Influence of CAD Routing Algorithms on Cryptographic Side Channel Attacks

A THESIS

Submitted by

B AKHIL SAI,EE13B006

For the award of the degree

BACHELORS IN ELECTRICAL ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2017

THESIS CERTIFICATE

This is to certify that the thesis titled **Influence of CAD Routing Algorithms on Cryptographic Side Channel Attacks**, submitted by **B Akhil Sai,EE13B006**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Bachelor of Technology in Electrical Engineering**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. 1
Chester Rebeiro
Assistant Professor
Dept. of Management Studies
IIT-Madras, 600 036

Place: Chennai

Date: 8th May 2017

Acknowledgments

I would like to thank my professor, **Dr. Chester Rebeiro** for giving me the opportunity to work on this project. These results would not have been possible without his guidance. I would also like to thank **Ramanjaneya Reddy** for his support and guidance throughout this period.

Abstract

Cryptography is the science of secrecy. Many great theoretical computer scientists worked for years to design ciphers that are theoretically secure. But, the implementations of cryptosystems are generally done by not so competent engineers, which may sometimes leave serious security loopholes. Exploitation of such security loopholes from the implementations of Cryptosystems are known as Side Channel Attacks. A Side-Channel Attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms. Unless the implementations are secure, the entire cryptosystem cannot be called secure.

Engineers design implementations of cryptosystems under various constraints like Performance, Execution Time, Size of the Circuit etc. The abstract of this project is to check whether any of these constraints play a role in determining the feasibility of a Side Channel Attack. Out of the huge class of SCAs, this project will deal with attacks based on the analysis of the Power Consumption of the circuit that executes a cryptosystem. The cryptosystem here is the famous 128-bit Advanced Encryption Standard, being implemented on an FPGA board. The template attack framework is used to attack the cipher. The constraints are Area Minimization and Speed Maximization. The implementations' routing is done in such a way that the constraints are obeyed. As both these constraints cannot go hand in hand with each other, "Under which constraint is it easier to attack the cipher?" is the question this thesis tries to answer. Also, there are many successful DPA attacks on the AES implemented on an FPGA but there are no reported instances of a successful template attack on the same. This project also discusses the feasibility of a template attack on the AES.

Table of Contents

Introduction

As specified above, exploiting security loopholes in the implementations of the cryptosystems is what is known as a Side Channel Attack. Power Analysis is one such sub-class of the Side Channel Attacks, where the power consumption of the device that is running the cryptosystem is measured and analyzed.

A Business Case

Cryptographic smart cards such as debit cards, credit cards etc. contain chips which have secret keys embedded in them. Whenever this card is swiped, the circuit on the chip is powered and the algorithm is executed. The power consumption of this execution can be measured and analyzed to find some information about the secret key embedded inside the chip.

Various card makers want to provide various services to their customers. Some of these companies want the swiping system to be extremely fast whereas some of them want the cards to be as small as possible. Each requirement has it's own set of CAD routing algorithms and constraints. If doing any of this makes the card easier to attack, then it is a serious issue the companies have to keep in mind.

Additionally, why AES 128-bit on an FPGA board has been chosen to attack is because the smart cards are made up of FPGA circuits. The 128-bit AES is a global standard encryption set by the NIST in 2001.

Apart from smart cards, many other security systems use FPGAs to build circuits and encrypt data. Hence, this forms a valid business case to tackle.

The Template Attack

As the name suggests, the template attack exploits the fact that power consumption depends on the data that is being processed. An attacker who has a sample device(s) running the cryptosystem measures power consumed for various sets of data. He then estimates the data dependency on these power traces and forms templates based on this dependency of power consumed. The attacker then stores these templates. If he gets access to a device whose secret key is not known, he will measure the power consumed by this device and match it with all the templates he stored back then. The best matched template gives the attacker a hint about what values were processed in the unknown device. This attack has 2 phases:

