

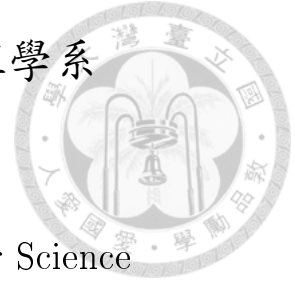
國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering
College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



未知替換盒之旁通道分析

Side-Channel Analysis of Unknown S-Boxes

童御修

Gi-Siu Tong

指導教授：鄭振牟 博士

Advisor: Chen-Mou Cheng, Ph.D.

中華民國 105 年 7 月

July, 2016

誌謝

感謝指導教授鄭振牟老師給予我們充足的資源和開放的研究環境，使我得以自由自在的發展自己的興趣。感謝陳君明老師引領我進入密碼學領域以及在許多專案和會議上的幫助。感謝陳君朋學長這兩年多次的相罩和學業、處事甚至生活上的各種關心與建議。謝謝洪維志老師、葉闊、李文鼎、吳忠憲和游世群，和你們的討論讓我對SCA有更深刻的認識，論文中許多想法也是在其中產生的。另外我也要謝謝銓安智慧科技的林子桓和黃瑞陽學長在智慧卡實驗上的諸多協助。

能夠和快速密碼學實驗室的同學們相遇相知是唸碩士班最大的收穫：郭博鈞介紹許多工作機會給我，還教我游泳、帶我登山。和陳明興學長談論時事政治相當有趣，改變了我對社會的一些既定看法。在憲哥身邊我學到很多實用的電腦知識。和李文鼎(謝謝你特地跑來聽我試講，讓我隔天能順暢地完成口試)、王國婷及王偉政在實驗室閒聊總是十分地放鬆愉快。還有一同備戰口試的郭育辰和李政諭、每週固定研討SCA技術的學弟們、帶大家在九州趴趴走的張運安、鼓勵我出國交換的李鎬、.....。謝謝631的每一個人，你們豐富了我的生命。

剛考上研究所時，我還是個隨波逐流、徬徨無措的大學生。現在，我帶著兩年來累積的想法和學到的知識，準備要堅定地踏出下一步了。很慶幸在學生生涯的尾溜有這一段能心無旁騖地學習成長的日子，再次謝謝師長和同學這段時間的關照。

最後，我要謝謝我的家人和朋友，沒有你們有形和無形的支持，我無法成為今日的我。謝謝冠蓉，你的出現讓我的存在有了意義，你的陪伴是我繼續前進的動力來源。

摘要

旁通道分析對密碼裝置的實作是一個強大的威脅，而差分能量分析是旁通道分析中的一個以高效率而聞名的分支。然而，當差分能量分析被應用在演算法中有未知替換盒的情況時，它會因為需要列舉過多的替換盒可能性而無法達成。本文使用代數旁通道分析來處理未知替換盒的問題。結果顯示，若存在一個模板能提供旁通道資訊，則代數旁通道分析可成功取得Serpent演算法的未知替換盒及回合密鑰。

關鍵字: 旁通道攻擊, 代數旁通道分析, 未知替換盒, *Serpent* 加密演算法



Abstract

Side-Channel Analysis (SCA) is a powerful threat against the implementation of cryptographic devices. And Differential Power Analysis (DPA) is a popular type of SCA because of its efficiency. However, when applying DPA to an algorithm with unknown S-Box, DPA could not work well due to the large enumerating space of S-Box. In this thesis, we use Algebraic Side-Channel Analysis (ASCA) to deal with the unknown S-Box problem. The result shows that the unknown S-Boxes and secret round keys of Serpent can be retrieved if a template which provides the side-channel information is given.

Keywords: *Side-Channel Attacks, Algebraic Side-Channel Analysis, Unknown S-Box, Serpent*



Contents

1	Introduction	1
1.1	Motivation	1
1.2	Related Works	2
1.3	Contributions	3
1.4	Roadmap	4
2	Preliminaries	5
2.1	Template Attacks (TA)	5
2.1.1	Template Building	6
2.1.2	Template Matching	7
2.2	Boolean Satisfiability Problem (SAT)	7
2.3	Serpent	8
3	Algebraic Side-Channel Analysis	10
3.1	Problem Definition	10
3.2	The Algebraic Representation of the S-Box	11
3.3	Represent the Intermediate Values by Unknown Variables	12
3.4	Introduce the Hamming Weight Information	15
3.5	Convert to the CNFs	17
3.6	Solve the CNF by SAT solver	19
4	Experiments	21

4.1	Setup	21
4.2	Results	22
5	Conclusion	24
5.1	Future Works	24
	Bibliography	26





List of Figures

3.1 Structure of Serpent	13
------------------------------------	----



List of Tables

1.1	Success Rate of TA for 1,000 traces [HTM09]	4
2.1	Truth tables of \vee , \wedge and \neg	7
2.2	The S-Boxes of Serpent	9
3.1	Truth tables of \oplus and \cdot	11
4.1	Result: Without Error Tolerance (No XOR-clause)	22
4.2	Result: Without Error Tolerance (XOR-clause)	22
4.3	Result: With Error Tolerance (No XOR-clause)	23
4.4	Result: With Error Tolerance (XOR-clause)	23



Chapter 1

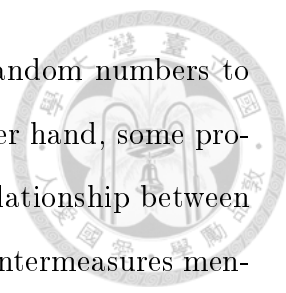
Introduction

1.1 Motivation

When a cryptographic algorithm is implemented on physical devices, the threat from Side-Channel Analysis (SCA) should be considered even the algorithm is secure under mathematical analysis. SCA utilizes the unintended data-dependent "leakage", for example timing, power consumption or electromagnetic radiation, leaked from the devices during their encryption or decryption. These leakages are called "side-channel leakages" and SCA is viewed as a "gray-box" cryptanalysis since it provides additional side-channel leakages information in addition to input/output.

