering DPA attempts by an attacker. The idea was introduced by Zafar et al. [3] and the concept was simulated using simulation software such as MentorGraphics ModelSim. This research differs in that it provides a realized implementation outside of previous theoretical simulated designs. The second countermeasure focuses on a hiding technique called bit-balancing. The current bit-balancing trend for AES is dual-rail logic, which concurrently evaluates the data and the inverse of the data at the gate level [2]. This research develops and implements a countermeasure technique that provides bit-balancing at the system level, therefore greatly decreasing the complexity and size of the circuit compared to AES with dual-rail logic gates. The countermeasure described in this research is similar to [4] in many key areas, but differs in both implementation and design.

The two countermeasures were chosen for their potential effectiveness for obfuscation of the circuit while minimizing costs to the user. Previous obfuscation techniques, such as masking, have shown to be effective in providing independence between the data and power traces, typically requiring 2x to 10x increase in circuit size and 3x to 90x decrease in circuit speed [5]. The goal of this research is to provide effective defense against power attacks on AES while minimizing costs in circuit area, circuit speed, and circuit power consumption.

This paper is organized as follows: an overview of the AES algorithm and common DPA techniques is given in Section 2. Section 3 provides results of DPA performed on hardware AES for the purpose of both developing a baseline for comparison and supporting the need for countermeasures. Section 4 discusses

E-mail: tandel@southalabama.edu

hiding countermeasures designed to misalign power traces and presents an implemented design of a randomized clock. Section 5 looks at dual-rail logic and bit-balancing countermeasures, as well as provides an effective new system-level bit-balancing implementation followed by conclusions in Section 6.

2. Background of AES and DPA

In order to put the remainder of the paper in context, a brief overview of the AES algorithm and DPA attacks is Provided**.**

2.1 AES Background

The AES algorithm is a form of the Rijndael algorithm and is the current standard for symmetric-key cryptography [6]. It is a symmetric block cipher that processes 128-bit data blocks and can operate on keys with lengths of 128, 192, or 256 bits. Each data block consists of 16 bytes and is four rows deep and four columns wide. Each individual block is processed through 10, 12, or 14 rounds, depending on the key size. For this paper, when referring to the AES algorithm, we assume a key length of 128 bits using a total of 10 rounds. A pictorial overview of AES is given in Figure 1. Each round consists of four individual transformations of the data: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The algorithm begins by adding the RoundKey and continues into the rounds. The last round does not contain the MixColumns function. Of the four transformations, only the SubBytes function is non-linear, and is therefore subject to great interest for power attacks [7]. SubBytes is a table lookup from a substitution box (SBOX), which simply substitutes a byte out for respective byte input. Also, as seen in Figure 1, the key is used to create round keys derived from the key schedule. More detail on the AES algorithm can be found from the FIPS publication [6].

2.2 DPA Background

Differential Power Analysis is a powerful attack that uses statistical analysis and error correction techniques to extract information correlated to the secret keys [1]. There are two main phases of a DPA attack: data collection and data analysis. Data collection is performed by sampling a circuit's power consumption or electromagnetic (EM) radiation during operation of the cryptographic algorithm. Data analysis is the more involved phase of DPA and often involves correlation of the collected power traces to a hypothetical power model created by the attacker. There are several models used to create hypothetical intermediate values used for correlation between the data and the power traces. The most common models are the bit model, Hamming Weight (HW) model, Hamming Distance (HD) model, and the Zero-Value (ZV) model [2].

The bit model looks at one bit of an intermediate value at a time and separates the collected power traces into two groups. The first group is whether the intermediate value should be a 0 for the current key guess, and the second group is if the intermediate value should be a 1. While straightforward, the bit model is limited in scope and is generally not as effective as the HW, HD, or ZV models [2].

The HW model counts the number of '1's in a binary number and assigns that to its value. For example, 00011000 would have a HW of two. The HW model is effective in modeling power traces because of the difference in power drawn from a circuit evaluating a '1' compared to a '0'.