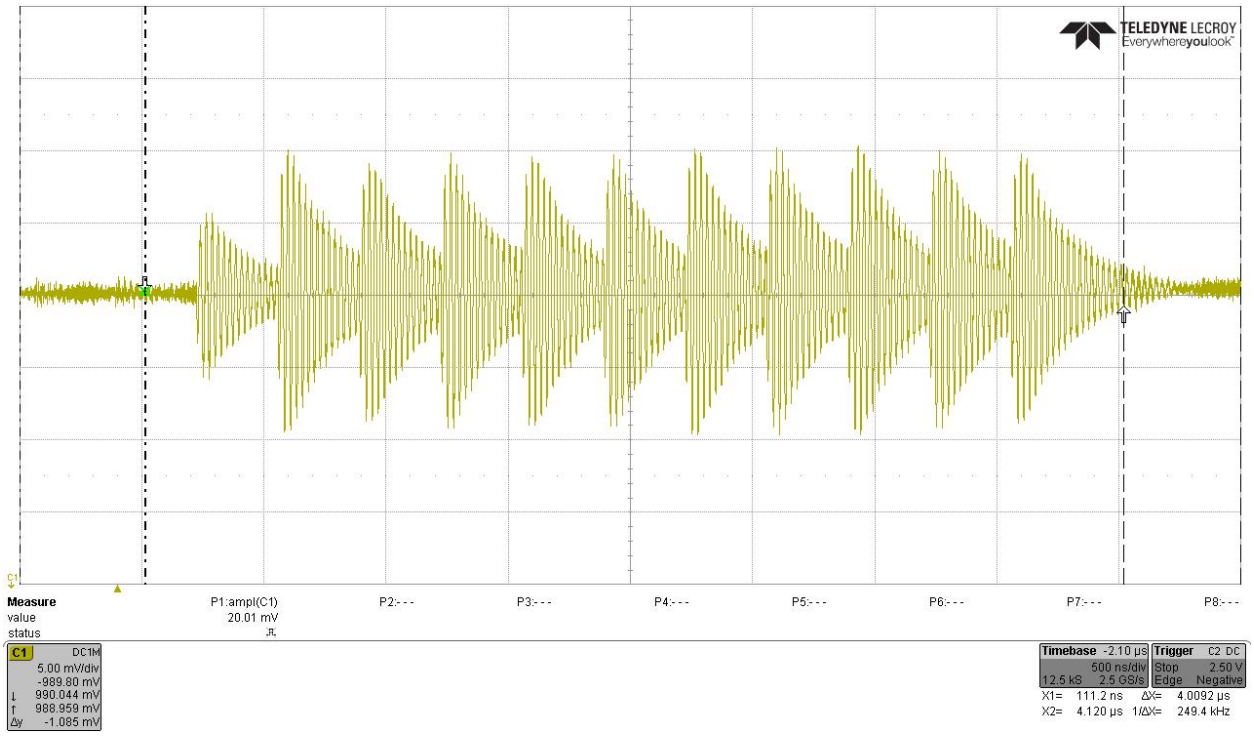
- **Template Building Phase**: The cipher has to be perfectly studied to know which part of it makes the most significant contribution to the power consumption. Interesting points in a power trace are those points that contain the most information about the characterized instruction. A strategy has to

be found to find such interesting points. As we go further, we see how difficult this task in case of an FPGA.

- **Template Matching Phase:** The power trace for the unknown device has to be measured and a suitable template matching algorithm has to be used to match this trace with the templates. Suitable noise estimation algorithms should also be used to eliminate any sort of physical or algorithmic noise. Different kinds of algorithms are used in this project to obtain the best possible result. Many challenges arise in this phase also, which are listed down below.

Analysis of Power Traces

The AES 128-bit encryption algorithm runs on a Xilinx Kintex 7 FPGA chip. It is a repetitive algorithm which runs for 10 rounds. The power trace comprises of 11 individual peaks, each representing the power trace of a single round. One round of the AES takes one clock cycle to run. The 11th additional round appears because of an incomplete execution of a round. As can be seen from the figure below, the 1st round, which is different from the other 10 rounds, is the odd one out, making it the incomplete round.



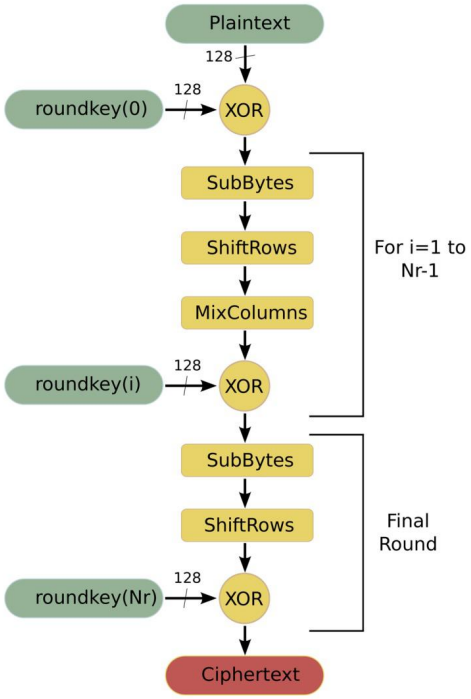
A Typical AES Power Trace

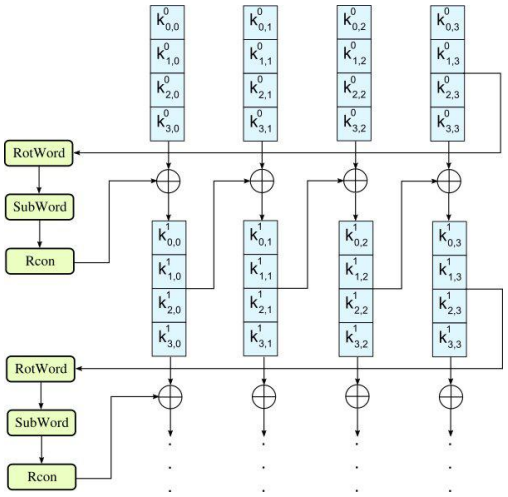
The AES block cipher, like every other block ciphers, has a substitution layer, a permutation layer and key addition. Apart from this, the key generation is also done using the same substitution layer used in AES core.

The 128-bit input is divided into 16 8-bit blocks, each of which is fed into an 8-bit SBox. In one round, 16 such SBoxes are executed. Along with this, the key generation algorithm also executes 4 such SBoxes, taking the overall count to 20.

The output from the substitution layer is forwarded to the Shift Rows layer, which permutes the 128-bit string according to some formation.