In 1999, Paul Kocher et al. proposed Differential Power Analysis (DPA) [KJJ99], which efficiently extracts the secret keys of DES by the power consumption along with some simple statistical techniques. DPA soon becomes popular and many variants are published in the following years. For example, Correlation Power Analysis (CPA) [BCO04], Mutual Information Analysis (MIA) [GBTP08] and Linear Regression-based DPA [DPRS11]. These new method improve the performance of the original DPA and successfully steal the secrets of many unprotected block ciphers including the AES standard Rijndael.

Some countermeasures against DPA are also studied in recent years. Masking



[BGK04][MPO05] and threshold implementation [NRR06] use random numbers to hide the secret intermediate values of the algorithm. On the other hand, some protections try to randomise the order of algorithm to cut off the relationship between power and the operations during encryptions [VMKS12]. The countermeasures mentioned above usually need much more space and time. Also, some countermeasures are still breakable, e.g. High-Order DPA against masking [JPS05], though more efforts for the attackers. Thus, some other countermeasures focus on finding a secure variants of the original algorithm to reduce the overhead. [RS05] is the first work that applies random isomorphic AES to protect from SCA. And in [JCCC07] [WSH⁺10], different but secure SubBytes, ShiftRows and MixColumns of AES are chosen on the fly. However, most of DPAs deal with unknown key scenario. They usually assume that the attackers know all of the algorithms except the key. Therefore, DPA may fails against these countermeasures.

1.2 Related Works

[Nov03] is the first paper that attempts to recover a lookup table implementation substitution block by SCA. Novak's work analyses GSM authentication algorithm that has several consecutive substitution tables. But in most of the block ciphers, there exist too much functions between two S-Boxes. Besides, some papers in the area of Side-Channel Attack Reverse Engineering (SCARE) discuss the similar problem. [GSM⁺10] is a notable work that uses SCARE techniques to retrieve the unknown S-Box of symmetric algorithms. Nevertheless, SCARE could not be a good solution to the countermeasures that choosing different S-Box online. Since SCARE needs attackers to hold the identical devices on hand. But to those countermeasures, simply changing to another S-Box can make SCARE in vain.

The first work that connected algebraic analysis and SCA is Renauld's Algebraic Side-Channel Attack (ASCA) [RSV09]. This work deals with AES in un-

known key and unknown plaintext/ciphertext scenario. An improved method based on Renauld's work is proposed at [MBZ⁺12]. The way we transform side-channel information in this thesis is inspired by this work.

A successful ASCA relies on precise templates. Template attack (TA)[CRR02] is proposed at 2002, which is a strongest form of SCA. TA can be viewed as an oracle that can tell us what property a particular intermediate value holds once a good template exists. In [APSQ06], TA is performed in the principal space of the power traces, that makes the templates more effective. [CK13] is another improvement that avoids numerous computations when building the templates.

1.3 Contributions

In this work, we firstly apply the Algebraic Side-Channel Analysis to deal with the SCA countermeasures which use similar variants of original standard ciphers. To simplify the analysis, only S-Box in the block cipher will be changed. Note that Template Attack also needs to profile some information. But in contrast to SCARE, the profiled template can be applied to another cipher with different S-Box. The targeted cipher of the analysis is Serpent, one of the AES finalists. The results show that the function of the S-Boxes along with the secret keys will be found by the attack in 3 traces.

We do not perform a practical Template Attack. But since TA is a mature technique that discussed in many literature, it is assumed that we have a template that always gives the correct answer. Actually, the experiment in [HTM09] shows that TA can achieve 99.5% success rate in a 8-bit microprocessor for the Hamming Weight (HW) of a byte. Despite 47% success rate for a 32-bit core, we can set our ASCA transformer to tolerate 1 Hamming Weight error such that 93% success rate is promised (See Table 1.1). The number of needed traces for this case increases inevitably, but it can still be solved in 9 traces.

Table 1.1: Success Rate of TA for 1,000 traces [HTM09]

HW error	8-bit Microprocessor	32-bit ARM
0	99.5	47.0
1	0.5	46.0
2	0.0	6.2
3	0.0	0.8
>3	0.0	0.0



1.4 Roadmap

In chapter 2, we introduce the preliminary knowledge including the idea of template attack, SAT problem and a sketch of Serpent algorithm. In chapter 3, we will precisely define the problem and present our main analysis. In chapter 4, the experiments and their results are demonstrated. In chapter 5, we make a brief conclusion and discuss the future prospects.



Chapter 2

Preliminaries

2.1 Template Attacks (TA)

Template attack (TA) is a strongest type of SCA. If the attackers want to perform TA, they need to hold a device identical to the target device. So that attackers can set any key, input any plaintext and even modify parts of algorithm. Thus, a lot of traces can be collected and with properly analysis, attackers are able to build a "Template" which we can view as an oracle that input a trace and give us the Side-Channel information.

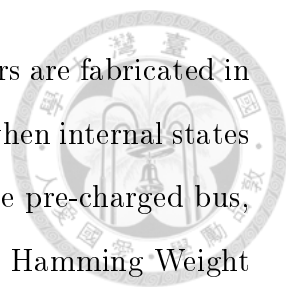
Typically, a template may be built for input key-plaintext pair, intermediate value or power models. For example, a template built for intermediate value is a oracle that reads a new trace and return the most possible intermediate value.

