Static Power Analysis of Cryptographic Devices

by

©Jiming Xu, M.A.Sc., B.Eng.

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Doctoral of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

January 2019

St. John's Newfoundland Canada

Abstract

Side-channel attacks are proven to be efficient tools in attacking cryptographic devices. Dynamic power leakage has been used as a source for many well-known side-channel attack algorithms. As process technology size shrinks, the relative amount of static power consumption increases accordingly, and reaches a significant level in sub-100-nm chips, potentially changing the nature of side-channel analysis based on power consumption. In this thesis, we demonstrate our work in side-channel attacks exploiting static power leakage. Our research interest is particularly focused on profiled attacks.

Firstly, we present recent developments of static power analysis and provide our results to further support some of the conclusions in existing publications. We also give a description of the template attack we developed for static power analysis of block ciphers. This template attack uses new distinguishers which are previously applied to other data analysis fields. The results of our study are achieved using simulations in a 45-nm and 65-nm CMOS environment, and demonstrate the viability of static-power-based template attacks.

Secondly, we bring kernel density estimation into the scenario of static power analysis. We compare the performance of the kernel method and conventional Gaussian distinguisher. It is demonstrated in our experiments that the static power leakage may not satisfy multivariate Gaussian distribution, in which case the kernel method results in better attack outcomes.

Thirdly, we perform template attacks on a masked S-box circuit using static and dynamic power leakage. We are the first to compare static power and dynamic power

in the scenario of profiled attacks against masked devices. The attacks are shown to be successful, and by performing multiple attacks and adding Gaussian noise, we conclude that in the 45-nm environment, dynamic power analysis requires a high sampling rate for the oscilloscopes, while the results of static-power-based attacks are more sensitive to additive noise.

Lastly, we attempt to combine static and dynamic power leakage in order to take the advantage of both leakage sources. With the help of deep learning technology, we are able to propose more complex schemes to combine different leakage sources. Three combining schemes are proposed and evaluated using a masked S-box circuit simulated with 45-nm library. The experiment results show that the hierarchical LSTM proposal performs the best or close to the best in all test cases.

Acknowledgments

I would like to take this chance to give my greatest gratitude to my supervisor, Dr. Howard M. Heys, for his thoughtfulness and helpfulness. I am strongly motivated by our discussions throughout my Ph.D. program. His help before and during my Ph.D. program has heavily influenced me.

I would like to thank my colleagues, Xuan Dong and Zijun Gong, who provided me with a lot of help and instructions during my research.

Finally, I would like to dedicate my thanks to my grandfather, without whom I wouldn't have made up my mind to start a Ph.D. program.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The cryptographic methods used in computing devices are constrained by the amount of computing hardware resources that can be utilized. A high-end processor used in smartphones has sufficient resources to perform a full AES [1] encryption without significant sacrifice of its overall performance, which ensures that complex cryptographic applications such as the TLS protocol can be supported. Lower-end devices like microcontrollers and smartcards have relatively limited resources which may be insufficient for a full AES encryption. While some of these devices integrate a cryptographic processor to provide additional computing hardware for cryptographic functions, some others emphasize the convenience of being lightweight. Stream ciphers and lightweight block ciphers are recommended for these devices. In this thesis we mainly focus on side-channel attacks on ciphers designed for lightweight devices. However, we believe that our methodology can be promoted to other cryptographic systems implemented using the same hardware technology.

1.1 Side-Channel Attacks

A side-channel attack (SCA) is based on the assumption that cryptographic computations have some physical leakage related to the data of actual computation. For a cryptographic device, the legitimate user provides inputs such as a plaintext and a key, and this device computes the ciphertext as output. Regarding this device as the main

channel, it is possible to leak information during computation and such information is referred to as side-channel leakage. The attacker can measure the side-channel leakage, and guess the input to the main channel. Usually we assume that the ciphertext or plaintext and ciphertext are known to the attacker, and the key needs to be guessed. Using this key guess the attacker would perform computation as the main channel does, and see if his computation has any correlation with the side-channel leakage. A correct key guess should result in the maximum correlation with leakage. Specifically in the scenario of lightweight devices such as microcontrollers or RFID tags, there is a component that executes the encryption function, the key could be stored in the register of the device. The plaintext input could be some intermediate value which is not known to the user. By acquiring the leakage information, the attacker would try to recover the fixed key stored in the device. Many physical properties can be used as sources of side-channel leakage, such as power consumption [2–5], temperature [6], electromagnetic emission [7], and an acoustic signal [8].

SCA can be very efficient compared with mathematical cryptanalysis. For example, the computational complexity of the best mathematical cryptanalysis against AES-128 is known to be $2^{126.1}$ [9], which means that it is computationally secure against such attacks. However with a side-channel attack based on power leakage, the last round key of AES-128 can be recovered using less than 20,000 inputs within 10 hours in some hardware environments [10]. In other words, we could break AES-128 using a side-channel attack within several days in such circumstances.

1.2 Power Analysis Attacks

Due to the efficiency and simplicity of SCA, it has become an important area of research in the development of both attacks and countermeasures. Side-channel attacks using

power leakage, namely power analysis, is one of the major areas of side-channel attack, and has been the focus of researchers for many years. Since the proposal of the first SCA based on timing analysis [2], many power analysis algorithms and improvements have been proposed, including simple power analysis [3], differential power analysis (DPA) [3], correlation power analysis (CPA) [4], template attack [11], and mutual information analysis (MIA) [5]. In this thesis we focus on template attack (also known as profiled attack), more details of this type of attack can be found in Section 2.5.1.

When the power analysis methods were initially proposed, they mainly focused on the dynamic power behavior when the internal state of circuits change. The total power consumption of a chip consists of two parts, dynamic power and static power. The static power is the power consumption when there is no state change in the circuit. Let the current flowing into the circuit be I_{static}, the static power can be measured as

$$P_{static} = I_{static} V_{dd}, \tag{1.1}$$

where V_{dd} represents the power supply voltage.

It is predicted by International Technology Roadmap for Semiconductors (ITRS) that the static power consumption can be significant in technologies in sizes of 90 nm and less [12]. Additionally, with the shrinkage of technology size, the increase in static power is more significant than the increase in dynamic power, making static power a potential source for power analysis. We will refer to the power analysis attacks using static power as the measurement target as *static power analysis* (although it is also referred to as leakage power analysis in [13] and [14]).

Various researchers have shown how to apply existing power analysis algorithms to static power and the practicality of static power analysis has been demonstrated in different types of platforms from simulations of CMOS circuits to Field Programmable

Gate Arrays (FPGAs). A DPA attack against the DES cipher using static power leakage was performed in [15], successfully showing that static power can give better attack results than dynamic power in 180-nm technology. More recent publications mainly use CPA as the attacking method. The static power behavior of 90-nm and 65-nm devices was analyzed by Alioto et. al. in [13], as well as the performance of CPA using static power on cryptographic circuits. After this, they further explored the effectiveness of static-power-based CPA against circuits enhanced with countermeasures for classical DPA attacks in [14]. The CPA in [13] and [14] are performed in a CMOS simulation environment. Moradi applied static-power-based CPA on real-world FPGA chips in [16], and static power analysis was shown to be successful against some masked AES S-box implementations. Del Pozo et. al. argued in [17] that the performance of static power analysis may not be better than that of dynamic power analysis, and recently Bellizia et. al. [18] proposed to perform multivariate template analysis on static power by varying the temperature.

1.3 Application of Machine Learning

In recent years, machine learning has become yet another heated topic in the SCA community. The set of *classification* problems in the machine learning domain maps directly to the category of *profiled attacks* in SCA algorithms. Hence, a lot of existing machine learning algorithms can be applied to SCA [19–23]. A comparison between machine learning techniques and the classical template attacks is given by Lerman et. al in [24]. More recently, the application of deep learning algorithms has emerged as a new potential direction. Maghrebi et. al. compared the performance of several different deep learning algorithms by applying them to the implementations of both unmasked and masked AES implementations on FPGA in [25]. Han et. al. attacked

the electromagnetic leakage of a PLC device by means of the long short-term memory (LSTM) network in [26]. Recently, Hettwer et. al. proposed to use the plaintext information to boost the performance of convolutional neural networks for power analysis in [27].

It was previously suggested in some publications [28, 29] that combining different leakage sources may improve the performance of SCA. Power consumption and electromagnetic radiation are used as the leakage sources for combination. However, the traces from different leakage sources are straightforwardly concatenated as the input to conventional template attack algorithms. Now with the help of deep learning techniques, we are able to develop more advanced ways of combining the leakage sources.

1.4 Our Work

In our research, we apply profiled attacks to static power of simulated CMOS devices in sub-90-nm technologies. Our target devices are low frequency devices using light-weight block ciphers. In the early period of our research, we verified the property of static power leakage on some small circuit components and analyzed its relation with input/output. Then we improved the conventional template attacks with our proposed distinguishers and verified them on the static power leakage of cryptographic circuits. After this we made a further attempt to optimize the performance of template attacks in the scenario of static power analysis by means of kernel-based methods. Later we compared the performances of static and dynamic power analyses with a masked cryptographic circuit. After this we began considering combining "the benefits of two worlds", which is combining the static power leakage and the dynamic power leakage to get a better performance. By means of deep learning techniques, we are able to

propose several combined schemes.

1.5 Thesis outline

This thesis is organized in the following manner. In the Chapter 2 we are going to introduce some necessary background used in our research and in Chapter 3 describe the target circuits of our research. Then in Chapter 4 we show some earlier research we have done to verify and improve previous static power attack methods. Next, in Chapter 5 we show our work in the kernel-based static power attacks. Then we compare the performance of static- and dynamic-power based attacks using a protected circuit in Chapter 6. Finally in Chapter 7, we demonstrate our proposed schemes to combine static and dynamic power in order to improve the attack results.

Chapter 2

Backgrounds

In this chapter, we give an introduction to the background used in our research. We focus on static-power-based attacks whereas most previous work focuses on dynamic power. First we will introduce some cryptographic algorithms, which may be the potential target for SCA. After this we will give a definition of static power leakage, and then we will introduce the methods used to measure the static power leakage. After this we will introduce some of the existing side-channel attack algorithms and their application in the static power domain.

2.1 Block Cipher Algorithms

An easy approach to understand cryptographic algorithms is to look into how they are used. Let Alice and Bob be two people who need to communicate via an insecure channel. The meaningful content that needs to be transmitted is referred to as plaintext p. A malicious person, Oscar, is trying to capture p, and Oscar is able to capture the information in the insecure channel. Hence p cannot be directly transmitted. Alice has a key k_1 and uses an encryption function to encrypt p as $enc(p, k_1) = c$. The ciphertext c is meaningless if not decrypted, so it can be transmitted through the insecure channel as it is secure even if Oscar captures c. When Bob receives c, he has a key k_2 which can be used to decrypt the information as $dec(c, k_2) = p$. Hence plaintext p can be securely transmitted via an insecure channel.

Major cryptographic algorithms can be divided into symmetric key ciphers and public key ciphers. Symmetric key ciphers use the same key for encryption and decryption, in other words, $k_1 = k_2$. Public key ciphers use different keys for encryption and decryption, usually referred to as the public key and secret key. The public key and secret key are generated by one communication party, and the public key is transmitted to the other party in an insecure channel. In this thesis we focus on the block cipher category in symmetric key ciphers, which processes a fixed length of plaintext data, also known as a block, in each encryption.

2.1.1 Substitution-Permutation Network

Most modern block ciphers are in the structure of product ciphers, which means they are constructed by combining multiple cryptographic operations. A substitution-permutation network (SPN) is a structure widely used for product ciphers.

The main components of an SPN cipher are two types of operations referred to as substitution and permutation. These two operations can be defined as following [30,31].

Definition 2.1.1. Let there be an l-bit plaintext. Substitution is defined as a mapping $\pi_S : \{0,1\}^l \to \{0,1\}^l$, which uniquely replaces the plaintext by a corresponding l-bit ciphertext.

Definition 2.1.2. Let there be ml-bits of input plaintext. Permutation is a mapping $\pi_P : \{1, ..., ml\} \to \{1, ..., ml\}$, which changes the order of bits in the plaintext. Regardless of the order change, no bit is added, deleted, or replaced in this operation.

The module that performs the substitution operation is commonly called substitution-box (S-box). Here we restrict the space of plaintext and ciphertext to blocks of bits because we are focusing on hardware implementations of ciphers. S-boxes can be

1: $w_0 \leftarrow x$
2: **for** $r \leftarrow 1$ to $N-1$ **do** ▷ Rounds 1 to $N-1$.
3: $\quad u_r \leftarrow w_{r-1} \oplus k_r$
4: $\quad$ **for** $i \leftarrow 1$ to m **do**
5: $\quad\quad v_r[i] \leftarrow \pi_S(u_r[i])$
6: $\quad w_r \leftarrow \pi_P(v_r)$
7: $u_N \leftarrow w_{N-1} \oplus k_N$ ▷ The last round.
8: **for** $i \leftarrow 1$ to m **do**
9: $\quad v_N[i] \leftarrow \pi_S(u_N[i])$
10: $y \leftarrow v_N \oplus k_{N+1}$

Figure 2.1: An SPN algorithm.

implemented as a separate logical unit in a circuit, and permutations can be implemented as simple wiring. In general, these operations can be applied to other spaces such as alphabets, strings, or vectors.