This string is then sent to the MixColumns layer, which further permutes the ciphertext. This step is executed only upto round 9.

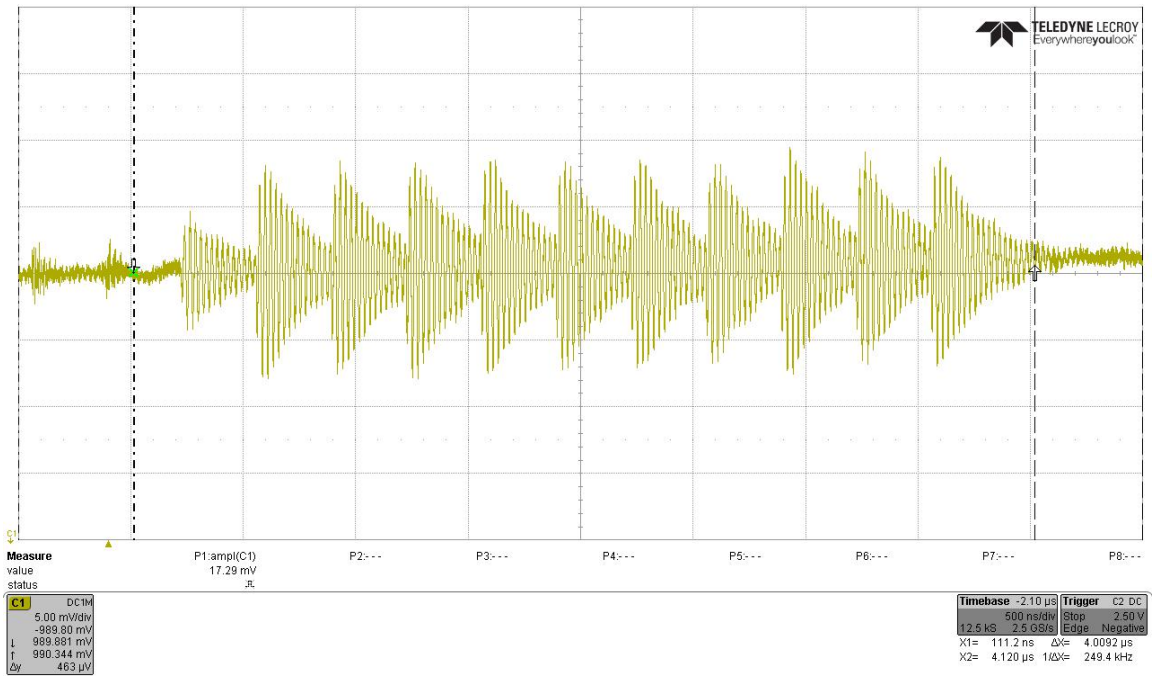




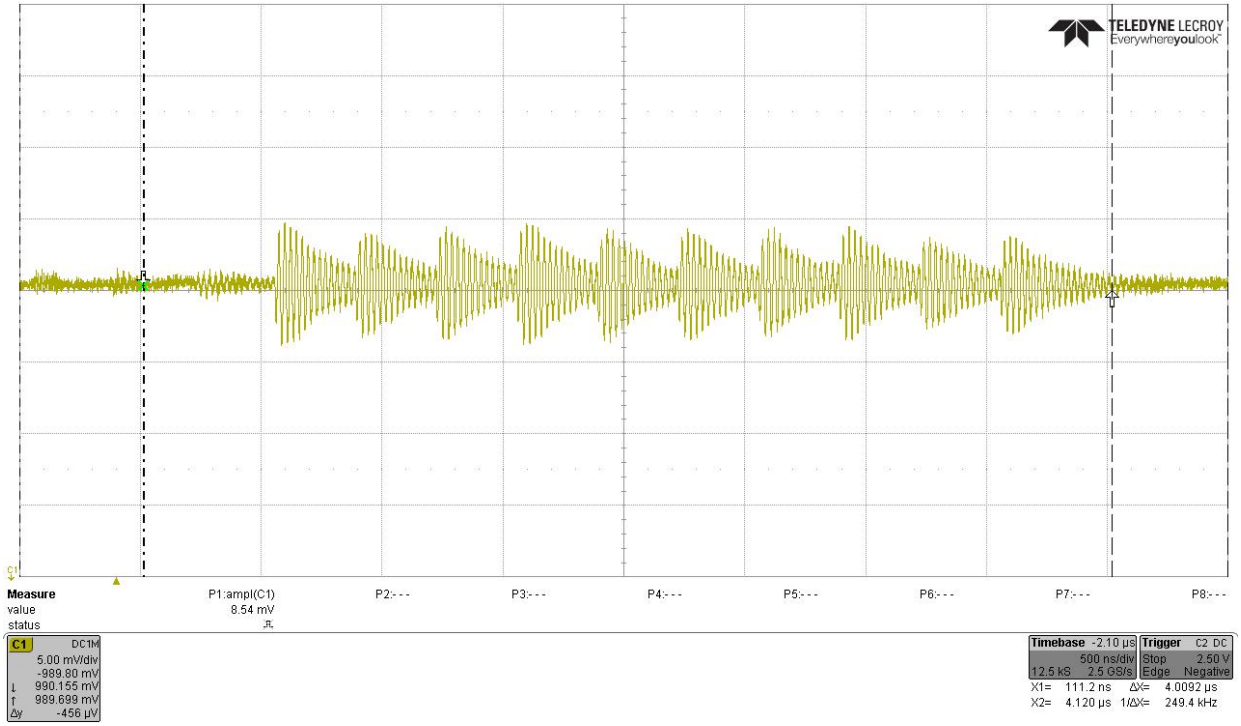
The Flowchart, credits: www.researchgate.net