The HD model is even more effective than the HW model, but requires knowledge of either the preceding or following intermediate value, as well as the current hypothetical intermediate value. The HD is the difference in '1's between two binary numbers. For example, if a number were to change from 00011000 to 11111111, the HD would be six. There is usually a significant change in power when a device in a circuit changes its output value from '0' to '1' or vice versa. In this way, the HD model can be a very effective model for hypothetical power consumption.

Finally, the ZV model assumes that the power consumption for the data value 0 is lower than the power consumption for all other values [2]. The data within a ZV model is set as either a 0 or 1 depending on whether the data evaluated is equal to 0.

Regardless of the chosen model, they all require an intermediate value to be calculated in order to operate. When evaluating only one byte at a time, it is beneficial to avoid the MixColumns function. Otherwise, more than one byte of the intermediate value (and therefore more than one byte for the key guess) would have to be assumed [6]. This approach would greatly increase the complexity of the DPA and exponentially increase the processing time. Because of this limitation, intermediate values before the first MixColumns operation or after the last MixColumns operation are attacked for DPA on AES.

3. DPA Results on Hardware AES

For a baseline design, a simple iterative Verilog version of AES is used [8], where each round is performed sequentially before the next byte is processed. Power traces are collected using the Riscure Inspector side channel test suite, including an EM high sensitivity probe, an EM Probe Station, and the Riscure Inspector software. Power traces are measured using a Lecroy WavePro 725Zi oscilloscope, and the AES design is loaded onto a Xilix Virtex-5 FX FPGA

evaluation board. A visual overview of the experimental setup is given in Figure 2.

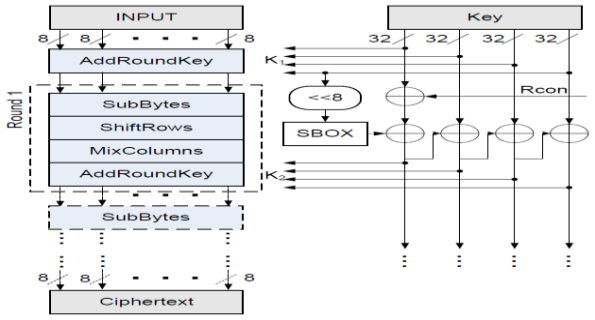


Fig. 1: The AES Algorithm [4]

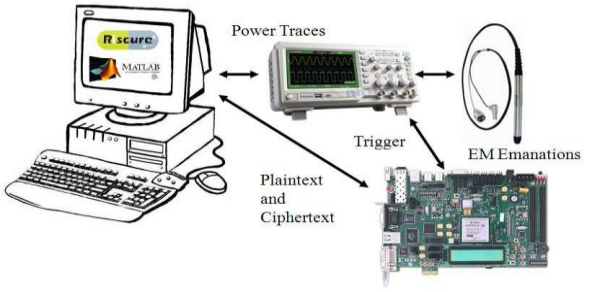


Fig. 2: Experimental Setup

The number of traces collected for each AES system tested in this research is varied between 1,000 and 3,000,000. The frequency of the bus is set to 100 MHz. To minimize unwanted noise, the traces are filtered using a bandpass filter between 90 and 210 MHz. Each trace uses the same key given in Equation 1, and the input data is randomized for every trace using Java's randomize function. An example of a power trace is given in Figure 3, with AES encryption occurring between 1.1 and 1.7 μsec.

```
Key = 0001 0203 0405 0607 0809 0A0B
0C0D 0E0F (1)
Last Round Key = 1311 1D7F E394 4A17
F307 A78B 4D2B 30C5
```

To effectively attack the AES algorithm, several different models were used on both the first and last round of the algorithm. Along with bit and ZV models, HW and HD models were used to correlate the traces to the data. During initial testing, the bit and ZV models proved less effective than the HW and HD attacks, and therefore were not considered in the final evaluation of correlation effectiveness. When performing HW and HD attacks on the circuit, seven different intermediate values or intermediate value pairs were chosen:

1.HW: First Round Before SubBytes
2.HW: First Round After SubBytes
3.HW: Last Round Before SubBytes
4.HW: Last Round After SubBytes
5.HD: First Round Between Values Before and After SubBytes
6.HD: Last Round Between Values Before and After SubBytes
7.HD: Last Round Between Values Before SubBytes and Ciphertext Output

When evaluating the last round, the original key cannot be used, but instead the last round key as given in Equation 1. This is due to the fact that each round adds its corresponding round key, and the round key is only equal to the original key for the first round. Subsequent rounds use keys generated during the key expansion process of AES. The correlation is performed using MATLAB software. After evaluating each modeling technique on the different intermediate values, it appears the HD on the last round between the intermediate value before the SubBytes and the ciphertext (choice 7) provides the highest correlation. The results on an attack on the first byte are given in Figure 4. The correlation for the correct key guess (19) is given by the solid blue line, and the attack is performed while the number of traces is varied. DPA is successfully performed when the number of traces is greater than or equal to 500,000 traces, as indicated when the correlation of the correct guess exceeds the peak false positive. The attack on the unprotected AES system gives a process baseline to compare to AES implementations with added countermeasures. The goal of this research is to effectively eliminate correlations such as seen in Figure 4 with minimal cost in added execution time and circuit area. The first countermeasure increases the challenge of aligning traces, therefore greatly increasing the difficulty of processing the power traces after collection. The second countermeasure effectively flattens the power signatures of the intermediate values in relation to the Hamming Weight and Hamming Distance of the bits.

4. Trace Alignment Countermeasures

In order to perform an effective DPA attack, trace alignment must be accomplished for correlation between the power traces and the power model. Without alignment, each power trace/expected intermediate value pair would have to be evaluated individually for each trace.

4.1 Existing Approaches

A relatively popular method for accomplishing misalignment is to randomly insert delay cells into the

algorithm. In general, inserting delay cells effectively misaligns the traces and hinders the effectiveness

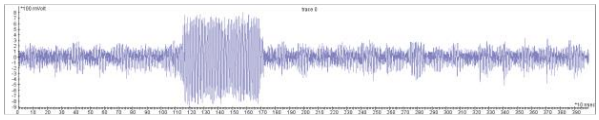


Fig. 3: Power Trace of Hardware AES

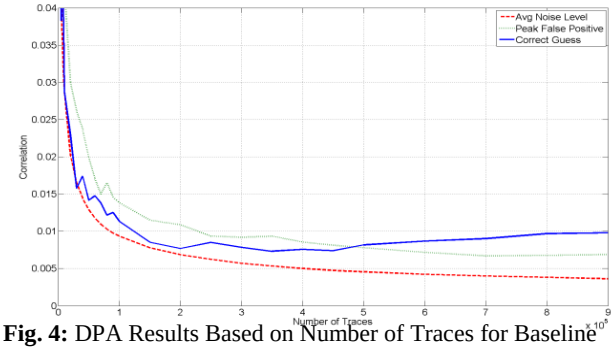


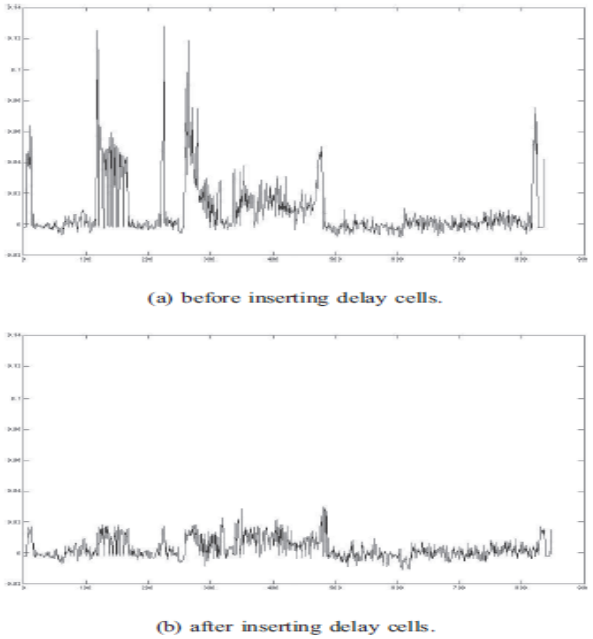
Fig. 4: DPA Results Based on Number of Traces for Baseline AES

of a DPA attack [9,10]. Figure 5 shows how misalignment can effectively minimize correlation between the traces and the data.