An SPN cipher is constructed with iterations of substitution and permutation, along with key mixing added in every round of the iteration. Here we describe a typical SPN cryptosystem. Let the size of an S-box be l. If there are m S-boxes in parallel in a substitution process, the size of plaintext and ciphertext is lm. The size of permutation is lm, as well. Let there be N rounds of encryption. An SPN that takes $x \in \{0,1\}^{ml}$ as input and produces $y \in \{0,1\}^{ml}$ as output is shown in Figure 2.1 [30]. Here r refers to the current round number, k_r refers to the key used in current round, w_r is the output from round r, and $\oplus$ refers to the bit-wise XOR operation. The intermediate values in round r are represented by u_r and v_r, and the intermediate values processed by the i-th S-box is represented by $u_r[i]$ and $v_r[i]$.

An SPN cipher satisfies the *diffusion* and *confusion* standard proposed by Shannon in [32]. Diffusion and confusion refer to the methods used to frustrate the attacking process. Diffusion tries to hide the statistical relation between plaintext and ciphertext, while confusion tries to hide the statistical relation between ciphertext and key. In an

SPN cipher, diffusion is achieved by having multiple permutations, and confusion is achieved by having complex nonlinear functions for the S-box.

Note that the SPN structure we show here is one possibility, however modifications such as using multiple S-boxes are possible in different cryptosystems.

2.1.2 Key Schedule

Common block ciphers take a key K as input for each plaintext block. The subkeys k_r for each round are generated from K via the key schedule process. Usually, the key schedule goes through a few iteration rounds. This could be done in parallel with the encryption rounds, so when the circuit is running encryption iterations, the subkeys for each encryption round are generated on-the-fly. Otherwise, key scheduling can also be done ahead of the encryption process, which requires extra registers to store the subkeys.

The values processed in the key schedule are directly related to subkeys. In a side-channel attack scenario, unprotected key scheduling is likely to leak information which has strong correlation with subkeys. Properly exploiting the key schedule can increase the chance of a successful attack.

2.1.3 PRESENT Cipher

AES has been a widely accepted block cipher. However, hardware implementations of AES require a large amount of resource. Environments such as RFID tags and sensor nodes may not have sufficient hardware resources for a complete AES encryption. Lightweight ciphers are proposed for devices that have limited hardware resources. PRESENT was proposed by A. Bogdanov et. al in [33], and it has been widely accepted as a lightweight SPN cipher.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2.1: The S-box used in PRESENT.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 2.2: The permutation layer in PRESENT.

PRESENT is a block cipher that supports 64-bit plaintext/ciphertext size and two key sizes, 80 and 128 bits. This cipher consists of 31 rounds, and in each round there are 3 transformations in sequential order:

- **addRoundKey:** Bit-wise XOR of the state and the round key.
- **sBoxLayer:** A substitution operation that uses a 4-bit S-box to transform the 4-bit blocks.
- **pLayer:** A permutation operation that shifts the order of bits.

Here we use a function $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ to represent the S-box function in PRESENT. The S-box is shown in Table 2.1.

Consider a bit in the state at position i, and let its permuted position be $P(i)$. All the possible positions in pLayer are shown in Table 2.2.

Another addRoundKey transform operation is performed after all the 31 rounds are finished. Hence 32 round keys are generated in the key schedule, while each round key has 64 bits. The key schedule algorithm is shown in Figure 2.2. Here *key* is the key input, k is the round key generated in each round, and $S(\cdot)$ is the S-box function.

1: **function** KeySchedule(**bit** $key[80]$, **bit** $k[64]$)
2: **for** $i \leftarrow 1$ to 32 **do**
3: $k[63, 62, \cdots, 0] \leftarrow key[79, 78, \cdots, 16]$
4: $key[79, 78, \cdots, 0] \leftarrow key[18, 17, \cdots, 0, 79, 78, \cdots, 19]$
5: $key[79, 78, 77, 76] = S(key[79, 78, 77, 76])$
6: $key[19, 18, 17, 16, 15] = key[19, 18, 17, 16, 15] \oplus i$

Figure 2.2: PRESENT key schedule function.

The values in the square brackets are used to represent the indexes of corresponding bits.

2.2 Static Power Leakage

As (1.1) shows, the static power leakage P_{static} is determined by static current I_{static} and power supply V_{dd}. Commonly V_{dd} for a given device is fixed, so P_{static} is determined by I_{static}. The static current I_{static} of a CMOS component can be divided into four major parts: [12]

- Sub-threshold Leakage I_{sub}: the current from drain to source when the MOSFET is not fully turned off.
- Gate Leakage I_{gate}: the current that flows from gate to the body of MOSFET.
- Gate Induced Drain Leakage I_{gidl}: the current from drain to body when there is a high drain to gate voltage V_{DG}.
- Reverse Bias Junction Leakage I_{rev}: the current caused by the effects of minority carrier and election/hole pairs in depletion regions.

Among these, I_{sub} is the major contributor to I_{static}. As a proof for this, we simulated an NMOS transistor in 90 nm. When $V_G = V_S = V_B = 0, V_D = 1.2\,\mathrm{V}$, the measurement results are $I_{sub} = 3.431\,\mathrm{nA}, I_{gate} = 84.223\,\mathrm{pA}, I_{gidl} = 1.265\,\mathrm{pA}$, and

$I_{rev} \approx 0$. Hence we could model the growth trend of P_{static} by focusing on I_{sub}, which can be approximated as [12]

$$I_{sub} = \mu C_{OX} V_{th}^2 \frac{W}{L} \cdot \exp \frac{V_{GS} - V_T}{n V_{th}}, \tag{2.1}$$

where μ and n are technology-dependent parameters, W and L represent width and length of the transistor, C_{OX} is the capacitance of the oxide layer, V_T is the threshold voltage, and V_{th} is the thermal voltage, calculated as [12]

$$V_{th} = kT/q, \tag{2.2}$$

where k is Boltzmann's constant, T is the absolute temperature, and q is the absolute value of electron charge. In a simulated environment, these parameters are reflected by the parameters in the design flow as well as the standard cell library. Hence the static power leakage of a circuit is theoretically a fixed value if the circuit remains in the same fixed state and the configuration of the standard cell library is not changed. If the input/output of the circuit is changed and the circuit remains in the same environment, then the only parameter that will possibly change is V_{GS}. As conclusion, a given input/output pair should result in a fixed static current leakage for the same circuit in the same environment. Hence, theoretically, in a synchronous sequential logic system, one sample per stable clock level should be enough to capture the static power leakage.

2.3 Measurement

When performing power analysis against cryptographic hardware, the first step is to acquire traces. A trace is a sequence of measurements taken over a period of time

during the execution of cryptographic operations. The measurement points used for attacks are usually referred to as points-of-interest (POIs). The measurement target can be voltage, current, or power. Power cannot be directly measured in a real-world scenario, but usually simulation tools provide the feature to calculate a power trace automatically.

In order to measure the static power leakage of a circuit, the input values, including the clock, and measurement target should be static. Due to the capacitance of the whole circuit, we need to wait for the measurement target to settle to a steady state after input changes. As is shown by our simulation, in a 65-nm environment, it takes 3 ns to 30 ns for a 2-input NAND gate to settle to a steady state when the input changes depending on the different input values. The capacitance of circuit grows larger when gate count increases. The settling time for larger circuits could be longer than this.

In order to deal with this settling time problem, we could execute the circuit for a few clock cycles then stop the clock when the inputs reach a desired value. After a long enough period of time, this will produce a very clear static current leakage which can be easily measured. This methodology is used in [17], and can be achieved in a simulation environment very easily. In a real-world scenario, the adversary may not be able to fully stop the clock, but many devices provide different clock configurations, which means the clock can be slowed down in order to ensure longer clock period, allowing more time for the measurement target to settle down. A few methods are proposed in [13] to guess the timing of the clock cycle of the target circuit in the scenario that the clock cannot be fully stopped. If simple guesses cannot get the correct clock cycle, the adversary may guess a few candidate clock cycles and perform the attack using the candidate clock cycles. The correct clock cycle should have the best attack result. When the clock cycle is determined, the adversary can measure

the leakage current or voltage after a period of time into each clock cycle, assuming that the current has settled at the measurement time.

Here we can see one of the benefits of static power measurement. In static power analysis, only a single value needs to be measured for each input value. Comparatively, in order to record the dynamic behavior of the circuit, dynamic power analysis always needs to use an oscilloscope with high precision to do transient measurement. The cost for setting up a measurement environment using static power analysis will be cheaper. Besides, the measurement result for static power analysis has fewer points in a trace compared with that for dynamic power analysis, which means it will need much less storage space and less calculation for the same number of traces.

2.3.1 Simulation Measurement Method

In a simulation environment, there are three levels of power measurement that can be obtained.

First is counting the number of switching events in a given period of time. The power consumption for a single gate is from a look-up table, and the same type of gates are assumed to consume the same power. This method is the fastest but the most inaccurate, and is mostly used for estimating the overall power consumption.

The second method uses a testbench to execute the target circuit, then monitors the gate status after every given time step. This method is more precise than the first one, and is widely accepted in dynamic power analysis scenario.

The third method also uses a testbench. The target circuit is simulated in SPICE level. The power simulation tool calculates the power consumption at each time step according to the SPICE model for each MOSFET component. This method is able to simulate the effect of placement and routing. It can provide the most accurate result,

but consumes more time than the previous two methods.

For all the measurements in our research, we use the third method unless we specify differently. In order to be consistent with real-world measurement, our measurement target is the current at the terminal of power supply. We will give a detailed explanation of how these methods can be applied in our design environment in Section 4.1.

2.3.2 Real-World Measurement Method

Real-world measurement refers to the measurement of cryptographic devices such as FPGAs, smart cards, and microcontrollers.

In order to measure power consumption, the common practice is to place a small resistor r (say, $1\,\Omega$) in series with V_{dd} or ground. Then the voltage V_m is measured across r with an oscilloscope. The static power consumption of the target circuit is

$$P_{static} = V_{dd} \cdot \frac{V_m}{r}. \tag{2.3}$$

In fact, it is not necessary to calculate the actual power consumption. Current is sufficient to represent the power behavior of a cryptographic device. Figure 2.3 is from the data set of DPA Contest v2 [34]. It is a power trace of a 10-round AES-128 encryption process, operated on a FPGA board. The X axis is the measurement count. There are 3253 measurement points in total, with a timing step of 0.2 ns. The Y axis is the voltage level measured across a resistor (proportional to current), with each step as $1.413\,51 \times 10^{-5}\,\mathrm{V}$. Different operations result in different shapes in the trace, while the shapes for similar operations are similar.

Note that as is shown in [35], the measurement results for a simulation environment and a real-world environment can be quite different, even if the circuit is exactly the same. As technology size shrinks, the manufacturing process becomes harder to get

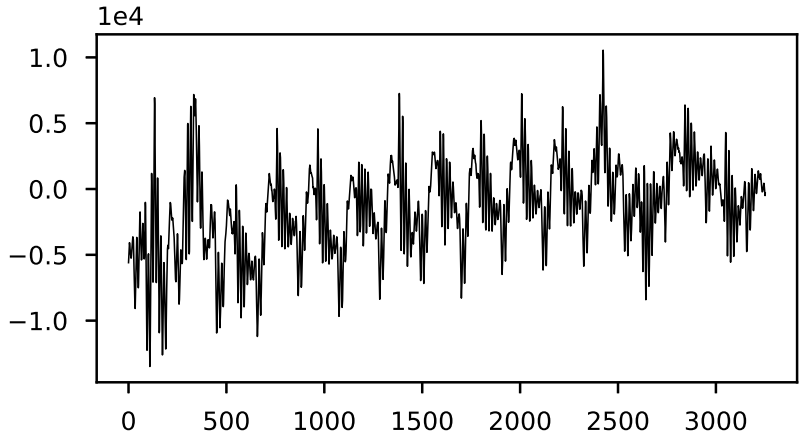


Figure 2.3: Power trace of AES-128 encryption. Data from [34].

identical chips for a single design. Inter-die and intra-die process variations cause larger measurement differences.

2.4 Preprocessing of Trace Data

Due to the measuring environment and power analysis requirements, an extra preprocessing step may be needed to process the measured traces. This step could be used to ensure better attack results or process the data into the format required by specific power analysis methods.