The Key Schedule, credits: https://en.wikipedia.org

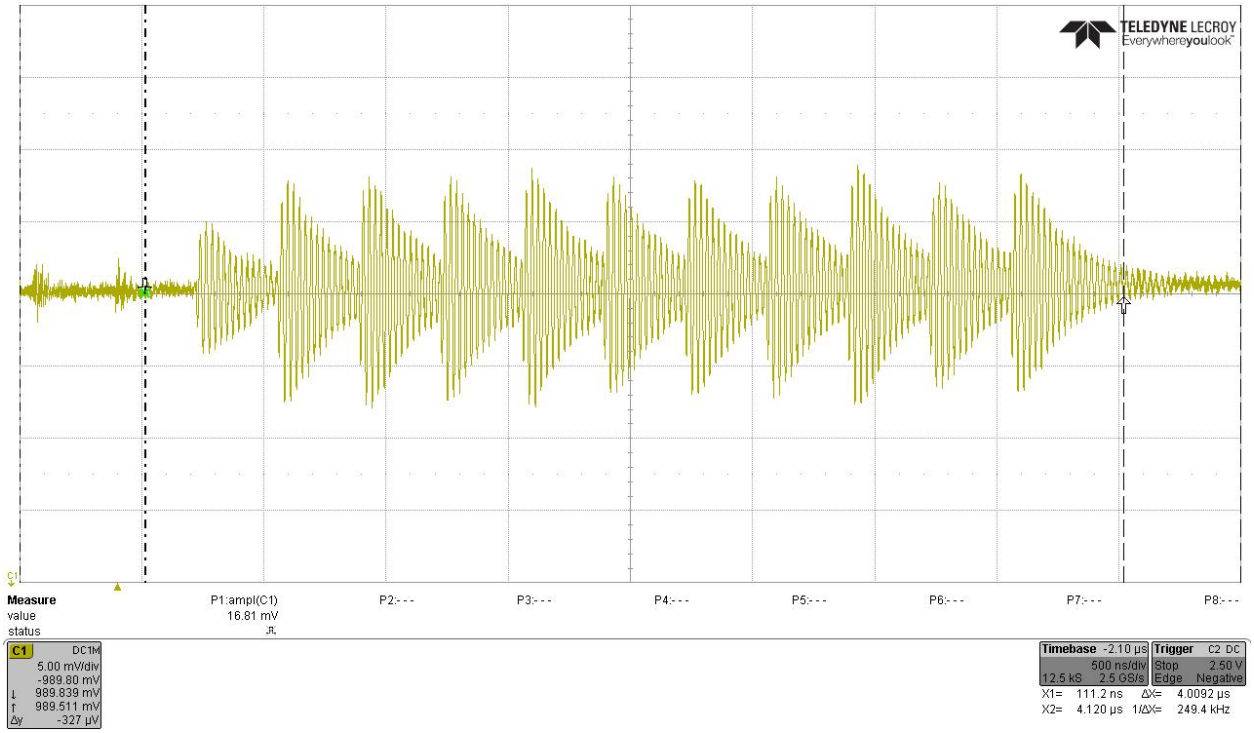
There also exists a Key Generation algorithm, which runs in parallel with the AES core, to generate the key for the next round. So, each cycle of power trace seen above is the combination of both the AES core's power consumption and Key Generation algorithm's power consumption. The Key Generation algorithm also processes 4 8-bit SBoxes, contributing a lot to the power consumption.