In this work, we are going to build templates with Hamming Weight power model.

Definition 2.1.1. Hamming Weight (HW)

Hamming Weight (HW) of a byte is the number of one in the byte, e.g. $HW(00101101) = 4$, $HW(11100101) = 5$.

Assumption 1. The power consumption of a byte processed by CPU is proportional to the Hamming Weight of the byte.



Assumption 1 is reasonable since most of the modern processors are fabricated in CMOS technology. And the CMOS consumes much more power when internal states in circuit changed. Therefore, when data transferring through the pre-charged bus, a part of the real power consumption will be proportional to the Hamming Weight of the transferred data. Furthermore, the remaining parts of power consumption can be modeled by a Gaussian distribution. So it is possible to characterize the power consumption to the Hamming Weight when a lot of consumption collected. In [MOP07], more discussions about the relation between power consumption and processed data are stated.

In the following we will briefly show how we build the template for bytes of interest in our work.

2.1.1 Template Building

Every byte of interest should have a template. We denote \mathbf{T}_i as the i^{th} trace, which is a $s \times 1$ vector with s sampling points. And v is the intermediate byte for which we want to build templates.

1. Choose random keys and plaintexts to perform the encryptions and record $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_n$.
2. Choose several points of interest, say m points. Let \mathbf{T}'_i be the i^{th} trace only with these m points.
3. Calculate v according to the keys and plaintexts then separate the traces by the Hamming Weight (0 to 8) of v
4. For each $HW(v) = w$, say includes the traces $\mathbf{T}'_1, \dots, \mathbf{T}'_{n_w}$, calculate:
 - $\mathbf{M}_w = \frac{1}{n} \sum_{j=1}^{n_w} \mathbf{T}'_j$ (Mean vector)
 - $\mathbf{C}_w = \frac{1}{n} \sum_{j=1}^{n_w} (\mathbf{T}'_j - \mathbf{M}_w)(\mathbf{T}'_j - \mathbf{M}_w)^t$ (Covariance matrix)



2.1.2 Template Matching

1. Record a trace \mathbf{t} and take the points of interest \mathbf{t}' .
2. For each $HW = w$, calculate the probability:

$$prob(HW = w) = \frac{\exp(-\frac{1}{2} \cdot (\mathbf{t}' - \mathbf{M}_w)' \cdot \mathbf{C}_w^{-1} \cdot (\mathbf{t}' - \mathbf{M}_w))}{\sqrt{(2\pi)^m \cdot det(\mathbf{C}_w)}}$$

3. Rank the probability to get the most likely HW value.

2.2 Boolean Satisfiability Problem (SAT)

Definition 2.2.1. Boolean Formula

An Boolean formula is constructed by variables using binary operators OR(\vee), AND(\wedge) and unary operator NOT(\neg).

For example, $(a \vee \neg c) \wedge (b \vee \neg c) \wedge (\neg a \vee \neg b \vee c)$ is a Boolean formula.

Definition 2.2.2. Formula Evaluation

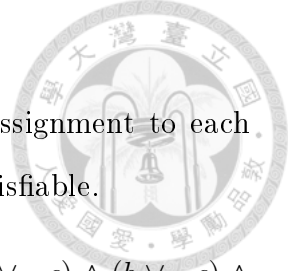
To evaluate a formula, each variable is assigned to 0 (FALSE) or 1 (TRUE). Then following the rules defined in Table 2.1, a formula will get a deterministic truth value 0 or 1.

Table 2.1: Truth tables of \vee , \wedge and \neg

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	$\neg a$
0	1
1	0



Definition 2.2.3. Boolean Satisfiability Problem (SAT)

Given a Boolean formula, check if there exists a appropriate assignment to each variable such that the formula is TRUE. If so, the formula is satisfiable.

For example, the assignment $a = 1, b = 0, c = 0$ satisfies $(a \vee \neg c) \wedge (b \vee \neg c) \wedge (\neg a \vee \neg b \vee c)$. And $(a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b) \wedge (\neg a \vee \neg b)$ is an unsatisfiable example since for all possible assignments, the result are always zero.

Definition 2.2.4. Literal

A literal is a variable or its negation, e.g. $a, \neg b$.

Definition 2.2.5. Clause

A clause is a series of literals connected by ORs, e.g. $a \vee \neg b \vee c, a \vee \neg c$.

Definition 2.2.6. Conjunctive Normal Form (CNF)

A Conjunctive Normal Form is a Boolean formula which consists of a series of clauses connected by ANDs, e.g. $(a \vee \neg c) \wedge (b \vee \neg c) \wedge (\neg a \vee \neg b \vee c)$.

The SAT problem has been studied for many years and it was shown to be a NP-complete problem in 1971. In the last two decades, the techniques of solving SAT improved a lot since the SAT problem can be applied into the field of electronic design and software verification. Thus, many SAT solvers have been proposed [Wik16b]. Modern SAT solvers usually take a Boolean formula in CNF as input and return if the formula is satisfiable. A valid assignment of variables would also be provided if so.

2.3 Serpent

Serpent[BAK98] is a block cipher with 128-bit block size and 256-bit key size. In the AES competition [Wik16a], Serpent was one of the finalist and it got the second more votes in the final round. The structure of Serpent is a 32-round substitution-permutation network and the basic block size is 4 bits.



Serpent's round function includes three parts:

1. Key mixing: XOR with 128-bit round keys
2. Substitution: Split 128 bits into 32 4-bit blocks and pass them through 32 S-Boxes.
3. Linear transform: Multiply by a 128×128 matrix. Note that in the last round, Linear transform are replaced by an additional key mixing.