However, as stated in [11], there are methods to pinpoint and eliminate the delay cells within the power trace, therefore eliminating the misalignment attempts. In order to prevent this re-alignment, Zafar et al. introduced in [11] a clock variance method to accomplish misalignment with the same results as delay insertion, but without the risk of the delays being manually removed during DPA processing. This idea is further improved in [3] by randomizing each clock cycle instead of changing the clock frequency after a specified period of time. By using this approach, trace alignment is virtually impossible to accomplish using traditional methods. The downside of randomizing every clock cycle is an increase in the delay of the system. If the frequency of the clock is varied from 100% to 50% (from 1x to 2x slower), the average frequency will be 75% of the original (1.5x slower on average). This degradation is significant when comparing to a handful of manually inserted single clockcycle delay cells added to the overall system runtime.

4.2 Randomized Clock Implementation

In order to randomize the clock in this research, a 16-bit linear feedback shift register (LFSR) is polled at the end of each clock cycle to pseudo-randomly choose between four clock cycle lengths. The LFSR is seeded with a pseudo-random value at the beginning of each encryption process to ensure variance between algorithm runs. Optimally, the clock should vary between 100% and 50% of the original frequency. However, for this research, the clock varies between 33% and 16.7%. Initially, the idea was to simply increase the clock frequency on the Virtex-5 bus by 300%, but the clock was already set near its maximum frequency (100 MHz with a limit of 125 MHz). An external clock could be added to the system, but that is out of the current scope of the project. The randomized clock design unique to this research is given in Figure 6. Note that the clock in Figure 6 represents the clock period. Therefore, the clock generator effectively slows the clock rate by the indicated amounts.



(a) before inserting delay cells.

(b) after inserting delay cells.

Fig. 5: Correlation Before and After Delay Cell Insertion [9]

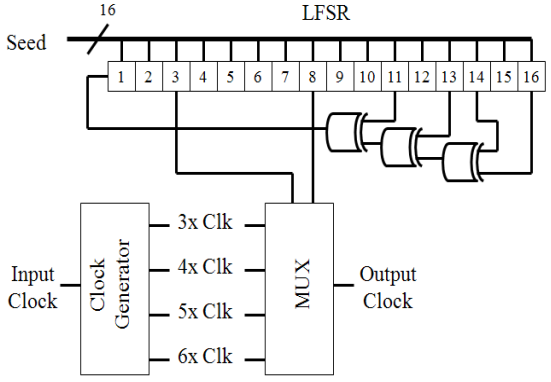


Fig. 6: Random Clock Countermeasure Module

A power trace of the algorithm running on a randomized clock is given in Figure 7. It is important to notice the variation when compared to Figure 3. In fact, the power signature varies greatly for each iteration when there is a randomized clock. In order to determine effectiveness in obfuscation of the circuit, the same DPA attack is performed on the circuit with a

randomized clock as performed on the original circuit. The key guess results for byte 1 are shown in Figure 8

As expected, there is no apparent correlation for the correct key guess. The main difficulty in performing DPA is aligning the traces of the AES algorithm using a randomized clock frequency. Riscure Inspector software includes several advanced alignment algorithms, however, they did not provide correct alignment against the randomized clock countermeasure.

By randomizing the clock, the circuit is effectively protected from common DPA attacks by preventing alignment of the traces. The total area of the circuit changes only minimally to add the random clock generator. However, the runtime of the circuit averages to 4.5x longer than the original, or an average 25% of the original frequency. Optimally, if an external clock was added to the system, the runtime of the circuit could be minimized to 1.5x longer than the original, or an average 75% of the original frequency.