Real-world measurements contain noise from many sources, e.g. environment, other chip components, and the probe. Also the measuring time may not be perfectly aligned, which means the n-th measurement point in trace a may not be exactly the n-th measurement point in trace b. The noise and alignment errors need to be filtered out in the preprocessing step.

In the scenario of static power analysis, only one measurement point is required for each input state. However, in order to accurately capture the power consumption of the target chip, the oscilloscopes used usually have a much larger bandwith than

the clock frequency of the target chip, which means that multiple measurement points are acquired for each combination of input values. These multiple points need to be combined as one point.

A simple and commonly used preprocessing technique is averaging, which means straightforwardly taking multiple measurement points and computing their average [10]. Some recent proposals also suggest using frequency domain techniques to preprocess the data, e.g. Fourier transformation [36] and wavelet transformation [37].

2.5 Static Power Analysis Methodology

The term *static power analysis* is used to describe the power analysis methods targeted at static power leakage. This refers to a set of algorithms that can be applied to the same target, unlike the terms DPA and CPA, which refer to a specific algorithm. Existing static power analysis methods include the algorithms adapted from DPA [15] and CPA [13,14,16,38]. There exists the possibility that other algorithms like template attack [11] and MIA [5] may also be applied to static power analysis.

Here we introduce our proposed method of applying the template attack to static power leakage, as well as known practical issues.

2.5.1 Profiled Side-Channel Attacks

Profiled SCA refers to a set of attack algorithms that require an additional profiling device. A classical example of such attacks is the template attack [11]. Here we explain the mechanism of profiled SCA by mainly referring to the template attack.

Profiled SCA differs from CPA and DPA in the sense that aside from the target device, it requires an extra profiling device that is similar in behavior to the target device. It is assumed that the attacker has full control over the profiling device, which

means that the attacker is able to change the key used for encryption. The target device which is under attack has a key that is unknown to the attacker. The attacker uses the profiling device to build statistical profiles (also known as templates) for each possible key, then finds the profile which is closest to the statistical pattern acquired from the target device. The attacking procedure is divided into two phases: the *profiling phase* and the *attacking phase*.

Let there be K different key possibilities. In the profiling phase, the attacker performs M_p^k encryptions using the profiling device for each key possibility $k, 1 \leq k \leq K$. Usually, we let $M_p^1 = M_p^2 = \cdots = M_p^K = M_p$ for simplicity. The plaintexts for these encryptions are randomly chosen. The power consumption in each encryption is measured as a trace with N measurement points. In other words, the power consumption for an encryption is represented by an N-dimensional vector. Then for each key possibility the attacker is able to build a statistical model based on the distribution of a set with M_p^k traces, referred to as a profile set or template set. Some preprocessing is usually required before building the distribution models. The purpose of preprocessing may include: a) reducing the noise, b) reducing the number of measurement points, also known as dimensionality reduction, and c) aligning the measured traces, which may be required in scenarios where the measurement points in different traces are not triggered at exactly the same time.

The preprocessed traces are used to build the statistical models for each key possibility. In template attacks the models are build by estimating the probability density functions (PDFs) of each set using multivariate Gaussian distribution. Given a set of traces, let the i-th captured trace using key k be $\mathbf{t}_{k,i}$, we compute the mean vector $\bar{\mathbf{t}}_k$ and covariance matrix $\mathbf{\Sigma}_k$ as follows:

$$\bar{\mathbf{t}}_k = \frac{1}{M_p} \sum_{i=1}^{M_p} \mathbf{t}_{k,i}, \tag{2.4}$$

$$\mathbf{\Sigma}_k = \frac{1}{M_p - 1} \sum_{i=1}^{M_p} (\mathbf{t}_{k,i} - \bar{\mathbf{t}}_k)(\mathbf{t}_{k,i} - \bar{\mathbf{t}}_k)^T. \tag{2.5}$$

Here $\bar{\mathbf{t}}_k$ and $\mathbf{\Sigma}_k$ are referred to as template parameters. The probability distribution function (PDF) for this multivariate Gaussian distribution is

$$f(\mathbf{t}|\bar{\mathbf{t}}_k, \mathbf{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}_k|}} \exp\left(-\frac{1}{2}(\mathbf{t} - \bar{\mathbf{t}}_k)^T \mathbf{\Sigma}_k^{-1} (\mathbf{t} - \bar{\mathbf{t}}_k)\right). \tag{2.6}$$

In the attacking phase, the attacker performs another M_t encryptions with random plaintext inputs using the target device. By taking measurements the attacker would acquire another set, named the target set, of M_t traces each having N measurement points. After applying the same preprocessing as the profiling phase, the attacker then tries to find a profile set which is closest to the target set. In a successful attack, the key used to generate this profile set is the same as the key in the target device. Different distinguishers can be used to compare the target set with the profile sets. In [39], we have introduced the metrics used to compare the statistical distance between the sets. In template attacks, the maximum likelihood is used to find the most likely key.

In this work, we are going to use the maximum likelihood [40] to distinguish the key possibilities, hence we describe the details of its application in template attacks below.

Given a trace $\mathbf{t}_i (1 \leq i \leq M_t)$ in the target set, according to Bayes' rule [41], the estimated probability that this trace is generated using key k is

$$\hat{\Pr}(k|\mathbf{t}_i) = \frac{\hat{f}(\mathbf{t}_i|k) \Pr(k)}{f(\mathbf{t}_i)}, \tag{2.7}$$

where $\hat{f}(\mathbf{t}_i|k)$ represents the estimated probability density of $\mathbf{t}_i$, and the estimation is derived from the profile set generated using key k. $\Pr(k)$ is the probability that key k is the correct key and $f(\mathbf{t}_i)$ is the PDF of $\mathbf{t}_i$. It is assumed that the possible values in k and $\mathbf{t}_i$ both have equal probability of been used, hence k and $\mathbf{t}_i$ satisfy uniform distribution. Since $\Pr(k)$ and $f(\mathbf{t}_i)$ are fixed values, we can define the likelihood that key k is used as

$$\mathcal{L}_k = \prod_{i=1}^{M_t} \hat{f}(\mathbf{t}_i|k) \propto \prod_{i=1}^{M_t} \hat{\Pr}(k|\mathbf{t}_i). \tag{2.8}$$

Then the predicted key is

$$\tilde{k} = \arg\max_k \mathcal{L}_k. \tag{2.9}$$

If the predicted key is the same as the actual key used in the target device, then this attack is successful.

The key mentioned here is not necessarily the key input to the circuit. In practical attacks it is usually an intermediate value. For example, the round key bytes are usually used to build the profile sets when attacking an AES implementation.

2.5.2 Issues with the Maximum Likelihood Distinguisher

The distinguisher based on Gaussian distribution in (2.7) is first proposed by [11], and it has been widely used [40,42]. However, this distinguisher has multiple practical issues concerning the performance of matrix inversion, data type overflow in multiplication of big matrices, distinguisher accuracy etc., that has lead researchers to consider other tools as shown in [41, 43].

For a static-power-based profiled attack, the number of chosen POIs is much smaller than that in a dynamic-power-based attack. For example, less than 40 POIs are selected in our experiments. Hence the size of the programming language's built-in

float data type is sufficient in our case. However, we do observe that if the number of POIs is too large, then the covariance matrix of the trace set generated using key k, or $\mathbf{\Sigma}_k$, could be singular, which means that it is non-invertible. In this case $|\mathbf{\Sigma}_k| = 0$ and $\mathbf{\Sigma}_k^{-1}$ cannot be computed.

This issue has been addressed and discussed in [41]. Covariance matrix $\mathbf{\Sigma}_k$ has to be full-ranked to ensure that it is invertible. $\mathbf{\Sigma}_k$ is the scaled summation of m matrices $(\mathbf{t}_{k,i} - \overline{\mathbf{t}}_k)(\mathbf{t}_{k,i} - \overline{\mathbf{t}}_k)^T$, while the rank of $(\mathbf{t}_{k,i} - \overline{\mathbf{t}}_k)(\mathbf{t}_{k,i} - \overline{\mathbf{t}}_k)^T$ is 1. It is proven in [44] that for two square matrices $\mathbf{A}$ and $\mathbf{B}$,

$$rank(\mathbf{A} + \mathbf{B}) \leq rank(\mathbf{A}) + rank(\mathbf{B}) \tag{2.10}$$

Hence in order to ensure that the rank of $\mathbf{\Sigma}_k$ is N, there has to be at least N square matrices. As is suggested in [41], $M_p > N$ is a necessary condition to ensure the non-singularity of $\mathbf{\Sigma}_k$. Practically this condition is not always sufficient. The claim that $M_p > N$ is based on the assumption that every square matrix is independent from each other, however we find that in our measured data some traces have very similar behavior and adding these traces would result in the case that $rank(\mathbf{A} + \mathbf{B}) < rank(\mathbf{A}) + rank(\mathbf{B})$. We find that in our practice $M_p > 4N$ guarantees the non-singularity of our measured data better.

If the attacker is unable to acquire enough number of traces, then $\mathbf{\Sigma}_k^{-1}$ can be computed as the Moore-Penrose pseudoinverse, and $|\mathbf{\Sigma}_k|$ can be computed as the pseudo-determinant. In this case (2.6) becomes the PDF of a degenerated multivariate Gaussian distribution. Another option is to use the diagonal covariance matrix. This option omits the covariance between different POIs and regards the POIs as a sequence of separate univariate Gaussian distributions.

2.5.3 Other Distinguishers for Profiled Attacks

Knowing two random variables P and Q represent some unspecified distributions, the purpose of distinguishers is to identify how much they differ from each other. We propose to use the distinguishers that use only the parameters such as mean and covariance to distinguish between different distributions. For practical purposes, if there are multiple attacking traces, the attacker would not have to manually loop through each trace to compute (2.7). Instead, the attacker would only need to generate the mean and covariance for the attacking set. The functions that compute these parameters are supported by many programming languages and has been well-optimized and automatically parallelized.

Regarding each set of profile traces and the attack traces as a distribution, there are in total $K + 1$ distributions. In the static-power-based template attack, different statistical metrics can be used as the distinguisher to evaluate the difference between the distribution of attack traces and the distribution of profile traces. Here we show two distinguishers we use in our experiments:

- Kullback-Leibler divergence (KL divergence) [45], also called relative entropy, is defined as

$$D_{KL}(P||Q) = \int_x p(x) \log \frac{p(x)}{q(x)}, \tag{2.11}$$

  here $p(\cdot)$ and $q(\cdot)$ represent the PDF functions of random variables P and Q respectively.

- Jensen-Shannon distance (JS distance) [46], which is based on the definition of KL divergence, is defined as

$$D_{JS}(P||Q) = \sqrt{\frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)}, \tag{2.12}$$

where $M = \frac{1}{2}(P + Q)$.

In Section 2.5.1, the Gaussian distribution is used to describe the distribution of POIs. If P and Q follow Gaussian distributions, let the mean and variance of P and Q be μ_P and σ_P^2 as well as μ_Q and σ_Q^2 respectively. Then $D_{KL}(P||Q)$ can be computed using these parameters. Given that the PDF of a Gaussian distribution with mean μ and variance σ^2 is

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.13}$$

the KL divergence from Q to P is

$$\begin{aligned}
D_{KL}(P||Q) &= \int_x p(x) \left(\log p(x) - \log q(x)\right) \\
&= \int_x p(x) \left(\frac{1}{2}\left(-\frac{(x-\mu_P)^2}{2\sigma_P^2} + \frac{(x-\mu_Q)^2}{2\sigma_Q^2}\right) + \log\frac{\sigma_Q}{\sigma_P}\right) \\
&= \frac{1}{2}\int_{x+\mu_P} p(x+\mu_P) \left(-\frac{x^2}{\sigma_P^2} + \frac{(x+\mu_P-\mu_Q)^2}{\sigma_Q^2}\right) + \log\frac{\sigma_Q}{\sigma_P} \\
&= \frac{1}{2}\int_{x+\mu_P} p(x+\mu_P) \left(-\frac{x^2}{\sigma_P^2} + \frac{x^2}{\sigma_Q^2} + \frac{2x(\mu_P-\mu_Q)}{\sigma_Q^2} + \frac{(\mu_P-\mu_Q)^2}{\sigma_Q^2}\right) + \log\frac{\sigma_Q}{\sigma_P} \\
&= \log\frac{\sigma_Q}{\sigma_P} + \frac{\sigma_P^2 + (\mu_P-\mu_Q)^2}{2\sigma_Q^2} - \frac{1}{2}.
\end{aligned} \tag{2.14}$$

Note that M in (2.12) is a mixture of Gaussians. Assuming equal weight of P and Q, if P and Q are univariate Gaussian random variables, then the mean μ and variance σ^2 of M are computed as

$$\mu_M = \frac{1}{2}(\mu_P + \mu_Q), \tag{2.15}$$

$$\sigma_M^2 = \frac{1}{2}\sigma_P^2 + \frac{1}{2}\sigma_Q^2 + \frac{1}{4}(\mu_P - \mu_Q)^2. \tag{2.16}$$

Knowing $D_{KL}(P||M)$ and $D_{KL}((Q||M))$, the JS distance can be computed using (2.12).