There are eight S-Boxes in Serpent as Table 2.2. In round i , $S_{(i \bmod 8)}$ is used. Besides, there is an initial bit permutation and a final bit permutation in the beginning and the end respectively. These details as well as the key schedule and the Linear Transformation table can be found in [BAK98].

Table 2.2: The S-Boxes of Serpent

S_0 :	[3, 8,15, 1,10, 6, 5,11,14,13, 4, 2, 7, 0, 9,12]
S_1 :	[15,12, 2, 7, 9, 0, 5,10, 1,11,14, 8, 6,13, 3, 4]
S_2 :	[8, 6, 7, 9, 3,12,10,15,13, 1,14, 4, 0,11, 5, 2]
S_3 :	[0,15,11, 8,12, 9, 6, 3,13, 1, 2, 4,10, 7, 5,14]
S_4 :	[1,15, 8, 3,12, 0,11, 6, 2, 5, 4,10, 9,14, 7,13]
S_5 :	[15, 5, 2,11, 4,10, 9,12, 0, 3,14, 8,13, 6, 7, 1]
S_6 :	[7, 2,12, 5, 8, 4, 6,11,14, 9, 1,15,13, 3,10, 0]
S_7 :	[1,13,15, 0,14, 8, 2,11, 7, 4,12,10, 9, 3, 5, 6]



Chapter 3

Algebraic Side-Channel Analysis

3.1 Problem Definition

Given a block cipher with unknown S-Box and unknown key, we want to recover the S-Box and the secret key simultaneously. Additionally, we assume that other parts of the cipher is known and we can do profiling to obtain a reliable templates. As the discuss in Section 1.3, we focus on the ciphers that is able to be reconfigured new S-Box and the remaining parts are same as the original one. Thus, the template built from a variant can be used to the other one. The target cipher is Serpent. The reason is that 4×4 S-Box in Serpent is less complex compared to the 8×8 one in AES. Also, the linear transform part of Serpent is easily to deal with. The following is the outline of our analysis:

1. Represent the intermediate values by unknown variables.
2. Build equations from Hamming Weight information.
3. Convert the equations to CNFs.
4. Solve the CNFs by SAT solver.

In the following section, the constructed equations are represented by Algebraic Normal Form.



Definition 3.1.1. Algebraic Normal Form (ANF)

ANF is a way to present the Boolean formula that uses two operators \oplus and \cdot . The entire formula will be evaluated as 1 (True) or 0 (False).

For example, $1 \oplus a \oplus b \cdot c$ is an ANF. Table 3.1 shows the rule of the operators.

Table 3.1: Truth tables of \oplus and \cdot

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

For the simplicity, \oplus is substituted by $+$ and \cdot will be omitted in the following text. Thus, $1 \oplus a \oplus b \cdot c$ becomes $1 + a + bc$.

3.2 The Algebraic Representation of the S-Box

Let $S : x_0x_1x_2x_3 \rightarrow y_0y_1y_2y_3$ be the 4×4 S-Box function. Each output bit can be written as the linear combination of all monomials that made up of the four input bits of the S-Box. That is, the linear combination of $1, x_0, x_1, x_2, x_3, x_0x_1, x_0x_2, x_0x_3, x_1x_2, x_1x_3, x_2x_3, x_0x_1x_2, x_0x_1x_3, x_1x_2x_3, x_0x_1x_2x_3$ with 64 coefficients $a_{0,0}, a_{0,1}, \dots, a_{3,15}$

$$\begin{aligned}
 y_0 &= a_{0,0} + a_{0,1}x_0 + a_{0,2}x_1 + \dots + a_{0,15}x_0x_1x_2x_3 \\
 y_1 &= a_{1,0} + a_{1,1}x_0 + a_{1,2}x_1 + \dots + a_{1,15}x_0x_1x_2x_3 \\
 y_2 &= a_{2,0} + a_{2,1}x_0 + a_{2,2}x_1 + \dots + a_{2,15}x_0x_1x_2x_3 \\
 y_3 &= a_{3,0} + a_{3,1}x_0 + a_{3,2}x_1 + \dots + a_{3,15}x_0x_1x_2x_3
 \end{aligned}$$

For example, the first S-Box of Serpent $S_0 = [3, 8, 15, 1, 10, 6, 5, 11, 14, 13, 4, 2, 7, 0, 9, 12]$ has the algebraic representation:



$$y_0 = 1 + x_0 + x_2 + x_3 + x_0x_1 + x_0x_2 + x_1x_2 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3$$

$$y_1 = 1 + x_0 + x_0x_2 + x_1x_2 + x_1x_3 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3$$

$$y_2 = x_1 + x_3 + x_0x_1 + x_0x_2 + x_1x_3 + x_0x_1x_2 + x_1x_2x_3$$

$$y_3 = x_0 + x_1 + x_2 + x_3 + x_0x_3$$

In fact, the number of a 4×4 S-Box is only $16!$, but the ANF above contains 2^{64} possibilities. So find out a more efficient representation of S-Box to lower the number of variables is another worthwhile study problem.