5. Dual-Rail and Bit-Balancing Countermeasures

One of the more popular hiding countermeasures is the attempt to flatten the power signature of all components directly within the circuit's hardware for all values of data. This approach commonly utilizes dual-rail or bit-balancing logic as countermeasures.

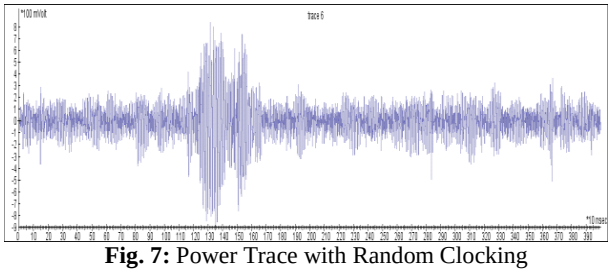


Fig. 7: Power Trace with Random Clocking

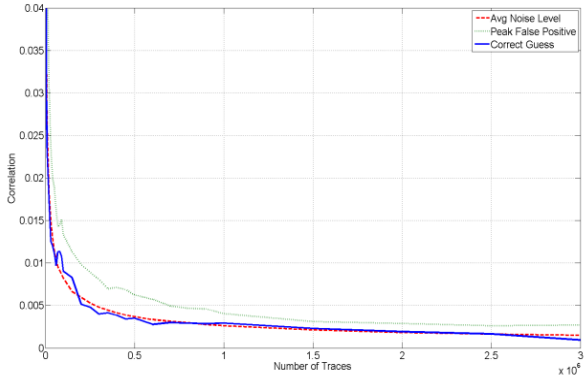


Fig. 8: DPA Results Based on Number of Traces for Randoml Clocked AES

5.1 Existing Approaches

An effective method is performed at the cell level using dual-rail precharge (DRP) logic blocks [2]. The idea behind DRP logic is to create logic cells that make power consumption constant during each clock cycle. Every input and output into a cell is paired with its inverse and therefore a constant balance of '0's and '1's are entering and exiting the cell at all times. Figure 9 gives an example of a dual-rail AND gate.

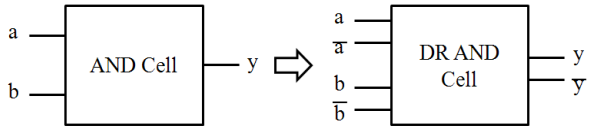


Fig. 9: Dual-Rail AND Gate

There have been several dual-rail designs attempted in order to protect cryptographic algorithms, and in general are quite effective [12,13,14]. However, one major problem with dual-rail systems is the increase in circuit area and associated decrease in speed. Designs such as [12] increase circuit area by 4.5x and increase runtime by nearly double. While dual-rail logic is effective in minimizing the effectiveness of HW and HD attacks, its complexity does not come without a price.

Similar to dual-rail logic, system-level bit-balancing attempts to balance the Hamming Weight for every intermediate value. Attempted by [4], system-level bit-balancing runs two concurrent cryptographic algorithms. The first algorithm performs as expected and delivers the correct output data. The second algorithm processes the inverse of the data and produces inverse output data. When evaluated as a whole, the Hamming Weight of the AES system remains constant during the entire encryption process. The challenge is combining the two circuits in such a way that an attacker cannot differentiate the power emanating from the two separate algorithms.

5.2 System Level Bit-Balancing Design

In addition to the randomized clock, this research develops a system level bit-balancing design to further obfuscate the circuit against HW and HD attacks. The bit-balancing design is similar to [4], but differs in several key areas. First, the key remains unchanged as opposed to [4]. Second, the key schedule is left untouched and the round keys remain the same. Third,

the input data is inverted before entering the AES algorithm, unlike the design in [4]. The design for the inverted circuit is given in Figure 10. To begin, the input data is inverted before entering the system. Due to their linearity, the AddRoundKey, ShiftRows, and MixColumns components of the AES algorithm retain the inversion of the data (inverted input = inverted output from the components). However, the main design challenge comes from the non linear SubBytes component. Finally, this design includes HD resistance, while [4] does not.