Letting $\mathbf{P}$ and $\mathbf{Q}$ be two multi-dimensional vectors that satisfy multivariate Gaussian distribution, and $\mathbf{M}$ be a Gaussian mixture of $\mathbf{P}$ and $\mathbf{Q}$, there is no closed form expression for $D_{KL}(\mathbf{P}||\mathbf{M})$ and $D_{KL}(\mathbf{Q}||\mathbf{M})$ [47]. The mean vector and covariance matrix of $\mathbf{M}$ needs to be approximated in order to compute the KL divergence between $\mathbf{M}$ and $\mathbf{P}$ and between $\mathbf{M}$ and $\mathbf{Q}$. Here we compute the mean vector and the covariance matrix for $\mathbf{M}$ as:

$$\overline{\mathbf{M}} = \frac{1}{2}(\overline{\mathbf{P}} + \overline{\mathbf{Q}}), \tag{2.17}$$

$$\mathbf{\Sigma_M} = \frac{1}{2}\mathbf{\Sigma_P}(\overline{\mathbf{P}} - \overline{\mathbf{M}})(\overline{\mathbf{P}} - \overline{\mathbf{M}})^T + \frac{1}{2}\mathbf{\Sigma_Q}(\overline{\mathbf{Q}} - \overline{\mathbf{M}})(\overline{\mathbf{Q}} - \overline{\mathbf{M}})^T. \tag{2.18}$$

The approximation method we use here is simple and quick to compute, but this method does not guarantee the best approximation result. If necessary, better alternatives are also provided in [48].

For multivariate Gaussians, the KL divergence is: [49]

$$D_{KL}(\mathbf{P}||\mathbf{Q}) = \frac{1}{2}\left(tr(\mathbf{\Sigma_Q^{-1}\Sigma_P}) + (\overline{\mathbf{Q}} - \overline{\mathbf{P}})^T\mathbf{\Sigma_Q^{-1}}(\overline{\mathbf{Q}} - \overline{\mathbf{P}}) - N + \log\frac{|\mathbf{\Sigma_Q}|}{|\mathbf{\Sigma_P}|}\right), \tag{2.19}$$

where $tr(\cdot)$ is the trace of the matrix, which is defined as the sum along diagonals of the matrix, and N is the dimension of the covariance matrices $\mathbf{\Sigma}_P$ and $\mathbf{\Sigma}_Q$, which is the number of POIs. Similar to the univariate Gaussian distribution case, the JS distance can also be computed using (2.12).

Both KL divergence and JS distance can give a value showing size of the difference

between two distributions. If the difference value is 0, it means that $\mathbf{P} = \mathbf{Q}$. Despite the fact that they can both be applied to template attacks, there are also differences between them. KL divergence is not symmetric, which means it is possible that $D_{KL}(\mathbf{P}||\mathbf{Q}) \neq D_{KL}(\mathbf{Q}||\mathbf{P})$. The range of KL divergence is between 0 and positive infinity. Hence, if the KL divergence between one template and the target set is extremely high, it could make the KL divergence between other templates and the target set to be hard to distinguish. In our practice, in order to add symmetric property to KL divergence, the actual distinguisher we use is the average of two KL divergences, also known as the Jeffreys divergence or J-divergence [50], which is computed as

$$D_J(\mathbf{P}||\mathbf{Q}) = \frac{1}{2}(D_{KL}(\mathbf{P}||\mathbf{Q}) + D_{KL}(\mathbf{Q}||\mathbf{P})). \tag{2.20}$$

There are still other methods that can add symmetry to the KL divergence, such as the resistor-average distance proposed in [51]. Note that it is not mandatory to keep the distinguisher symmetric. We use the J-divergence in order to implement a distinguisher which is comparable to the JS distance.

JS distance is symmetric, and it is proven in [50] that the range of JS distance is bounded. In fact, JS distance satisfies the definition of a statistical metric according to [46]. In our experiment, JS distance performs slightly better than the J-divergence, but it requires more computation time because of the calculation of $\mathbf{M}$. In Chapter 4 we will compare the performance between our proposed distinguishers and the original Gaussian distinguisher.

2.5.4 Dimensionality Reduction

Dimensionality reduction is one of the major purposes for preprocessing in profiled attacks, Two different methods are used in our research: principal component analysis

(PCA) [42] and Fisher's linear discriminant analysis (LDA) [52]. Here we introduce these two methods.

2.5.4.1 Principal Component Analysis

Principal component analysis (PCA) is a technique commonly used for dimensionality reduction and projection of multivariate data [42]. In our research, PCA is used for both dimensionality reduction and avoiding matrix singularity.

The PCA of N-dimensional data can be done by applying singular value decomposition (SVD) to its covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$, which results in

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{2.21}$$

where the columns of matrix $\mathbf{U} \in \mathbb{R}^{N \times N}$ are the left singular vectors, and the elements in the diagonal matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ are the corresponding singular values. Matrix $\mathbf{V} \in \mathbb{R}^{N \times N}$ is the matrix of right singular vectors, and we do not use it in this scenario. In order to reduce the dimensionality of the original data from N to M, we pick the singular vectors from $\mathbf{U}$ with the corresponding M largest singular values. Combining the singular vectors together we get a new matrix $\mathbf{W} \in \mathbb{R}^{N \times M}$ which projects the axes of the original N-dimensional space into the axes of a M-dimensional space. The original data can be projected to the new dimensionality by multiplying it with the projection matrix.

In our scenario, the data we need to process is labeled as different sets referred to as templates. In our work we have implemented three approaches to apply PCA to labeled data:

1. Combine all the data from different sets and compute the overall covariance matrix. Reduce the dimensionality of each set using an overall projection matrix.

2. Compute the covariance matrix for each different set, and project each set to the new axes using its own projection matrix accordingly.

3. Compute the average covariance matrix for all sets combined and use this to derive the projection matrix, which is then applied to each set. This method was proposed by C. Archambeau et. al. in [42].

In approach 3, the average covariance matrix is defined as follows. Let there be K sets of data. Compute the mean of each set as $\mathbf{t}_k, 1 \leq k \leq K$, and compute the average of the K means as $\overline{\mathbf{t}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{t}_k$. The average covariance matrix is given as

$$\mathbf{\Sigma} = \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{t}_k - \overline{\mathbf{t}}\right)\left(\mathbf{t}_k - \overline{\mathbf{t}}\right)^T. \tag{2.22}$$

This covariance matrix is used to compute the projection matrix, and the projection matrix is applied to each set.

In our experiments, we find no significant difference between the performance of the three approaches. Although we have decided to use the third approach to perform PCA in our attacks, we believe that similar results can be achieved using the other two approaches.

2.5.4.2 Linear Discriminant Analysis

Fisher's linear discriminant analysis [52] is yet another method that has been used in our research for dimensionality reduction when preprocessing the traces. Intuitively, LDA seeks a dimensionality projection that maximizes the variance between different sets, while PCA combines all sets together and seeks a projection that maximizes the variance of the combined set. It has been observed by several existing works [41, 53, 54] that LDA gains a better performance than PCA.

Here we give a brief introduction to LDA. Let the covariance matrix of each profiling set be $\mathbf{\Sigma}_k$, the mean in each profiling set be $\boldsymbol{\mu}_k$, and the mean of all these sets be $\boldsymbol{\mu}$.

The within-class covariance $\mathbf{\Sigma}_W$ is defined as

$$\mathbf{\Sigma}_W = \sum_{k=1}^{K} M_p^k \mathbf{\Sigma}_k, \tag{2.23}$$

and the between-class covariance $\mathbf{\Sigma}_B$ is defined as

$$\mathbf{\Sigma}_B = \sum_{k=1}^{K} M_p^k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T. \tag{2.24}$$

Similar to PCA, LDA seeks a dimensionality projection matrix that projects the original N-dimensional traces to a new dimensionality. An N-dimensional unit vector $\mathbf{u}_i$ in the projection matrix tries to maximize the ratio of $\frac{\mathbf{u}_i^T \mathbf{\Sigma}_B \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{\Sigma}_W \mathbf{u}_i}$.

2.5.5 Kernel Density Estimation

As we shall see in Chapter 5, in static power analysis, there are circumstances where a Gaussian distribution may not reflect the distribution of the trace data. In such cases, we consider kernel density estimation (KDE).

KDE uses kernel functions to model the probability density function (PDF) of the target data. A kernel function is added to the PDF estimation where there is a sample point x_i. The estimation result is a sum of kernel functions $\mathcal{K}(\cdot)$, represented in [55] as

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} \mathcal{K}\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{K}_h\left(x - x_i\right), \tag{2.25}$$

where h is referred to as the bandwidth and n is the total number of sample points.

For d-dimensional KDE, (2.25) becomes

$$\hat{f}(\mathbf{x}) = \frac{1}{n|\mathbf{H}|} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}_{\mathbf{H}}\left(\mathbf{x} - \mathbf{x}_i\right), \tag{2.26}$$

where $\mathbf{H}$ is a $d \times d$ bandwidth matrix composed of elements $h_{i,j}, 1 \leq i, j \leq d$. For simplicity of expressions, we use $\mathcal{K}_h$ and $\mathcal{K}_{\mathbf{H}}$ to represent kernel functions $\mathcal{K}(\cdot)$ with bandwidth parameter h and bandwidth matrix $\mathbf{H}$.

There are multiple choices for the kernel function $\mathcal{K}(\cdot)$. Commonly used functions include Gaussian, Epanechnikov, and cosine function. In our experiments, we have found that the kernel function can be arbitrarily chosen since we observe little difference when fitting our data using different functions. Similar observations are also stated in [56] and a theoretical analysis that proves the fitting performance of some commonly-used kernel functions are similar is given in [57].

The bandwidth parameter needs to be carefully chosen. Choosing it to be too large results in underfitting the PDF, while choosing to be too small causes overfitting. In our research we use Scott's rule of thumb, which in [55] is described as

$$h_{i,i} = \hat{\sigma}_i n^{-1/(d+4)}, \tag{2.27}$$

where $h_{i,i}$ is the diagonal element in the bandwidth matrix, and $\hat{\sigma}_i$ is the standard deviation of the i-th dimension of the target data. All the other elements in $\mathbf{H}$ are 0.

In our work, we choose the Gaussian function as the kernel function, since (2.27) is initially derived for the Gaussian kernel function in [57].

2.6 Deep-Learning-Based Side-Channel Attacks

Deep learning is used in our research to combine different leakage sources when performing a profiled attack. Here we give an introduction to the deep learning techniques we have used in our research.

2.6.1 Long Short-Term Memory Network

Long short-term memory [58] (LSTM) is a specific type of building unit for the recurrent neural network [59] (RNN), which belongs to the neural network category of machine learning.

A simple neural network unit, or cell, can be defined as

$$y = \sigma(Wx + b), \tag{2.28}$$

where x and y are the input and output of the unit, respectively, W is a weight parameter, b is a bias parameter, and σ is a nonlinear activation function. In a neural network there are multiple such units and together they form a hidden layer, which takes inputs from the input layer, and sends its outputs to the output layer.

There may be more than one hidden layer in a neural network structure, in which the outputs from the previous hidden layer are the inputs to the next hidden layer. Usually the number of units in a layer is referred to as the width of this layer, and the number of hidden layers is referred to as the depth of the network. The term *deep learning* is usually used to represent a class of machine learning algorithms which contain multiple hidden layers in the network.

Simple neural network units such as (2.28) have only one direction for the movement of its information, that is, from inputs to outputs. RNN is a set of neural networks

which contain loops in the hidden layers. The previous unit sends some intermediate value to the next unit in the same layer. Hence, the input to the previous unit will also have effect on the processing of the input to the next unit in the loop. It is believed that such structure would ensure that RNN is more capable of processing sequence data, in which the previous input may have certain logical relation with the next input. However, it was discovered [60] that in practice RNN is not capable of solving long-term dependencies between units. This is known as the vanishing and exploding gradient problem.

LSTM is proposed to solve such problems in RNN. By bringing the concept of cell state, LSTM handles long-term dependencies better than the original RNN.

In the LSTM unit, we let the input from previous hidden layer to the current unit be x_t, the cell state be C_t and the hidden state output be h_t. The cell state and the hidden state are used as the input to the next unit and the hidden state output also goes to the next hidden layer. The inputs and outputs are all vectors. We shall use the subscript $t-1$ to represent corresponding variables from the previous unit. We use W to represent the weight matrix and use b to represent the bias vector. The subscripts are used to differentiate the weights and biases in each layer component. The architecture of a LSTM unit can be represented by the following equations.