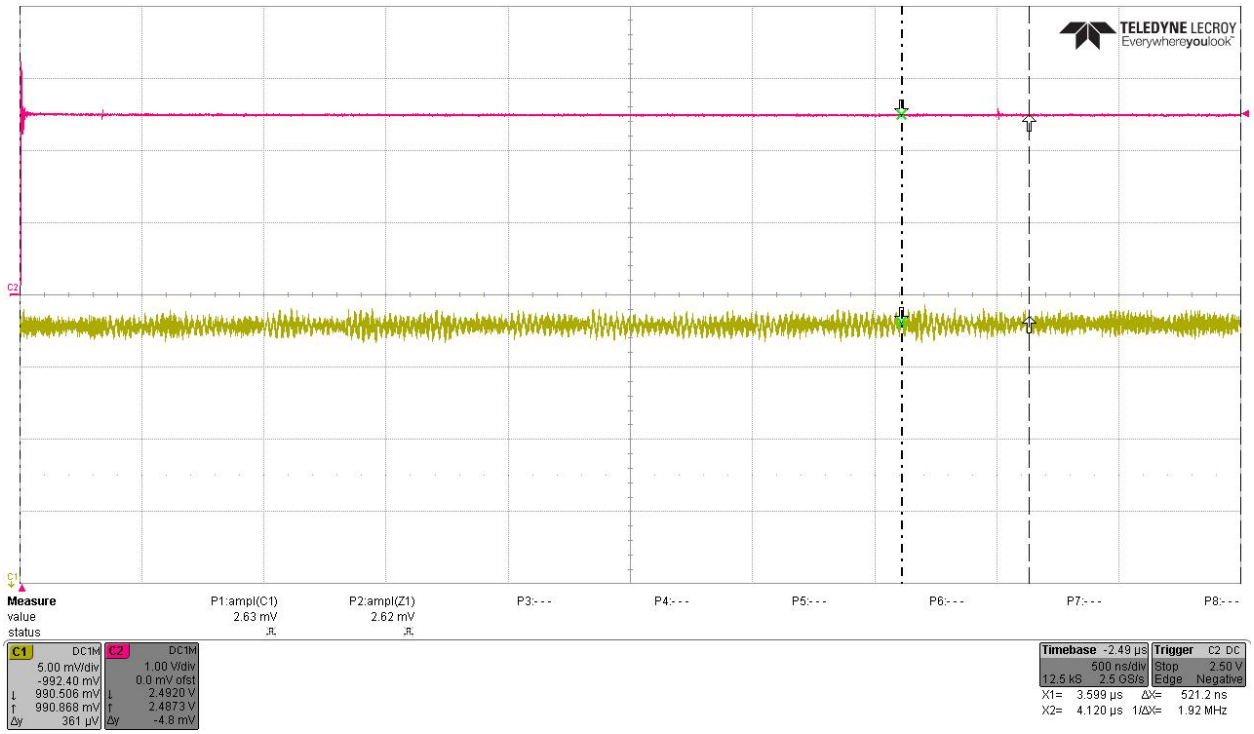
Power Trace without Mix Columns



Power Trace just with Key Generation, without AES core



Power Trace without Shift Rows and Mix Columns



Power Trace without SBoxes

Looking at each layer in detail, we get to know that the SBox layer is highly operation intensive, and consumes a lot of power.

As seen above, the SBox layer consumes the highest amount of power. The SBox is the part of the cipher which has to be attacked. The SBox is implemented using Galois field operations, which are very operation intensive.

What to Attack?

The standard models of a template attack involve the Hamming Weight Model and the Hamming Distance Model. The Hamming Weight model works out nicely for the program running in a microcontroller hardware, and the Hamming Distance model works out for a DPA kind of an attack. Below are the various ways of forming templates :

- Hamming Weight Model - Hamming Weight of Input of SBox - Let's call it method 1- 9 possible template values
- Hamming Weight Model - Hamming Weight of Output of SBox - Let's call it method 2 - 9 possible template values
- Hamming Distance Model- Hamming Weight of the difference between Input and Output of the SBox - Method 3 - 7 possible template values
- Intermediate Value 1- Method 4 - Multiplication of the number of 1's in one intermediate value with the number of 1's in another intermediate value at the same level. 16 possible template values.

The intermediate values above, have been selected on the basis of Fan-out values and the number of gates the wire is being sent as an input. The more the Fan-out value of a particular wire, the more power it takes to switch the value of the wire.

So, based on the above variables, template formation takes place.

The Attack

Template Formation

The template attack has a standard framework. Each power trace is assumed to be a multi-variate Gaussian distribution. Hence, for building templates, the traces that correspond to the same pair are grouped together by estimating their mean and co-variance vectors. As the device which is being attacked is an FPGA, the entire power trace in itself can be considered important. The Maximum Likelihood estimation algorithm is generally used for attacking unknown data sets. But, as the noise level is so high in case of an FPGA, the number of traces to be taken are also high. When the number of traces to be operated on becomes very large, the ML-estimation algorithm becomes computationally in-feasible.

A simplified ML-estimation algorithm is the Least-Square algorithm or the LSQ algorithm. This algorithm doesn't take the Co-Variance matrix into account. The LSQ algorithm takes the difference between the template and the trace, squares the difference and adds it to the total. The lesser this value, the more the traces are close to each other.

$$\ln p(t;m) = -1/2*(\ln(2.\pi)^N + (t-m)'.(t-m))$$

Only the mean templates are formed by just taking the average overall traces of a particular template.

Challenges

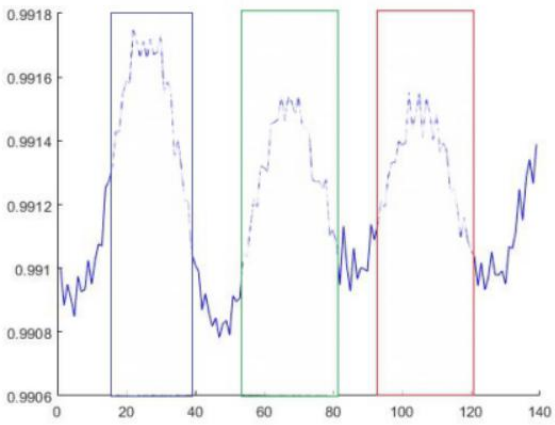
Templates are formed by grouping traces which have the same template value together. This grouping can be done by just adding the traces directly. However, this causes multiple problems:

- The SBox is a complicated set of machinery. The values on the basis of which templates are to be formed are very difficult to analyze. To give you an idea about what happens inside of an SBox, the SBox contains close to 500 logic gates and a 100 intermediary values or wires. Apart from this, the main disadvantage doesn't stem from the complication of the SBox. It stems from the limitation of the hardware available:
    - In case of a microcontroller, the execution of a program happens via something known as sequential instruction execution. The power consumed by the chip can be separated instruction by instruction. See the figure below for a better understanding: It is called the time wise separation of instruction wise power consumption. If instruction wise power can be separated, attacking the cipher is a much much easier task. But sadly, this is not what happens in case of an FPGA.