3.3 Represent the Intermediate Values by Unknown Variables

The notations used in this section are listed as follows. Figure 3.1 shows the block diagram of Serpent with our notation. We assume that the plaintext, ciphertext, initial/final permutation and the linear transform function is known. The unknown part is the round keys and S-Box functions. The round index is started at $r = 0$.

r : the r^{th} round

p_i : the i^{th} bit of the plaintext

c_i : the i^{th} bit of the ciphertext

$k_{r,i}$: the i^{th} bit of the r^{th} round key

$x_{r,i}$: the i^{th} bit of the r^{th} key mixing output

$y_{r,i}$: the i^{th} bit of the r^{th} S-Box output

$z_{r,i}$: the i^{th} bit of the r^{th} Linear Transform (LT) output

$a_{r,i,j}$: the coefficient r^{th} round S-Box

ip_i : the i^{th} bit after initial permutation that changes the order of plaintext bits

fp_i^{-1} : the i^{th} bit before final permutation

$a_{r,i,j}$: the coefficients of the r^{th} S-Box

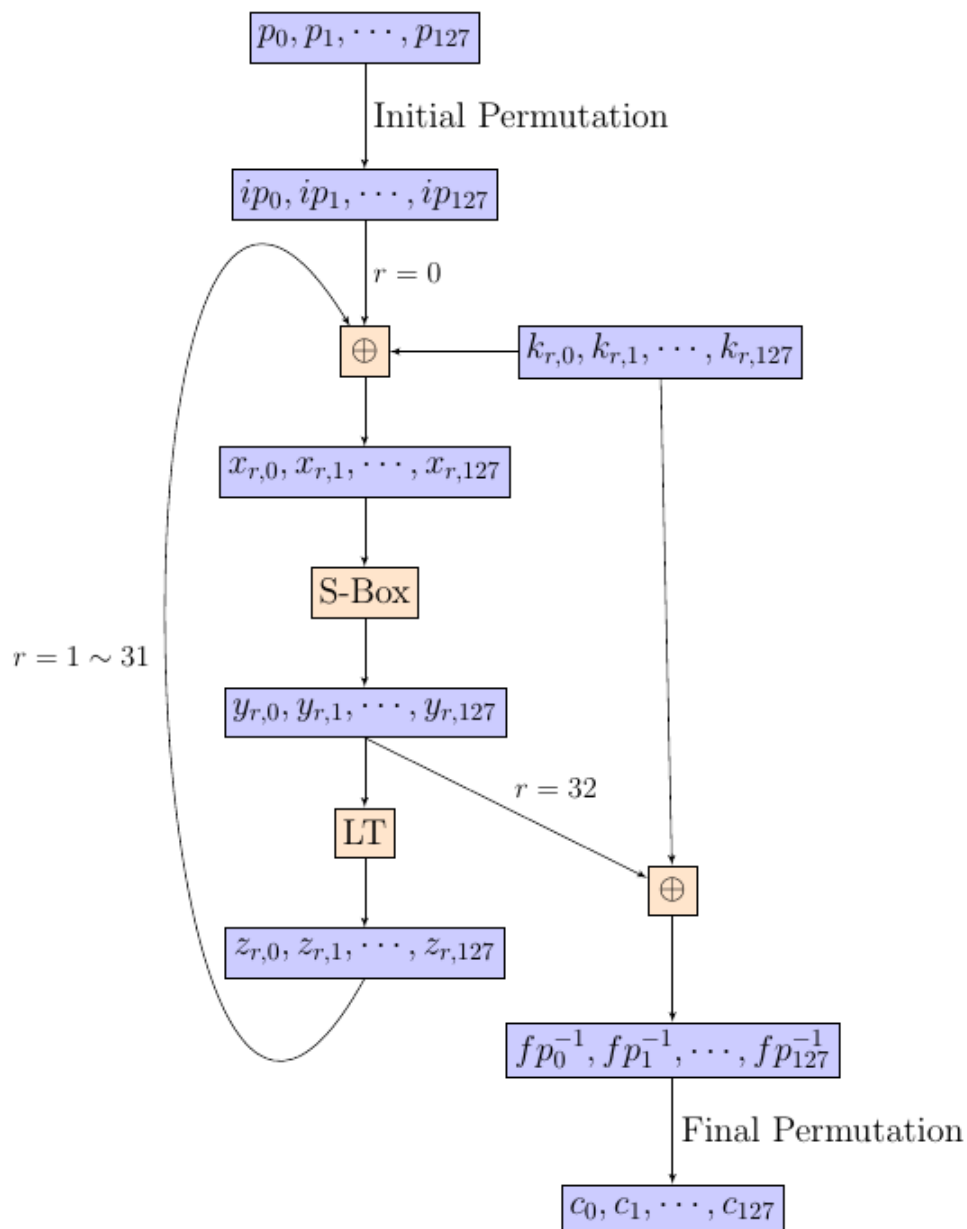


Figure 3.1: Structure of Serpent



1. Key mixing:

For $0 \leq i \leq 127$,

$$x_{0,i} = ip_i + k_{0,i}$$

For $1 \leq r \leq 31$ and $1 \leq i \leq 127$,

$$x_{r,i} = z_{r,i} + k_{r,i}$$

For $0 \leq i \leq 127$,

$$fp_i^{-1} = y_{31,i} + k_{32,i}$$

2. S-Box: Construct by the method mentioned in Section 3.2.

For $0 \leq r \leq 31$ and $0 \leq j \leq 31$,

$$y_{r,4j+0} = a_{r,0,0} + a_{r,0,1}x_{r,0} + a_{r,0,2}x_{r,1} + \cdots + a_{r,0,15}x_{r,0}x_{r,1}x_{r,2}x_{r,3}$$

$$y_{r,4j+1} = a_{r,1,0} + a_{r,1,1}x_{r,0} + a_{r,1,2}x_{r,1} + \cdots + a_{r,1,15}x_{r,0}x_{r,1}x_{r,2}x_{r,3}$$

$$y_{r,4j+2} = a_{r,2,0} + a_{r,2,1}x_{r,0} + a_{r,2,2}x_{r,1} + \cdots + a_{r,2,15}x_{r,0}x_{r,1}x_{r,2}x_{r,3}$$

$$y_{r,4j+3} = a_{r,3,0} + a_{r,3,1}x_{r,0} + a_{r,3,2}x_{r,1} + \cdots + a_{r,3,15}x_{r,0}x_{r,1}x_{r,2}x_{r,3}$$

3. Linear transform: See the details of the LT table in [BAK98]. Each output bit can be represented by the addition of several input bits.