Firstly there is a forget gate layer defined as

$$f_t = \text{sigmoid}(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.29}$$

which discards the unwanted information. Here sigmoid$(\cdot)$ is a function defined in [61] as $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. To add new information to the cell state, an input layer i_t is used to select the information to be added, and a tanh function prepares the candidate

information vector $\tilde{C}_t$, defined as

$$i_t = \text{sigmoid}(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.30}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{2.31}$$

The $\tanh(\cdot)$ function is defined in [61] as $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Then the current cell state is

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t, \tag{2.32}$$

where $\circ$ represents element-wise multiplication. Finally the output of hidden layer h_t is controlled by an output gate layer o_t, both defined as

$$o_t = \text{sigmoid}(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.33}$$

$$h_t = o_t \circ \tanh(C_t). \tag{2.34}$$

We are applying LSTM in our work due to the nature that LSTM is capable of dealing with sequence data and power traces are a type of timing sequence data with underlying logical dependencies. The desired output from this network is the likelihood that the current trace is generated by the corresponding possible key. In order to translate the hidden layer outputs to a probability distribution model, we stack a dense layer that uses softmax function [61] to the output of the final LSTM layer. This layer is simply defined as

$$\hat{p} = \text{softmax}(W_d \cdot h_T + b_d), \tag{2.35}$$

where h_T refers to the hidden state output from the last LSTM unit in the hidden layer. Vector $\hat{p}$ is the estimated probability that each key candidate is the correct key.

In the true probability vector p, the correct key has a probability of 1 while all other keys have a probability of 0. To maximize the correctness of estimation, we select the cross entropy function as the loss function for our network, which is defined as

$$L(\hat{p}, p) = -\sum_{k=1}^{K} p_k \log \hat{p}_k. \tag{2.36}$$

By means of back propogation, the neural network tunes the weight and bias parameters in a given number of iterations, also known as epochs, in order to minimize the loss function.

2.6.2 Dropout

Ideally, we would arbitrarily increase the depth and width of our neural network. However this is not true in practice. Training with a deep network can quickly overfit the training data. In this work we use the dropout [62] mechanism to reduce the effect of overfit.

Dropout is a simple yet effective mechanism. It randomly discards part of units in the hidden or input layer. In implementation, a dropout layer is added after the hidden or input layer and the probability for a unit to be discarded can be set separately for each dropout layer. Setting the dropout rate too high would result in slow learning process, while setting it too low would have little effect on reducing overfitting. In our work, in order to reduce the work of parameter tuning and make the comparison more straightforward, we maintain the same discard rate for each dropout layer in a network. This is an optional layer, and is not used in other works that employ LSTM, such as [25, 26]. In our experiments, we found it necessary in addressing the overfitting problem with our training data.

2.7 Metrics

We now give a brief introduction to the metrics we are going to use in our work. We shall use the mean integrated squared error and Kullback-Leibler divergence to evaluate the statistical distance between distributions, use the signal-to-noise ratio to evaluate the hardness to attack our target device, and use the success rate to evaluate the performance of our attacks.

2.7.1 Mean Integrated Squared Error

As a commonly used metric for verifying the quality of PDF estimation, the definition of mean integrated squared error (MISE) between two PDFs f and g is given in [55] as

$$\mathrm{MISE}(f,g) = \mathrm{E}\left\{\int \left[f(x)-g(x)\right]^2 dx\right\} = \int \mathrm{E}\left\{\left[f(x)-g(x)\right]^2\right\} dx = \int \mathrm{MSE}_{f,g}(x)dx, \tag{2.37}$$

where $\mathrm{MSE}_{f,g}(x)$ is the mean square error (MSE) between f and g at sampling point x.

2.7.2 Kullback-Leibler Divergence

KL divergence is used to measure the extent to which PDF f differs from PDF g. It is defined in [63] as

$$D_{\mathrm{KL}}(f||g) = \int f(x)\log\frac{f(x)}{g(x)}dx. \tag{2.38}$$

Note that KL divergence is non-symmetric, which means that $D_{\mathrm{KL}}(f||g) \neq D_{\mathrm{KL}}(g||f)$. A simple solution to this is to compute the symmetric KL divergence, or the J-

divergence [48]:

$$D_{\mathrm{J}}(f||g) = \frac{1}{2}\left(D_{\mathrm{KL}}(f||g) + D_{\mathrm{KL}}(g||f)\right). \tag{2.39}$$

2.7.3 Signal-to-Noise Ratio

The idea of using signal-to-noise ratio (SNR) to evaluate the noise in power analysis attacks was first proposed by Mangard in [64]. Later Durvaux et. al. [65] extended SNR to the scenario of labeled data. Given K sets of power traces, with M traces in each set, the estimated SNR of observed data is defined as

$$\hat{\mathrm{SNR}} = \frac{\hat{\mathrm{var}}\left(\mathrm{signal}\right)}{\hat{\mathrm{var}}\left(\mathrm{noise}\right)} = \frac{\hat{\mathrm{var}}_k\left(\hat{\mathrm{E}}_m\left(\mathbf{t}_k^m\right)\right)}{\hat{\mathrm{E}}_k\left(\hat{\mathrm{var}}_m\left(\mathbf{t}_k^m\right)\right)}, \tag{2.40}$$

where $\mathbf{t}_k^m$ is the m-th trace in the k-th set and functions $\hat{\mathrm{var}}(\cdot)$ and $\hat{\mathrm{E}}(\cdot)$ compute the variance and mean of the sampled data. For these two functions, the subscripts m and k refer to performing this function on the set of traces which is acquired by iterating all possible m or k. In cases where there are multiple points in a power trace, (2.40) is computed for each point in the traces, hence the computed SNR result is a trace with the same number of points as the power traces.

2.7.4 Success Rate

We use the success rate proposed in [66] to evaluate the performance of an SCA algorithm. Let there be a distinguisher function D which returns the likelihood or probability that a key guess is the correct key given the leakage traces. Ranking all the key guesses by their likelihood/probability, the o-th order success rate is defined as the probability that the correct key is among the o highest ranked key guesses. A formal definition of the o-th order success rate is given in [66].

In our scenario we only use the first order success rate to evaluate our attack results. The success rate is computed as the number of attacks that successfully recover the correct key divided by the number of all attacks. Note that another metric named guessing entropy is also proposed in [66] for evaluating the performance of side-channel attacks. It is easy to show that guessing entropy is related to success rate [67], so in our work only success rate is used.

2.8 Conclusion

In this chapter, we have introduced the background knowledge we are going to use in our research. Some issues with the classical template attack are also discussed in this chapter. In the next chapter, we will introduce the target circuits that we attack in our research.

Chapter 3

Target Cryptographic Circuits

In this chapter we describe the cryptographic circuits we attack using our proposed template attacks. There are some other circuit components that we analyze but do not perform actual attacks on. These circuits are introduced in Chapter 4 separately.

3.1 PRESENT S-Box

The first target circuit is an unprotected implementation of the S-box of the PRESENT cipher [33]. The PRESENT cipher (standardized by ISO/IEC 29192-2:2012) is a lightweight block cipher designed for lower-end hardware environments such as RFID tags, smartcards and microcontrollers. The unmasked PRESENT S-box is a 4-bit to 4-bit substitution as is shown in Table 2.1.

The structure of this target circuit is shown in Figure 3.1, which contains the S-box followed by a 4-bit XOR operation, as well as three 4-bit registers. Similar 4-bit S-box structures have also been used in other publications in the field of static power analysis [13, 14, 18].

This circuit is implemented in two different ways in our research. In our earlier research shown in Chapter 4, we used the decoder-switch-encoder structure and hand-wired the gates, in order to get a bit-sliced implementation of the S-box. We shall introduce more details about this structure in Chapter 4. In later chapters, the final circuit is generated by CAD tools after synthesis and layout processes. The pre-

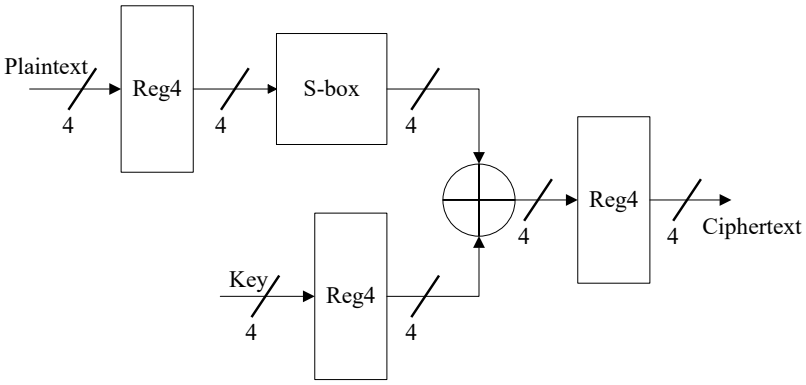


Figure 3.1: Structure of a S-box circuit.

synthesis RTL design of the S-box component is implemented using a straightforward look-up table without any countermeasures. The synthesized S-box component is comprised of only combinational logic. After performing layout, the post-layout netlist is imported into Virtuoso for simulation. The final post-layout circuit used for simulation is shown in Figure 3.2.

During the simulation process, the imported target circuit is tested by a testbench circuit. The target circuit is a component in the testbench, while another drive circuit generates the inputs to the target circuit and records the outputs from the circuit. The structure of these components is shown in Figure 3.3. The drive circuit is developed using VerilogAMS, and the outputs from the drive circuit are idealized. In other words, a digital 1 output from the drive circuit is fixed at 1.1 V as is shown in the figure. Similar testbench circuits are used for all other target circuits, and we set $V_{dd} = 1.1\,\mathrm{V}$ for all testbench circuits.

There are 3 clock periods (which includes 2 rising edges in between) needed for a full execution of this circuit. In the first period the inputs are sent to the registers. In the next period the inputs are stored in the input registers and the encryption results are generated. In the last period the ciphertext is stored in the output register. In order to best exploit the functionality of this circuit and to represent the realistic

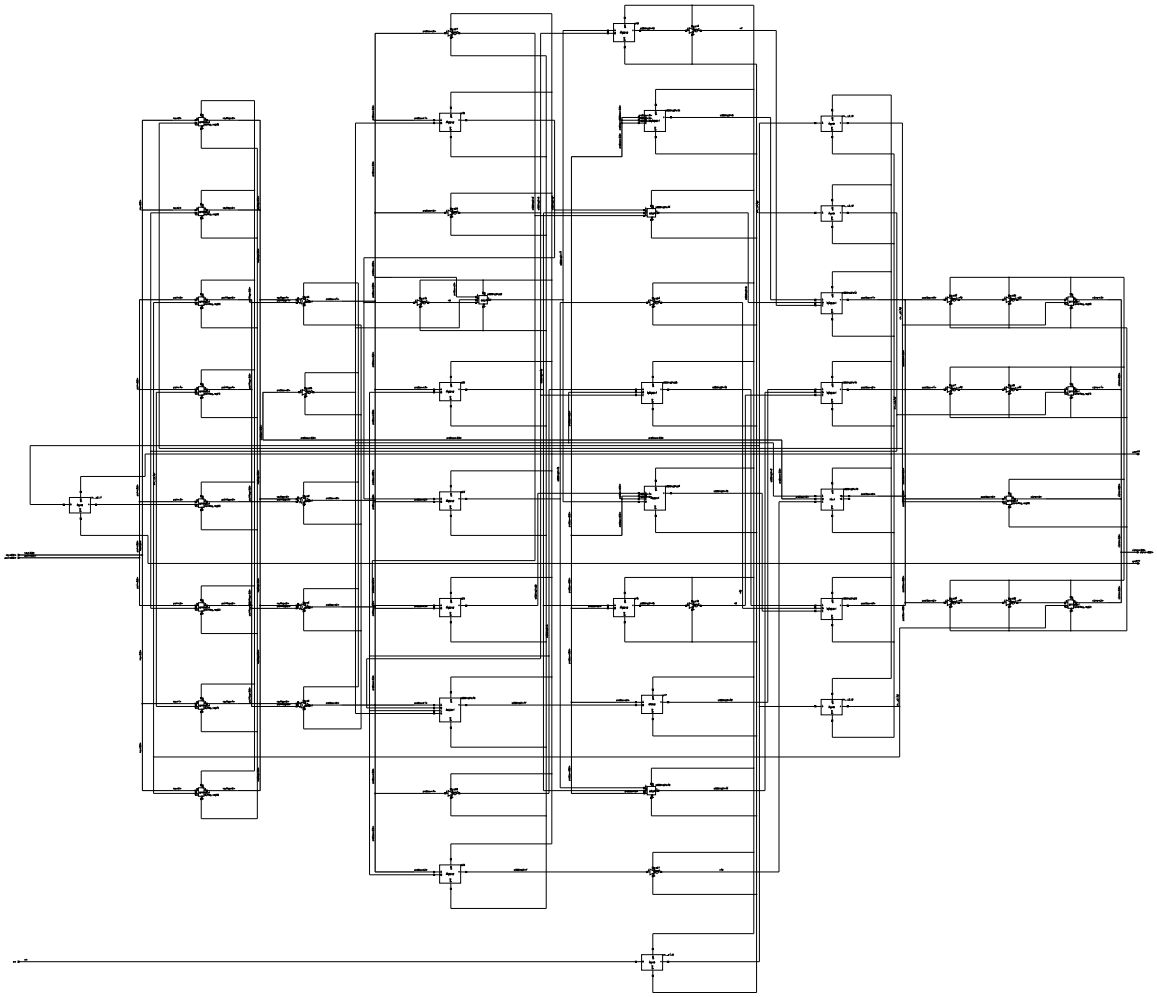


Figure 3.2: Post-layout netlist of the S-box circuit.