3 following instructions

- MOV a,b
- ADD c= a+b
- MOV c

The following boxes represent the power consumption for each of the above instructions

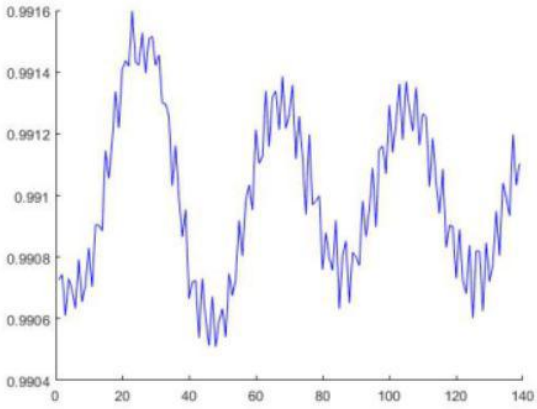


*The power trace is not taken from the microcontroller, is put there just for representation

- In the case of an FPGA, the power consumption does not follow the same pattern. The power trace outputted is a function of all the instructions executed. The time separation as shown above is not possible, which makes the attack very difficult.

- MOV a,b -------- I_1
- ADD c= a+b --- I_2
- MOV c ---------- I_3

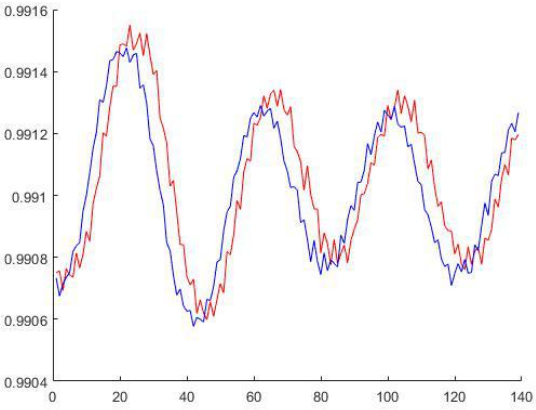
Power Consumed, P= $f(I_1,I_2,I_3)$, where I_1,I_2,I_3 contain the power required to change the previous wire values, gate power consumption, wire capacitances etc.



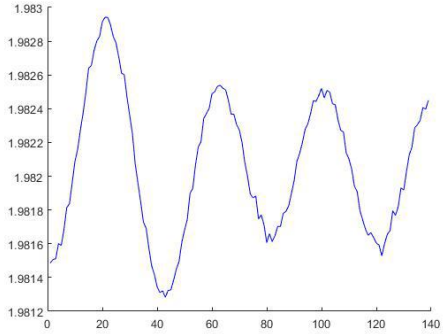
*The power trace is not for these particular instructions, it is just put there for representation.

- So, to get an estimate about close to 500 gates and 100 intermediate wires is not an easy task.
- The major challenge is to figure out which part of the SBox consumes the highest amount of power. Even if this is figured out, attacking the cipher on the basis of power traces is very difficult.

- The traces may not align with each other. As the traces are oscillating sinusoid kind of waves, such displaced trace addition can cause a serious disturbance to the shape of the trace.

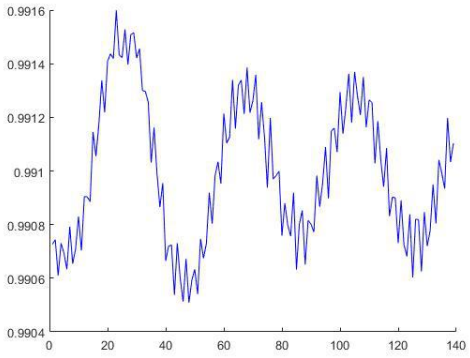


Misaligned traces`



Resultant trace out of shape

- Noise addition and elimination - Traces which have low noise levels should be used for template formation, as high noise levels mask out the valuable information present in the trace.



A Noisy Trace- the distortion of peaks

- An extension of the challenge above, if too many noisy traces are added, instead of improving the template, will worsen it. Hence, different trace groupings have to be done, to get the best template out of them.

Algorithms used and improvements made

Nothing much can be done regarding the 1st problem and work has to be done keeping those constraints in view.

To tackle the 2nd problem, a move and check filter has been implemented. This filter moves the trace on top of the template and checks where it overlaps perfectly, and adds it. To check where the "to be added trace" overlaps perfectly with the template, a co-variance based approach has been used. The displacement at which the co-variance is maximum is chosen, the trace is displaced by that amount and is added to the template.