In order for the output data from the SBOX to be inverted when provided an inverted input, it is helpful to look at how the SBOX handles the data. For the specific AES algorithm used in this research, a lookup table is used for the SubBytes function. The SBOX lookup table is given in Table 1 [6].

Table 1: SBOX [6]

xy	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cd	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
C	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
F	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Each cell is indexed by the input value. For example, an input of 0x03 would output 0x7B. Because the output needs to be inverted, each value within the SBOX is replaced with its inverted value (0x7B would be replaced with 0x84) for the inverted AES algorithm. However, one more step must be taken to ensure an inverted output. Because the input data will be inverted, indexing must also be inverted. Therefore, what was in the top left cell must now be moved to the bottom right, etc. Once this index "rotation" is accomplished, the new SBOX is ready to output inverted data given an inverted input. The new SBOX for the inverted circuit is shown in Table 2. Note that 0x84 (the inverse of 0x7B) is now indexed by 0xFC, the inverse of the original index 0x03.

Simply inverting all the intermediate values only protects against HW attacks and not HD attacks. This is a problem, especially since HD attacks are commonplace and relatively easy to perform. The system-level bit-balancing design for this research also differentiates itself by including features to resist HD attacks along with its HW resistance. In order to protect against HD attacks, gate level pre-charging has been used in the past [2,12,13]. However, this research uses system-level bit-balancing, so system-level pre-charging is needed. To minimize the vulnerability from HD attacks, 10 buffer cycles are added between the rounds to clear the intermediate values within the rounds. This system-level precharging is accomplished by sending a 0 as the input data and key into the round. This buffering allows for the
HW bit-balancing to effectively prevent against HD attacks as well.

Of course, the output of the rounds must be stored between clock cycles while the system is clearing the intermediate values. This register storage introduces a potential vulnerability. To mitigate the potential for an attacker to perform HD correlation on the registers storing the intermediate values between rounds, an extra feature is added to the design. An LFSR is connected to a multiplexer, and the outputs of the rounds are stored in one of four locations randomly chosen by the LFSR. In this way, there is only a 25% chance of the HD recorded by the EM probe matching the data. Therefore, the HD leakage is effectively minimized. Depending on security requirements the number of possible register locations could be increased, but at a cost of required circuit area and speed. A graphical view of the correlation between the traces and power models is given in Figure 11. As expected, the system-level bit-balancing design is resistant to common DPA attacks with up to three million or more traces collected.

For the bit-balancing design, the added 10 clock cycles adds minimal delay to the system, while the area and power consumed by the countermeasure are around triple that of the unprotected design. These costs are still desirable over comparable dual-rail logic designs. The system-level bit-balancing design is resistant to both HW and HD attacks, and is implemented as one interspersed module on the chip - making differentiation of the two halves of the algorithm significantly difficult.

Table 2: Inverted Rotated SBOX

xy	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	e9	44	ab	4f	f0	d2	66	be	97	bd	19	40	f2	76	5e	73
1	20	d7	aa	31	16	78	e1	64	6b	71	26	96	ee	67	07	1e
2	61	e2	3e	79	46	a8	ca	9e	f1	09	fc	b7	99	4a	c1	8f
3	75	74	42	b4	e0	8b	22	17	39	4b	59	e3	d1	da	87	45
4	f7	51	85	9a	15	0b	a9	93	56	b1	2a	72	92	c8	37	18
5	86	1b	6a	6e	9d	53	2c	3d	a3	db	f9	b6	f5	c5	cd	1f
6	24	f4	a1	21	eb	47	11	b9	77	6f	d5	dd	23	b0	7e	9f
7	8c	e6	a2	9b	c2	81	58	3b	e8	bb	68	a0	13	ec	f3	32
8	2d	0c	00	ef	de	25	49	43	0a	c7	62	6d	70	bf	5c	ae
9	57	60	c3	af	80	fd	06	ba	7a	cc	b2	bc	04	55	10	2f
A	30	a7	b3	b5	c6	41	34	95	a4	4e	03	df	12	ff	2e	ac
B	7b	d0	1c	d6	4c	29	c4	ad	5f	a5	91	e4	e5	d3	7c	f6
C	8a	4d	d8	14	1d	7f	ed	f8	65	fa	69	e7	3c	dc	38	fb
D	ea	ce	27	8e	0e	1a	5a	cb	33	08	c0	c9	d9	6c	02	48
E	3f	8d	5b	63	50	5d	2b	52	0f	b8	a6	05	82	36	7d	35
F	89	54	28	01	d4	98	fe	cf	3a	90	94	0d	84	88	83	9c