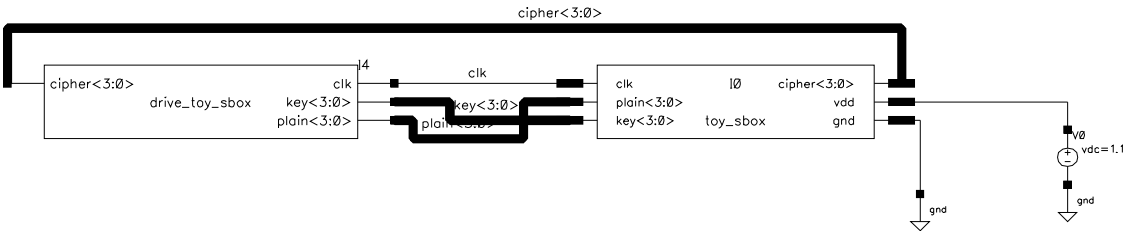


Figure 3.3: Testbench used to simulate the S-box circuit.

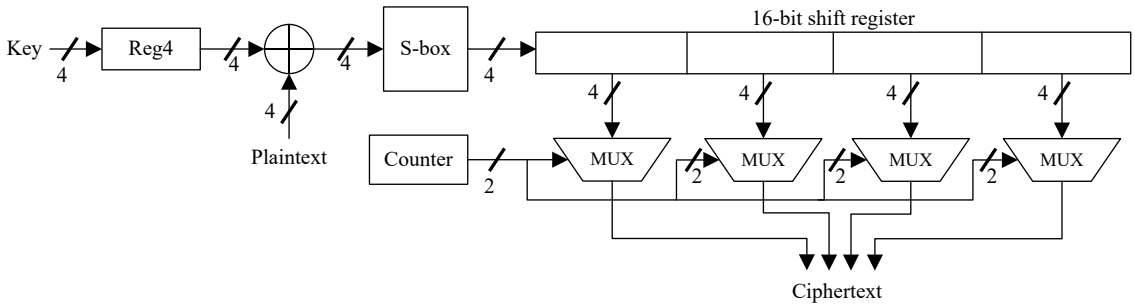


Figure 3.4: Structure of a serialized SPN circuit. The multiplexers are used to implement the permutation structure shown in Figure 3.5.

operation of the circuit, we focus on the second clock cycle and run the circuit in pipeline mode. The output of the previous encryption in its third cycle and the input of the next encryption in its first cycle contribute as noise to the overall power trace values correlated to the data manipulation of the second clock period. As described in Section 4.3, there are 2 POIs measured, for high and low clocks, when attacking the static power leakage of this circuit, which are taken in the second cycle of each encryption.

3.2 Serialized Substitution-Permutation Network

The second target circuit is a serialized implementation of a simple substitution-permutation network (SPN) cipher. Its structure is shown in Figure 3.4. After XORing the plaintext with a 4-bit key stored in the register, its substitution part incorporates a PRESENT S-box, and multiplexers are used to implement the permutation network shown in Figure 3.5. Similar to the S-box circuit, this circuit is also designed using the standard design flow. The circuit is synthesized from RTL level using Synopsys Design Compiler and the layout is designed using Cadence Encounter. The final post-layout netlist imported into Cadence Virtuoso is shown in Figure 3.6.

This cipher takes 8 clock periods to perform a full encryption. A 4-bit plaintext is

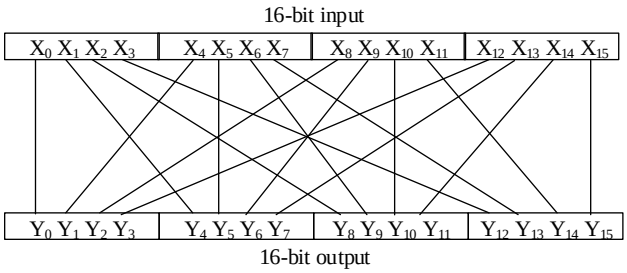


Figure 3.5: Permutation network used in the SPN circuit. The X labels refer to the input pins in the network, while the Y labels refer to the output pins in the network.

encrypted and shifted into the shift register in each of the first 4 clock periods. In the last 4 cycles, the multiplexers use the parallel output of the shift register to generate a 4-bit ciphertext output in each clock period. There are 16 POIs measured when attacking the static power leakage of this circuit, with each clock period containing 2 POIs.

In a real-world attack with actual chips, since the attacker has access to both the profiling device and the attacking device, it is reasonable to assume that in some scenarios the attacker has control over all the plaintext inputs, which is a chosen plaintext attack. However, we want to explore the more generic scenario of SCA in the context of static power, hence we pose a challenging assumption on the attack: we assume that the attacker has no knowledge or control of the plaintext. Hence the only information exploited in the attack comes from the power trace. Such an assumption would also make it impossible to perform an exhaustive key search even though the attack target is only 4 bits, since the attacker cannot know the exact plaintext/ciphertext. In our attacks, we randomly choose different plaintext inputs for each different template and the target set. The distribution of plaintext satisfies a uniform distribution.

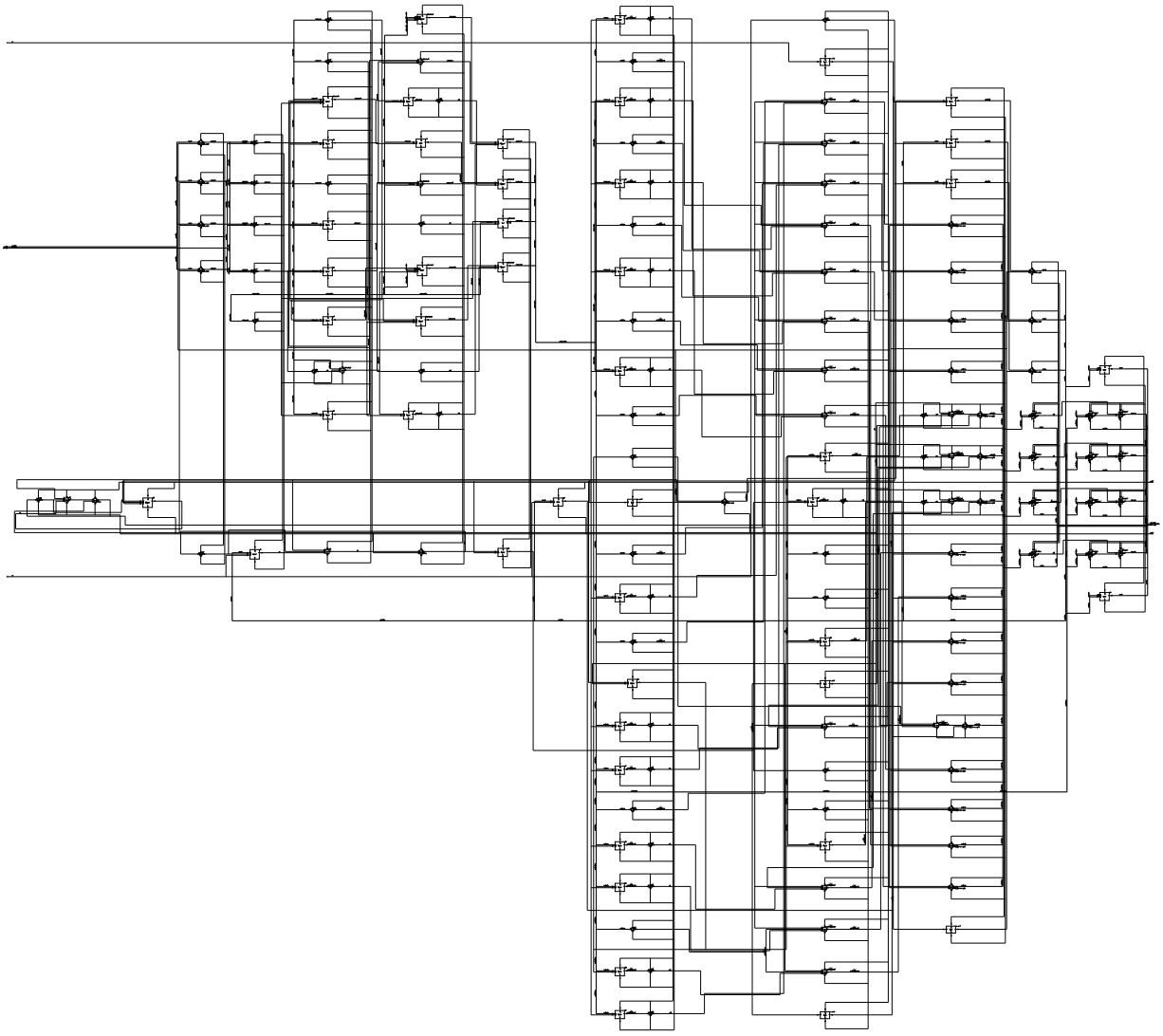


Figure 3.6: Post-layout netlist of the SPN circuit.

3.3 Simplified PRESENT Key Schedule Circuit

This target circuit is a simplified version of the PRESENT key schedule circuit proposed in [68]. To ensure that the simulation process is done in reasonable time, we implement this circuit with reduced rounds and fewer key bits. Specifically, we reduce the key size to 24 bits, and the round number to 2. Among the 24 bits stored in the key register, only the 4 least significant bits are used as the round key. We also used a 4-bit state register to store plaintext inputs. The value in the state register is XORed with the round key and then permuted. This provides minimal encryption capability. Since the key schedule of PRESENT is not affected by the plaintext input, the power consumption generated by the encryption logic is regarded as noise for our attack purpose. The final post-layout netlist used for simulation is shown in Figure 3.7.

It takes 21 clock periods to finish a key schedule process with two rounds of encryption. The functionality of the clock periods are:

- 0: plaintext and key inputs;
- 1-5: key inputs;
- 6, 13: XOR and S-box operation for the state register;
- 7-11, 14-18: shift the round keys bits to the original position;
- 12, 19: permutation of state register, round key generation;
- 20: XOR the value in the state register with round key and output the result as ciphertext.

Since 1 sampling point is taken per clock state, there are in total 42 sampling points that we take in our experiments.

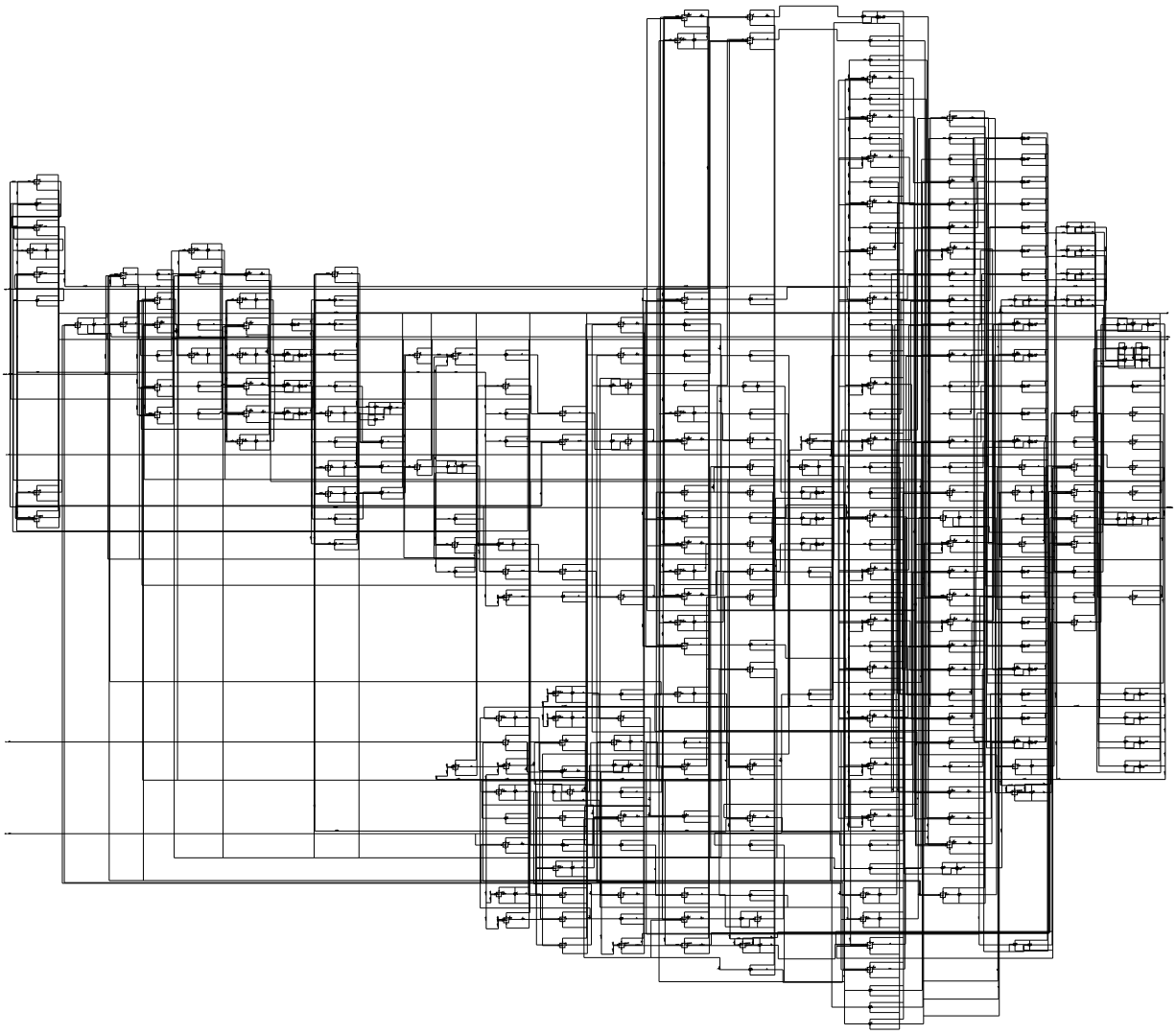


Figure 3.7: Post-layout netlist of the key schedule circuit.