To tackle the 3rd problem, for every data set, 100 power traces are taken and averaged. It may not eliminate the noise, but will at least assure the same level of average noise for all data sets. As can be seen in the above pictures, the amplitude of the power trace can go down to values as low as 0.4mV. Even a small disturbance in the connecting cable can create a noise that can completely override the signal.

As for the 4th problem, another approach can also be used to improve the SNR. Amplitude of each wave can be measured and only those waves whose amplitude is low can be chosen to form templates. As the noise level influences the amplitude, the lower the amplitude, the lower the noise. This has not been implemented because of lack of time, but is a very crucial idea which can improve the results drastically.

Template Matching

As described above, the template matching can be done using the ML-Estimation algorithm or the LSQ algorithm. But, results can be improved further by using additional matching techniques. Some of them are listed below:

- The ML-estimation algorithm is computationally in-feasible because inverting a 100x100 matrix whose determinant is very low can take a lot of time. The advantage gained out of this is also very low.
- The LSQ algorithm , as described above, takes the "to be attacked" power trace, takes a point to point difference with the template, squares the difference at every point and adds them to a total. The lower the total is, the better the trace matches the template. The lowest of all such totals is the best template match.
- The Co-Variance method, where the template matching is done by calculating the Co-Variance of the template and the trace. The higher the Co-Variance, the better the match.
- Euclidean Distance method, where the template matching is done by calculating the Euclidean distance between the template and the trace. The lower the distance, the better the match.

In all the above template matching methods, the 2nd challenge written down in the above section arises. The "to be attacked" power trace may not be aligned properly with the templates. Hence, the same move and check filter has been implemented for all the above 4 methods to get the best possible result.

The best result was obtained when the Euclidean distance method was used. Method wise results are presented below:

Results

The experiment was run for 2 routing algorithms- Area optimized simulation and Speed maximized simulation. In the case of Area optimization, saving the space is the key, irrespective the length of the wires. Hence, the wires curl round and round, trying to optimize the area used. Whereas in case of speed optimization, the wires should be kept as short as possible so as to ensure that there are no propagation delays. These are 2 completely different circuits and scenarios that are being attacked.

This section demonstrates the results obtained for both the above routing algorithms, for 3 different data sets each, and for all the 3 template matching algorithms.

A sample of 500 traces is being attacked, with templates formed out of 30000 traces. The following is a metric of the success of the attack:

- 10 points - If the match obtained out of the attack correctly matches the actual value
- 5 points - If the match obtained out of the attack misses the actual value by +- 1

- 0 points - If the match obtained out of the attack is random and doesn't come anywhere near the actual value

Area Optimized Case:

Test Case 1:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	19	41	440	395	49	121	330	1095	110	167	223	1935	20	30	450	350
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	51	105	344	1035	85	155	260	1625	92	169	239	1935	86	78	336	1250
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	19	40	441	390	20	54	426	470	118	193	189	2145	20	30	450	350

In the case of the 1st data set, the best possible result is in the case of Hamming Distance Model and Euclidean Distance algorithm, with a total score of 2145. However, the results are much more impressive in case of Method 4 with Co-Variance algorithm where the total score is only 1250. The reason being the number of prospective values in case of Hamming Distance method is only 7 (1,2,3,4,5,6,7). Whereas in case of Method 4, the number of possible values are 16, thereby making the results more interesting.

Test Case 2:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	53	123	324	1145	80	150	270	1550	86	162	252	1670	80	42	378	610
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	25	76	399	630	74	129	297	1385	80	168	252	1640	37	39	424	565
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	59	122	319	1200	69	141	290	1395	33	101	366	835	60	88	352	1040

The highest score in this case is again from the Hamming Distance model, when LSQ algorithm was used. The highest score is 1670.

Test Case 3:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	55	112	333	1110	74	156	270	1520	94	173	233	1805	57	28	415	710
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	35	79	386	745	71	129	300	1355	82	150	268	1570	19	48	433	470
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	8	53	439	345	16	29	455	225	64	141	205	1345	27	73	400	635

The highest score in this test case also, comes out of the Hamming Distance Model, when the LSQ algorithm was used. The highest score in this case is 1805.

Coming to the Speed-Maximization part,

Test Case 1:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	111	162	227	1920	87	143	270	1585	102	164	234	1840	69	59	372	985
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	51	87	362	945	70	151	269	1455	113	145	252	1855	58	75	367	955
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	122	168	210	2060	24	48	428	480	52	94	354	990	45	45	410	675

The maximum score in this case 1920, which is obtained when Method 1 is used and the LSQ template matching algorithm is used.

Test Case 2:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total

	86	144	270	1580	66	158	266	1450	112	167	221	1955	72	65	363	1045
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	58	108	324	1120	75	162	263	1560	83	180	237	1730	61	65	364	935
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	104	177	219	1925	82	173	245	1685	44	90	366	890	61	53	386	875

The highest score in this case is 1955, which is obtained as a result of the Hamming Distance Model and the TM algorithm being the LSQ algorithm. The ED algorithm used under Hamming Weight 1 Model is also not far behind, with 1925 score.