For $0 \leq r \leq 31$,

$$z_{r,i} = y_{r,16} + y_{r,52} + y_{r,56} + y_{r,70} + y_{r,83} + y_{r,94} + y_{r,105}$$

$$z_{r,1} = y_{r,72} + y_{r,114} + y_{r,125}$$

⋮

$$z_{r,127} = y_{r,32} + y_{r,86} + y_{r,99}$$



3.4 Introduce the Hamming Weight Information

It is hardly to get the correct answer by solving the above equations since there are too many possible solutions. That is why we need the side-channel information. In this section, the method to encoding the Hamming Weight information is introduced.

[MBZ⁺12]

Assume we have a intermediate byte v during the encryption. And assume that we have a precise template let us know that $HW(v) = w$. In the following we define v_i as the i^{th} bits of v .

The above statement is equal to

$$HW(v) \leq w \wedge HW(v) \geq w$$

So let's derive the equations of these two parts respectively:

1. $HW(v) \leq w$

⇔ At most w "1"s in v

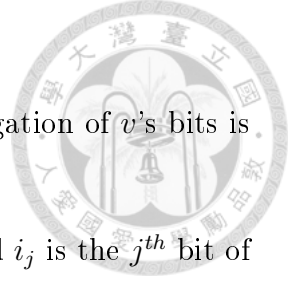
⇔ Every $w + 1$ bits in v contains at least one "0"

⇔ The product of each $(w + 1)$ -combination of v 's bits is zero.

i.e. $\bigwedge_{HW(i)=w+1} (\prod_{i_j \neq 0} v_j = 0)$, where i is a byte and i_j is the j^{th} bit of i

2. $HW(v) \geq w$

⇔ At most $8 - w$ "0"s in v



\Leftrightarrow Every $8 - w + 1$ bits in v contains at least a "1"

\Leftrightarrow The product of each $(8 - w + 1)$ -combination of the negation of v 's bits is zero.

i.e. $\bigwedge_{HW(i)=8-w+1} (\prod_{i_j \neq 0} (v_j + 1) = 0)$, where i is a byte and i_j is the j^{th} bit of i

To make this transformation more clear, a $HW(v) = 3$ example is illustrated as follows.

First, $HW(v) \geq 3 \Leftrightarrow$ at most 3 ones in $v \Leftrightarrow$ every 4 bits in v contains at least one zero. Thus, $\binom{8}{4} = 70$ equations will be generated.

$$v_0 v_1 v_2 v_3 = 0$$

$$v_0 v_1 v_2 v_4 = 0$$

\vdots

$$v_4 v_5 v_6 v_7 = 0$$

Similarly, $HW(v) \leq 3$ can be expanded to $\binom{8}{6} = 28$ equations by the second process mentioned above.

$$(v_0 + 1)(v_1 + 1)(v_2 + 1)(v_3 + 1)(v_4 + 1)(v_5 + 1) = 0$$

$$(v_0 + 1)(v_1 + 1)(v_2 + 1)(v_3 + 1)(v_4 + 1)(v_6 + 1) = 0$$

\vdots

$$(v_2 + 1)(v_3 + 1)(v_4 + 1)(v_5 + 1)(v_6 + 1)(v_7 + 1) = 0$$

Note that for $HW(v) = 0$ or $HW(v) = 8$, the information is very clear. So just let all bits equal to zero or one.

This conversion is beneficial to including error information. As mentioned in Chapter 1, sometimes TA can not make highly success rate but it may still effective if error exists. Here a error metric is defined in advance. If the distance between

the real Hamming weight of v and the Hamming Weight given by the template is d , it is called a $d - HW$ error. That is, $|HW_{Real}(v) - HW_{TA}(v)| \leq d$. The above conversion process of $HW(v) = w$ is divided into two inequalities. Similarly, to detect $d - HW$ error, only the values at the right hand side of the two inequalities need to be modified. For example, $HW(v) = 3$ will be divided into $HW(v) \leq 4$ and $HW(v) \geq 1$ to detect $1 - HW$ error. Of course that the less precise of Hamming Weight information lead to multiple solutions for a system. But in Chapter 4 we will show that this flaw can be make up by recording several traces more.

3.5 Convert to the CNFs

Once the equations are constructed, the last thing we have to do is convert them to a CNF such that SAT solver can solve it. Although many equations, they are all composed of two basic operators: $+$ and \cdot , i.e. $a + b = c$ and $a \cdot b = c$ respectively. In [BCJ07], a simple method is introduced to solve a low-degree sparse systems. Here we apply this way to generate CNFs.

1. $a + b = c$

$$\begin{aligned}
 & a + b = c \\
 \Leftrightarrow & a + b + c = 0 \\
 \Leftrightarrow & \neg \text{ odd ones in } a, b \text{ and } c \\
 \Leftrightarrow & \neg(a \neg b \neg c \vee \neg a b \neg c \vee \neg a \neg b c \vee abc) \\
 \Leftrightarrow & (\neg a \vee b \vee c)(a \vee \neg b \vee c)(a \vee b \vee \neg c)(\neg a \vee \neg b \vee \neg c)
 \end{aligned}$$

2. $a \cdot b = c$

$$\begin{aligned}
 & a \cdot b = c \\
 \Leftrightarrow & \neg((a = 0 \wedge c = 1) \vee (b = 0 \wedge c = 1) \vee (a = 1 \wedge b = 1 \wedge c = 0)) \\
 \Leftrightarrow & \neg(\neg ac \vee \neg bc \vee ab\neg c) \\
 \Leftrightarrow & (a \vee \neg c)(b \vee \neg c)(\neg a \vee \neg b \vee c)
 \end{aligned}$$



Above shows basic example with few variables. But there may exists some equations like $a + bcd + ef + g + h = 0$. How to extend this method to much more variables?