3.4 Masked PRESENT S-Box

This target circuit is a masked implementation of the PRESENT S-box. The target circuit protects this S-box against power analysis attacks using dynamic power measurements with a masking scheme named threshold implementation [69]. This scheme masks the S-box by separating it into shares known as F-boxes and G-boxes. There are 3 F-boxes and 3 G-boxes needed to mask the PRESENT S-box. Different profiles for masking are proposed in [69]. In our research we target the *Profile 2*, which performs masking by sharing the data path.

Figure 3.8 demonstrates the detailed structure of this masked S-box implementation. Before the plaintext is stored in the state register, it is XORed with two randomly generated 4-bit data masks Md1 and Md2. The XORing result of state data and the key input as well as the masking data are used as the input to the G-boxes, marked as X1, X2 and X3. The outputs from the G-boxes are stored in three 4-bit registers labeled as Y1, Y2 and Y3, then the values in these registers are sent to the F-boxes to generate the final outputs S1, S2 and S3. The final outputs of this circuit satisfy

$$\text{S1} \oplus \text{S2} \oplus \text{S3} = \text{S-box}\left(\text{Plaintext} \oplus \text{Key}\right), \tag{3.1}$$

where S-box$(\cdot)$ refers to the unmasked PRESENT S-box, which is shown in Table 2.1. This circuit takes 4 clock periods to perform an encryption, hence there are 8 sampling points for the static power trace. The final post-layout circuit used for simulation is shown in Figure 3.9.

Here we list the detailed algebraic normal forms of the F-boxes and G-boxes defined in [69]:

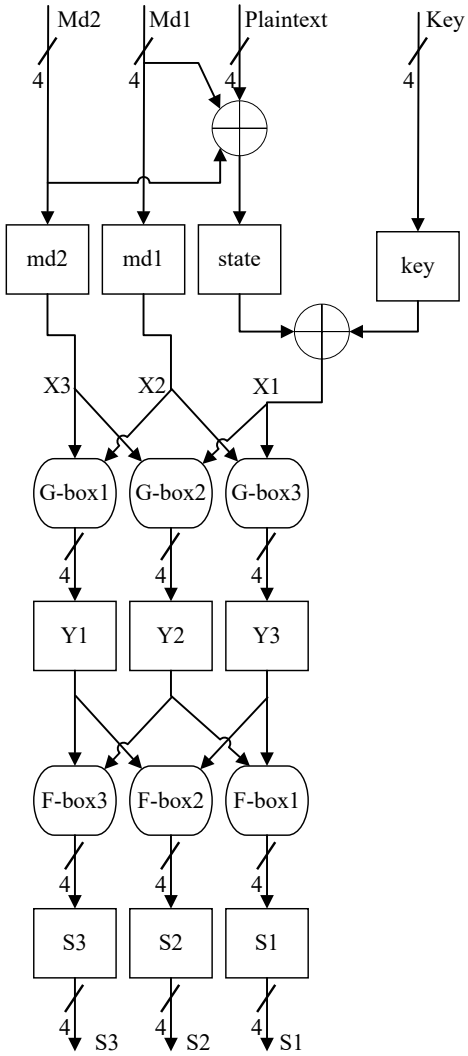


Figure 3.8: Structure of the masked PRESENT S-box.

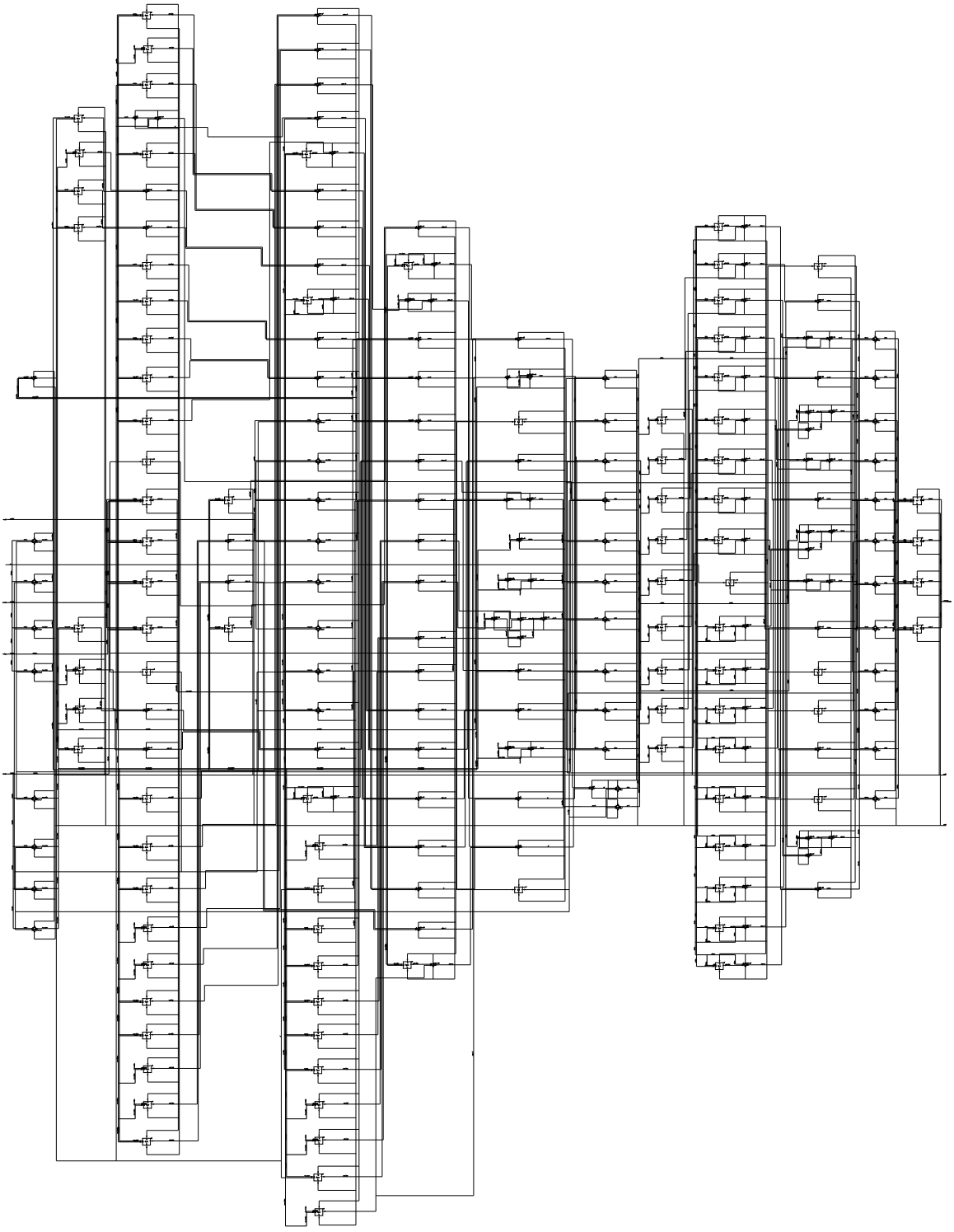


Figure 3.9: Post-layout netlist of the key schedule circuit.

$$G_1(x_2, y_2, z_2, w_2, x_3, y_3, z_3, w_3) = (g_{13}, g_{12}, g_{11}, g_{10}),$$

$$g_{13} = y_2 + z_2 + w_2, g_{12} = 1 + y_2 + z_2,$$

$$g_{11} = 1 + x_2 + z_2 + y_2w_2 + y_2w_3 + y_3w_2 + z_2w_2 + z_2w_3 + z_3w_2,$$

$$g_{10} = 1 + w_2 + x_2y_2 + x_2y_3 + x_3y_2 + x_2z_2 + x_2z_3 + x_3z_2 + y_2z_2 + y_2z_3 + y_3z_2;$$

$$G_2(x_1, y_1, z_1, w_1, x_3, y_3, z_3, w_3) = (g_{23}, g_{22}, g_{21}, g_{20}),$$

$$g_{23} = y_3 + z_3 + w_3, g_{22} = y_3 + z_3,$$

$$g_{21} = x_3 + z_3 + y_3w_3 + y_1w_3 + y_3w_1 + z_3w_3 + z_1w_3 + z_3w_1,$$

$$g_{20} = w_3 + x_3y_3 + x_1y_3 + x_3y_1 + x_3z_3 + x_1z_3 + x_3z_1 + y_3z_3 + y_1z_3 + y_3z_1;$$

$$G_3(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2) = (g_{33}, g_{32}, g_{31}, g_{30}),$$

$$g_{33} = y_1 + z_1 + w_1, g_{32} = y_1 + z_1,$$

$$g_{31} = x_1 + z_1 + y_1w_1 + y_1w_2 + y_2w_1 + z_1w_1 + z_1w_2 + z_2w_1,$$

$$g_{30} = w_1 + x_1y_1 + x_1y_2 + x_2y_1 + x_1z_1 + x_1z_2 + x_2z_1 + Y_1z_1 + y_1z_2 + y_2z_1;$$

$$F_1(x_2, y_2, z_2, w_2, x_3, y_3, z_3, w_3) = (f_{13}, f_{12}, f_{11}, f_{10}),$$

$$f_{13} = y_2 + z_2 + w_2 + x_2w_2 + x_2w_3 + x_3w_2, f_{12} = x_2 + z_2w_2 + z_2w_3 + z_3w_2,$$

$$f_{11} = y_2 + z_2 + x_2w_2 + x_2w_3 + x_3w_2, f_{10} = z_2 + y_2w_2 + y_2w_3 + y_3w_2;$$

$$F_2(x_1, y_1, z_1, w_1, x_3, y_3, z_3, w_3) = (f_{23}, f_{22}, f_{21}, f_{20}),$$

$$f_{23} = y_3 + z_3 + w_3 + x_3w_3 + x_1w_3 + x_3w_1, f_{22} = x_3 + z_3w_3 + z_1w_3 + z_3w_1,$$

$$f_{21} = y_3 + z_3 + x_3w_3 + x_1w_3 + x_3w_1, f_{20} = z_3 + y_3w_3 + y_1w_3 + y_3w_1;$$

$$F_3(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2) = (f_{33}, f_{32}, f_{31}, f_{30}),$$

$$f_{33} = y_1 + z_1 + w_1 + x_1w_1 + x_1w_2 + x_2w_1, f_{32} = x_1 + z_1w_1 + z_1w_2 + z_2w_1,$$

$$f_{31} = y_1 + z_1 + x_1w_1 + x_1w_2 + x_2w_1, f_{30} = z_1 + y_1w_1 + y_1w_2 + y_2w_1.$$

The four binary values in a 4-bit input bus to a F-box or G-box are represented by x, y, z, w. The subscripts of the input bits match the order of X and Y as is shown in Figure 3.8.

3.5 Conclusion

In this chapter we have introduced the target circuits we are using for our attacks. In the next chapter we are going to introduce the earlier researches we carry out to verify the property of static power leakage and our proposal of improved distinguishers for template attacks applied to static power analysis.