Test Case 3:

TM Algorithm	Method 1				Method 2				Method 3				Method 4			
LSQ	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	85	143	272	1565	87	129	284	1515	100	154	246	1770	74	60	366	1040
Co-Variance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	53	117	330	1115	74	137	289	1425	115	171	214	2005	45	65	390	775
Euclidean Distance	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total	10 pts	5 pts	0 pts	total
	119	180	201	2090	29	56	415	570	44	90	366	890	59	74	367	960

The highest score in this case is 2090, which is obtained as a result of the Hamming Weight Model and the TM algorithm being the Euclidean Distance algorithm.

Interpretation of the results

In case of the Area Optimized case, the best results for all the 3 test cases came from the Hamming Distance Model. The template matching algorithms defer between LSQ and Euclidean Distance, but the results are more or less the same. The score was exceptionally high in the 1st test case but was relatively lower in the 2nd and 3rd test cases. The power trace has much lesser amplitude compared to the speed maximized case's power trace, which makes it more difficult to launch a side-channel attack.

In case of the Speed Maximized case, the best results for two of the test cases came from the Hamming Weight 1 model and the Hamming Distance model giving the best result for one of the test case. Even in this test case, the scores generated by HW1 and HD models differed only by 50 points. The

Hamming Weight 1 model can thereby be assumed to give out the best results for the Speed Maximized case.

One difference that can be observed is the average best score in case of speed maximization is much higher than in the Area Optimized case. Another important observation is the difference in the models which give out the best scores. In the Area Optimized case, the Hamming Distance gave the best score whereas in the Speed Maximized case, the Hamming Weight model is the best suited. This tells us a lot about the influence of routing algorithms on side-channel attacks.

In the Area optimization case, minimization of the area occupied by the circuit has to be minimized. Hence, the wires are curled around to save space. In the speed maximization case, the wires are kept as short as possible so as to prevent any propagation delays. In the 1st case, as the wires are long, it takes a lot of power to change the values in the wires. **The more the length of the wires, the more they resist change**. This is exactly what the Hamming Distance model captures. This is **one possible explanation** for the above peculiar behavior. These results give rise to future prospective work that can be done in this field.

Prospective Work

The above results give rise to prospective work which can be done in this field. Some of these future roadmaps are listed below:

- **Improving the attack :** As described above, the SBox is a complicated piece of machinery which has hundreds of gates and wires in it's circuit. The wires which connect many gates and which diverge to many different gates are the ones which can be attacked easily. Especially in the case where Area is optimized, this can improve the attack substantially. An in-depth analysis into the RTL schematic of the SBox implementation will help identify such important wires and gates which contribute significantly to the power consumption.

- **Launching a full scale template attack:** From the available results, how to launch a template attack on the AES block cipher? A **Chosen Plaintext Attack** can be launched, where 8 bits of the plaintext are kept fixed and the other 120 bits are chosen at random. If a sufficient number of traces are averaged, then the data in the resultant trace will be independent of these 120 bits and will take an average value. Apart from the 120 bits data, the 128 bit key generation related data is also included in the power trace. Each power trace is a combination of 20 SBoxes and we need to extract data about 1 SBox. Generally, an unknown 128-bit key is attacked.

$$\begin{aligned} \text{Power} &= 20 \text{ SBoxes} \\ &= 16 \text{ AES Core SBoxes} + 4 \text{ Key Generation SBoxes} \\ &= 1 \text{ CPA SBox} + 15 \text{ Random SBoxes} + 4 \text{ Key Gen SBoxes} \end{aligned}$$

Averaging many traces of data will average out the 15 Random SBoxes, but the Key Generation SBoxes are a problem. So, to get an estimate of the power consumed by the 4 Key Gen SBoxes, the average of 16 SBoxes has to be subtracted from the total Power.

4 Key Gen SBoxes Power = Total Power - 16 Random SBoxes average

1 CPA SBox Power = Total Power - 15 Random SBoxes - 4 Key Gen SBoxes

This SBox can be attacked after all this is done.But, the success of this attack is extremely doubtful as the amplitude of the signal of a single SBox is so small that it may be masked or overridden by noise, be it algorithmic or physical. The number of traces that need to be processed may also be as high as 100000, making the attack extremely hard.

- **Checking the influence of such routing algorithms for other side channel attacks like the DPA :** As seen above, a full scale template attack can be extremely difficult and impractical to launch. But efficient techniques like the Differential Plaintext Attack can be used to check the influence of routing algorithms on the feasibility of such an attack. The DPA also works on the Hamming Distance model, which turned out to be the most efficient mode of attack in the Area Optimized case. Also, what was attacked here was just a single SBox. In case of the full AES, the influence of the routing algorithm is much stronger, as there is so much to optimize(20 such SBoxes).For example, take the Area Optimization case. The wires curl around more and more as the circuit is much more complicated and space saving becomes the key. The DPA is also much simpler to launch compared to a template attack.

Bibliography

- Relevant material taken from the book "Power Analysis Attacks - Revealing the Secrets of Smart Cards" by Stefan Mangard, Elisabeth Oswald and Thomas Popp.
  http://www.springer.com/in/book/9780387308579

- All the power trace measurements were done using Teledyne Lecroy oscilloscope.

- All the figures apart from the power traces were generated using MATLAB 2016a, IITM license. All the simulations and the results were all obtained using MATLAB 2016a, IITM License.