For \cdot , it is a quite pure problem since we do not have much choice. The dummy variables will be introduced for a long \cdot sequence. For example, $a \cdot b \cdot c \cdot d = e$ will be converted to

$$\begin{aligned}
 x &= a \cdot b \\
 y &= x \cdot c \\
 e &= y \cdot d
 \end{aligned}$$

where x, y and z are dummy variables. After the conversion, we can follow the rule stated above to generate equations.

For $+$, it also needs dummy variables. For example, $a + b + c + d + e + f = 0$ can be converted to

$$\begin{aligned}
 a + b + x &= 0 \\
 x + c + y &= 0 \\
 y + d + z &= 0 \\
 z + e + f &= 0
 \end{aligned}$$

where x , y and z are dummy variables.

This conversion process is called "splitting". An interesting question is that how to choose the "cutting number", i.e. the length of each splitted clause. The example above shows a cutting number 3 conversion. Here a trade-off exists. Since small cutting number will give more clauses. On the other hand, a clause with fewer variable may let SAT solvers work more efficiently. A typical cutting number choice is 6, which is suggested in [BCJ07].

Finally, in the introducing Hamming Weight stage, a lot of equations like the form of $(x_0+1)(x_2+1)(x_7+1) = 0$ are generated. Take this equation as an example. If we expand this equation, we get

$$x_0x_2x_7 + x_0x_2 + x_0x_7 + x_2x_7 + x_0 + x_2 + x_7 + 1 = 0$$

By the above method we use, five equations and four dummy variables are generated

$$a = x_0x_2x_7$$

$$b = x_0x_2$$

$$c = x_0x_7$$

$$d = x_2x_7$$

$$0 = a + b + c + d + x_0 + x_2 + x_7$$

Obviously, converting these XOR and AND equations will generate more clauses. However, $(x_0+1)(x_2+1)(x_7+1) = 0$ can be directly translated to $x_0 \vee x_2 \vee x_7$ since the meaning of the equation is that at least a one among x_0 , x_2 and x_7 .

3.6 Solve the CNF by SAT solver

In this paper, we use CryptoMiniSat 5.0 [Soo16a], which won several parts in SAT 2016 competition [SAT16]. CryptoMiniSat can read an extended Conjunctive Nor-

mal Form that contains XOR-clauses. This feature is very good to our analysis since there are many XOR-clauses in the constructed equations. One advantage of XOR-clause is that no cutting is needed, which avoids the numerous dummy variables. And the other is that the solver may reduce the variables by XORing two clauses.

For example, in the Linear Transformation phase

$$z_{1,31} = y_{1,3} + y_{1,118}$$

$$z_{1,48} = y_{1,3} + y_{1,14} + y_{1,25} + y_{1,100} + y_{1,104} + y_{1,118}$$

are generated. If we XOR two clause,

$$0 = z_{1,31} + z_{1,48} + y_{1,14} + y_{1,25} + y_{1,100} + y_{1,104}$$

is obtained. This is simpler and if all the variables are assigned temporary values, SAT solver can determine whether the assignment is valid immediately. The author of CryptoMiniSat claims that preserving XOR-clauses may achieve 2 times speed up [Soo16b]. In Chapter 4, we will show the difference of solving time between using CNF and using extended CNF.



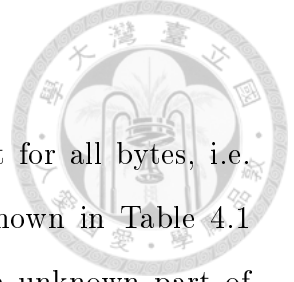
Chapter 4

Experiments

4.1 Setup

The experiment is run on Intel Xeon CPU with two 2.40GHz cores and the memory is 36GB. And the SAT solver used in this experiment is CryptoMinisat 5.0 [Soo16a]. The transforming process, from ANF to CNF, as discussed in Chapter 4 is performed with the help of SageMath [Dev16], an open-source mathematics software based on Python.

As the analysis in Chapter 3, the intermediate bytes used are the 128 bits after key mixing, 128 bits after substitution boxes and 128 bits after the Linear Transformation, for each round. Nevertheless, in the following experiment, only one round information is included to extract the first round key and the function of the first S-Box. Because once the first round information is totally encoded, the input of the second round is also known. Then the situation is just like the first round. So we only perform the experiment in the first round and the following rounds are expected to be successful too.



4.2 Results

In the first experiment the Template is assumed totally correct for all bytes, i.e. the Hamming Weight of all bytes are known. The result is shown in Table 4.1 (without XOR-clause) and 4.2 (with XOR-clause). In fact, the unknown part of the experiment is only S-Box but round key at the beginning. However, the CNFs can be solved for both S-Box and round key unknown. The least number of traces needed is 3. But as the table shows, there is a trade-off between the solving time and the number of traces. The more traces, the more information contained such that the variables leading to conflict can be removed sooner. Note that the solving time is median here since median is more representative due to the large variance of solving time.

Besides, the solving time with XOR-clause is faster than without XOR-clause. The number of literal and clause are also fewer. We have discussed the reason in Chapter 3. This result shows that CryptoMiniSat is really good at dealing with XOR-clauses as the author states.