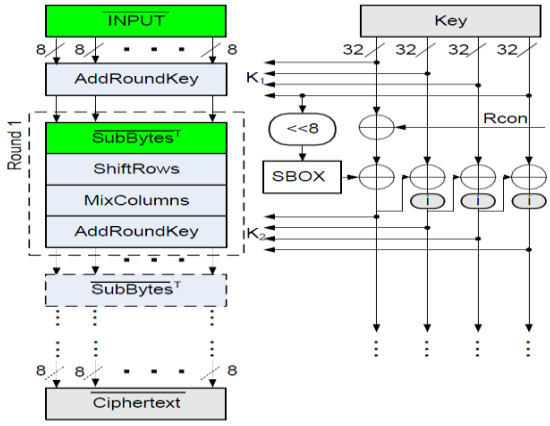


Fig. 10: System Level Bit Balancing Design

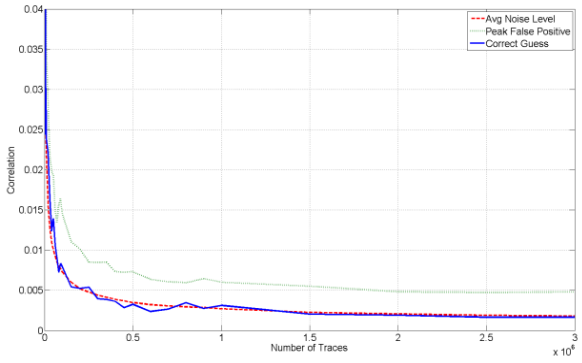


Fig. 11: Max Correlation of all 256 Key Guesses for Bit-Balanced AES

6. Conclusion

The countermeasures developed and tested in this research are effective in obfuscating the key and intermediate data of the AES algorithm from standard DPA attacks. Those attacks include correlation with bit, Hamming Weight, Hamming Distance, and Zero-Value models used to determine the secret key. The randomized clock prevents an attacker from easily aligning traces, a step necessary to perform DPA. The second countermeasure, system-level bit-balancing, was designed with HW and HD attacks in mind, and is implemented on a single chip. Of course, there are added costs to the user in terms of circuit delay, area, and power consumed, but all costs are within reasonable levels compared to similar hiding countermeasures. The results are summarized in Table 3. The DPA resistance is given in number of traces needed to attack the circuit. However, it is important to note that the randomized clock and bit-balancing countermeasures were never successfully attacked, therefore concluding that over three million traces would be needed to eventually attack the circuits.

Table 3: Experimental Results for all Designs

Metric	Baseline	Random Clock	Bit-Balancing
DPA Resistance	500k Traces	>3,000k Traces	>3,000k Traces
Norm. DPA Resistance	1x	>6x	>6x
Circuit Area	1x	0.95x	3.15x
Power Consumed	1x	1x	3.43x
Circuit Delay	1x	3x - 6x	1.77x

Because of the relative independence between the two countermeasures, combining the randomized clock and system-level bit-balancing design would only add to the side-channel attack resistance of the circuit, and should be considered for maximizing obfuscation. This research provides contributions for the protection of information processed using hardware AES. The results and analysis indicate successful demonstration of the goals of this research, mainly to obfuscate sensitive data against side-channel attacks. Not only was a design developed and presented, but implementation and real-world testing were performed using an FPGA system. It is hypothesized that results would be similar in customized application specific integrated circuit (ASIC) designs.