Table 4.1: Result: Without Error Tolerance (No XOR-clause)

#Trace	#Literal	#Clause	Solving Time (Median)
3	17,331	235,211	545 sec
4	24,203	333,036	140 sec
5	30,366	416,701	34 sec

Table 4.2: Result: Without Error Tolerance (XOR-clause)

#Trace	#Literal	#Clause	Solving Time (Median)
3	7,104	37,784	423 sec
4	9,408	50,437	21 sec
5	11,712	63,007	13 sec

To make the algebraic SCA successful with the existence of error, some modifications are indispensable. The details are described in Section 3.4. It is undoubted that more traces are needed to make the solution of SAT problem unique. And the size of CNFs generated by the encoder increases certainly. Thus, it takes much more time to solve the equations. Table 4.3 shows the results with $1 - HW$ error, i.e. the system allows the Hamming Weight given by the template is away from the real Hamming Weight by at most 1.

Table 4.3: Result: With Error Tolerance (No XOR-clause)

#Trace	#Literal	#Clause	Solving Time (Median)	Success Rate
17	165,719	2,402,210	513 sec	92%
18	177,340	2,568,688	568 sec	98%
19	186,512	2,701,551	616 sec	98%
20	196,341	2,851,993	600 sec	98%
21	203,488	2,948,686	578 sec	100%
22	214,972	3,113,149	555 sec	100%

Table 4.4: Result: With Error Tolerance (XOR-clause)

#Trace	#Literal	#Clause	Solving Time (Median)	Success Rate
17	39,360	192,490	175 sec	96%
18	41,664	203,769	169 sec	90%
19	43,968	215,116	153 sec	98%
20	46,272	226,445	156 sec	98%
21	48,576	237,777	167 sec	100%
22	50,880	249,125	112 sec	100%



Chapter 5

Conclusion

In this work, an algebraic analysis combined with side-channel information is applied to solve the unknown S-Box problem. Based on Template Attack and Algebraic Side-Channel Analysis, many equations corresponding to operations of the cryptographic algorithm are constructed. And by an ANF-to-CNF converter, the equations can then be solved by a SAT solver.

The target block cipher is Serpent, which is a finalist of AES competition. The experimental results show that this method is successful once a good template is given. Moreover, this method can also tolerate error up to 1-HW error. That is, a template that gives a wrong Hamming Weight that is ± 1 away from the correct Hamming Weight is still work.

5.1 Future Works

A sound template is the base of our analysis. Thus, an important thing is to build a template in reality. The technique of building template is not difficult. For a device that does not have any countermeasure against SCA, high success rate templates are expected to be able to achieve. But some experiences are required to perform trace alignment, points choosing and numerical problem.


This type of Side-Channel Analysis is suitable for those countermeasures that


aim at changing round functions of block ciphers. In this work, we only deal with unknown S-Box situation. However, a practical countermeasure may even replace other non-linear parts. So finding out an efficient algebraic representation to write down the equations of other parts, for example, including key schedule, is another work. Finally, this method can be also applied to other block cipher like AES. This is another challenge since the S-Box in AES is 8×8 .




Bibliography

- [APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.
- [BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In *FSE*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.
- [BCJ07] Gregory V. Bard, Nicolas Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over $\text{GF}(2)$ via sat-solvers. *IACR Cryptology ePrint Archive*, 2007:24, 2007.
- [BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
- [BGK04] Johannes Blömer, Jorge Guajardo, and Volker Krümmel. Provably secure masking of AES. In *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
- [CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.

- 
- [CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
- [Dev16] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.1)*, 2016. <http://www.sagemath.org>.
- [DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.
- [GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
- [GSM⁺10] Sylvain Guilley, Laurent Sauvage, Julien Micolod, Denis Réal, and Frédéric Valette. Defeating any secret cryptography with SCARE attacks. In *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 273–293. Springer, 2010.
- [HTM09] Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown plaintext template attacks. In *WISA*, volume 5932 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2009.
- [JCCC07] Ming-Haw Jing, Zih-Heng Chen, Jian-Hong Chen, and Yan-Haw Chen. Reconfigurable system for high-speed and diversified AES using FPGA. *Microprocessors and Microsystems*, 31(2):94–102, 2007.
- [JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On second-order differential power analysis. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

- 
- [KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
- [MBZ⁺12] Mohamed Saied Emam Mohamed, Stanislav Bulygin, Michael Zohner, Annelie Heuser, and Michael Walter. Improved algebraic side-channel attack on AES. *IACR Cryptology ePrint Archive*, 2012:84, 2012.
- [MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
- [MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
- [Nov03] Roman Novak. Side-channel attack on substitution blocks. In *ACNS*, volume 2846 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2003.
- [NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.
- [RS05] A.G. Rostovtsev and O.V. Shemyakina. Aes side channel attacks protection using random isomorphisms. *Cryptology ePrint Archive*, Report 2005/087, 2005. <http://eprint.iacr.org/>.
- [RSV09] Mathieu Renaud, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: why time also matters in DPA. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.

- 
- [SAT16] Sat competition web page, 2016. Available at <http://www.satcompetition.org/>.
- [Soo16a] Mate Soos. Cryptominisat 5.0, 2016. Available at <https://github.com/msoos/cryptominisat>.
- [Soo16b] Mate Soos. Xor clauses, 2016. Available at <http://www.msoos.org/xor-clauses/>.
- [VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
- [Wik16a] Wikipedia. Advanced encryption standard process — Wikipedia, the free encyclopedia, 2016. [Online; accessed 7-July-2016].
- [Wik16b] Wikipedia. Boolean satisfiability problem, 2016. [Online; accessed 7-July-2016].
- [WSH⁺10] Mao-Yin Wang, Chih-Pin Su, Chia-Lung Horng, Cheng-Wen Wu, and Chih-Tsun Huang. Single- and multi-core configurable AES architectures for flexible security. *IEEE Trans. VLSI Syst.*, 18(4):541–552, 2010.