Acknowledgement

This material is based in part upon work supported by the National Science Foundation under Grant No. 1305369

References

[1] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In M. Wiener, editor, *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.

[2] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[3] Y. Zafar, Jihan Park, and Dongsoo Har. Random clocking induced DPA attack immunity in FPGAs. In *Industrial Technology (ICIT), 2010 IEEE International Conference on*, pp. 1068–1070, 2010.

[4] J.A. Ambrose, S. Parameswaran, and A. Ignjatovic. MUTEAES: A Multiprocessor Architecture to Prevent Power Analysis Based Side Channel Attack of the AES Algorithm. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pp. 678 –684, nov. 2008.

[5] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and Franois-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems*,

CHES 2010, volume 6225 of *Lecture Notes in Computer Science*, pp. 413–427. Springer Berlin Heidelberg, 2010.
[6] NIST. Announcing the Advance Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001.
[7] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pp. 413–423. Springer Berlin Heidelberg, 2005.
[8] Akashi Satoh. AES Encryption/Decryption Macro. Tohoku, University [Online] (http://www.aoki.ecei.tohoku.ac.jp/crypto/), 2007.
[9] S. Soydan. Analyzing the DPA Leakage of the Masked Sbox via Digital Simulation and Reducing the Leakage by Inserting Delay Cells. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pp. 221 –227, july 2010.
[10] Yingxi Lu, M.P. O'Neill, and J.V. McCanny. FPGA implementation and analysis of random delay insertion countermeasure against DPA. In *ICECE Technology, 2008. FPT 2008. International Conference on*, pp. 201–208, 2008.
[11] Yousaf. Zafar, , and Dongsoo Har. A Novel Countermeasure to Resist Side Channel Attacks on FPGA Implementations. *International Journal On Advances in Security*, vol 2, issue 1, pp. 1–7, June 2009.
[12] Thomas Popp and Stefan Mangard. Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In JosyulaR. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pp. 172–186. Springer Berlin Heidelberg, 2005.
[13] Zhimin Chen and Yujie Zhou. Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems-CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pp. 242–254. Springer Berlin Heidelberg, 2006.
[14] D. Sokolov, J.Murphy, A. Bystrov, and A. Yakovlev. Design and analysis of dual-rail circuits for security applications.
Computers, IEEE Transactions on, vol. 54, issue 4, pp. 449–460, 2005.

Todd Andel is an Associate Professor at the University of South Alabama's School of Computing. Dr. Andel received his Ph.D. in Computer Science from Florida State University in 2007, his M.S. in Computer Engineering from the Air Force Institute of Technology in 2002, and his B.S. in Computer Engineering from the University of Central Florida in 1998. His research interests include computer and information security, side-channel analysis, hardware/software partitioning, network security protocols, and formal methods.

Austin Fritzke received his B.S. in Electrical Engineering from the U.S. Air Force Academy in 2010 and a M.S. in Electrical Engineering from the Air Force Institute of Technology in 2012. His research interests include microcontrollers, control systems, computer architecture, VLSI, side-channel analysis, and hardware encryption.

Jeffrey Humphries is an Associate Professor of Computer Science in the Department of Computer Science at Covenant College. Dr. Humphries received a B.S. in Computer Science from the U.S. Air Force Academy, a M.S. degree in Computer Science from Georgia Institute of Technology, and a Ph.D. in Computer Science from Texas A&M University in 2001. His research interests include cryptography, computer/network security, information assurance, cyber operations, and software protection.

Jeffrey "Todd" McDonald is an Associate Professor in the School of Computing at the University of South Alabama. Dr. McDonald received his Ph.D. in Computer Science from Florida State University in 2006, his M.S. in Computer Engineering from the Air Force Institute of Technology in 2000, and his B.S. in Computer Science from the U.S. Air Force Academy in 1990. His research interests include program protection and exploitation, secure software engineering, and information assurance.