Chapter 4

Basic Verifications of Static Power Leakage

In previous chapters we have introduced the necessary background and the target circuits we are going to use in our work. In this chapter, some early verifications of static power leakage are demonstrated. Specifically, we try to analyze the static power leakage of a specific circuit structure, which is the bit-sliced structure. After this we show our attempts to improve the template attacks applied to static power by changing the distinguisher. The work in this chapter has been published in [39, 70].

Our proposed research uses simulations to explore CMOS technology sizes equal to and less than 90 nm. Although simulations have less noise than real-world measurements, we hope to get a clear understanding of how the behavior of sub-90nm circuits will be by starting our research from simulations.

We also keep in mind that our attack method should be suitable for a real-world scenario. It has been stated in [17] that static power analysis does not have significant advantage against dynamic power analysis in high frequency devices. Hence our target circuits run at the clock frequency of 5 MHz or less, which falls into the working frequency range of RFID tags. The number of gates in RFID tags is usually small, which means that lightweight ciphers or stream ciphers are preferred in these applications.

Here we describe the steps undertaken in our research. First we explore the characteristics of bit-sliced structures. After discovering that the correlation in a bit-sliced structure is not as simple as the linear relation described in [13], we decide to apply some generic models to our attack method. In this early period of research we choose the template attack, although other profiled attacks may also apply. Since template attack is the first profiled attack published, a static-power-based template attack is felt to be a good start point. Then based on this method, we try to attack several cryptographic circuits designed using 45nm technology. Transistor level simulation of a fully-functioning lightweight cipher takes a lot of time (1.3 hours is required to simulate one encryption with PRESENT cipher). Thus we decide to start from simple cryptographic circuits, then expand to larger circuits when we can verify the feasibility of our method. The attack targets include from simplest to the most complex: 1) a toy cipher which consists of only a 4-bit S-box and an XOR operation; 2) a serialized implementation of a 4-round SPN block cipher; 3) the reduced-round key schedule circuit for PRESENT cipher. The detailed functions of these circuits are explained in Chapter 3.

4.1 ASIC Design Environment

We simulate the target circuit in 90nm, 65nm, and 45nm environment. The 90nm and 65nm libraries we use are generic analog libraries from Taiwan Semiconductor Manufacturing Company (TSMC) and provided by CMC Microsystems. Although the access to the digital standard cells is provided by these libraries, we have not been able to use synthesis tools to generate gate level circuits from register-transfer level (RTL) source files. Hence, we build the gates using PMOS and NMOS transistors, and measure their properties using transistor-level simulation tools.

There are two libraries used in the 45nm environment: the analog/mixed signal (AMS) methodology kit provided by Cadence [71] and the open-source FreePDK kit provided by North Carolina State University [72]. We started our simulation in 45nm using the AMS kit, but found that ratio of the static power consumption versus the overall power consumption of the circuits built using this kit is lower than 0.1%, which is much lower than a realistic range. After switching to the FreePDK kit, we are able to acquire a percentage of around 30% for static power consumption, which is consistent with a real-world scenario, and is also consistent with the measurement results shown in [16, 17].

A limitation of the FreePDK kit is that it does not provide the capacitance table file for the cell components, which is mandatory if we want to use waveform files to control the input values to the circuit when approximating the percentage of static power consumption. If no specific waveform input is chosen, the design environment would use random values as the input values. Hence the plaintext and key inputs are randomly set, which may not be exactly the same as how they are set in a real-world application. As a result of this, the accuracy of static power percentage mentioned above is limited by the lacking of the capacitance table file. On the other hand, the NanGate cell library, which is based on the FreePDK kit, contains the capacitance table file. Both AMS kit and FreePDK kit allow access to digital standard cells, which means we are able to perform design automation such as synthesis and placement-and-routing (PNR) in our current design environment. Later on in our research we do not rely on the overall percentage of static power consumption, leaving the evaluations using capacitance table unnecessary to us. Hence, we use only the FreePDK kit when attacking the target circuits in later chapters.

Here we describe the design flow we use in our research. As is shown in Figure 4.1, first, we design each component of the target circuit using RTL-level Verilog. After

using testbenches to verify the correctness of our design, the Verilog files and the data files from the design kit library are used as the inputs for the synthesis step. The available synthesis tools include RTL compiler provided by Cadence and Design Compiler provided by Synopsis. This step generates the gate-level Verilog file of the target circuit and the design constraints (sdc file). Now we can use another testbench to verify the correctness of the generated gate-level Verilog file. Different from the RTL-level testbench, the gate-level testbench uses an extra input file which is the parameter file from the library. This verification step can also be used to generate the waveform file which can be used for gate-level power measurement. The verified gate-level Verilog, the constraints file, and the data file from the library are used as the input to the PNR step (also known as layout step). We use Cadence Encounter for automatic PNR. In order to ensure the universality of our PNR result, the layouts of all the target circuits are generated by automation script, and no manual modification is made. After the PNR step is successfully performed, we export the post-layout gate-level Verilog file and the delay file (sdf file). The post-layout Verilog file is the most accurate circuit architecture we can get in the simulation level.

Now we can perform three levels of simulated power measurement based on this design flow.

- We can use capacitance table and waveform file as the input in Encounter, and generate the overall consumption of internal power, switching power, and leakage power. Dynamic power is the summation of internal power and switching power, and static power is the leakage power. This is the least accurate method in our scenario, and is only used to estimate the percentage of static power.
- The real-time power consumption can be measured using gate-level power estimation tools such as PrimeTime and Encounter Power System. The inputs

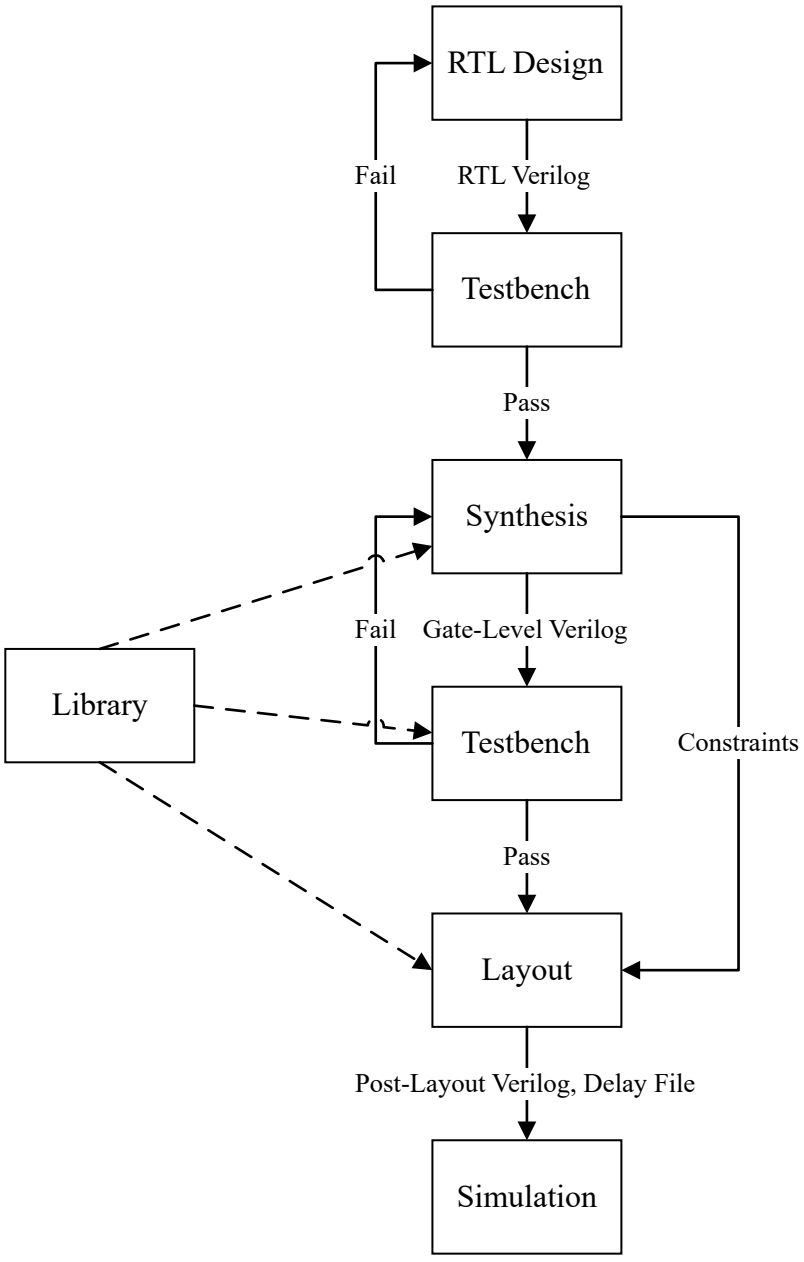


Figure 4.1: The process of designing our target circuits.

include post-layout Verilog file, constraints file, delay file, library data file, and waveform file. This method is fast but less accurate than the transistor-level simulation. These power estimation tools are not accurate enough to capture the difference between different static power stages. Hence they are not applicable in our scenario.

- The most accurate method is to import the post-layout Verilog file into transistor-level simulation environments such as Virtuoso. The imported Verilog is used as a netlist file, and it utilizes the standard cells provided by the library to create the whole circuit. The delay file is used during simulation to ensure better accuracy. This is the method we use for all our measurements.

4.2 Analysis of Bit-Sliced Structure

An analysis in [13] explains that the bit-sliced structure in a target device produces static power leakage that is linearly correlated to Hamming weight, which can be exploited to perform a successful attack. A bit-sliced structure refers to the structure for which a circuit has multiple identical blocks in parallel. Common examples of this structure include registers, ALUs, and buses.

In the early stage of our research, we try to analyze the behavior of bit-sliced structure in sub-90nm circuits and explore its feasibility in generic cryptographic circuits.

4.2.1 Verification of Bit-Sliced Model

The correlation between input and static current of a register is shown in [13]. Their result shows that the linear correlation exists in registers. However, the influence

of the difference between input/output and clock have been ignored. We start our research by verifying their simulations of registers using our libraries.

We use 90nm TSMC library for this simulation. Due to the lack of D flip-flop cells in the library, we implement the D flip-flop logic using the schematic design in Virtuoso environment. The structure we use in our D flip-flop is shown in Figure 4.2. This structure is commonly used in CMOS chips such as CD4013BC [73]. After implementing the D flip-flop, we combine 4 D flip-flops in parallel to implement a 4-bit register.

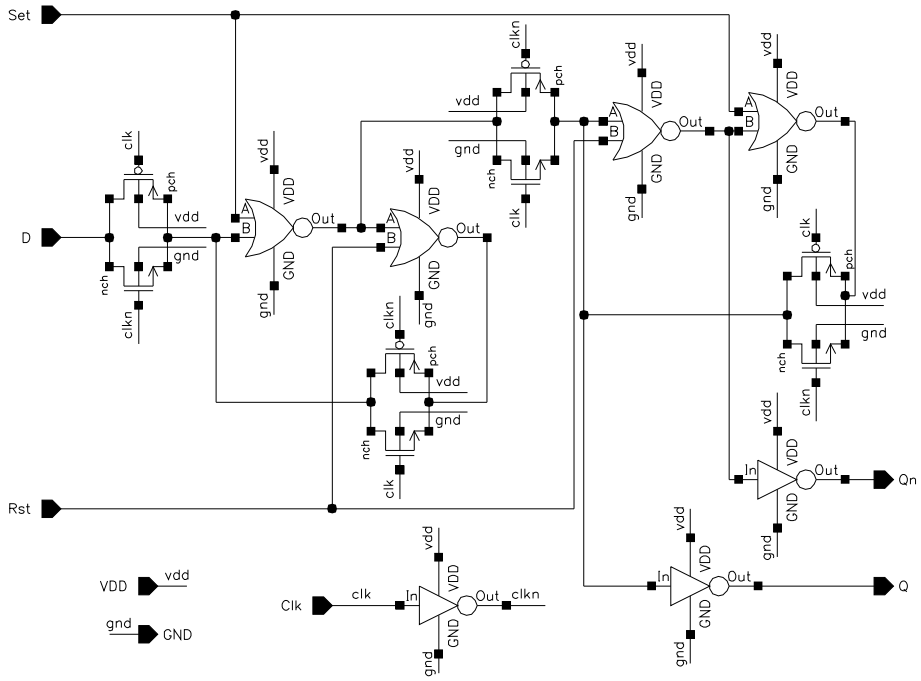


Figure 4.2: The CMOS structure of our D flip-flop implementation.

We simulate the register using mixed-signal simulation in Virtuoso environment. The input to the register is controlled by a Verilog functional module which generates ideal 0 and 1 outputs. We set an input to the register, then generate a rising edge in the clock (*clk*) input. Then we measure the current at the power input of the register. After the current settles to a steady value, we pick a time point in this clock period

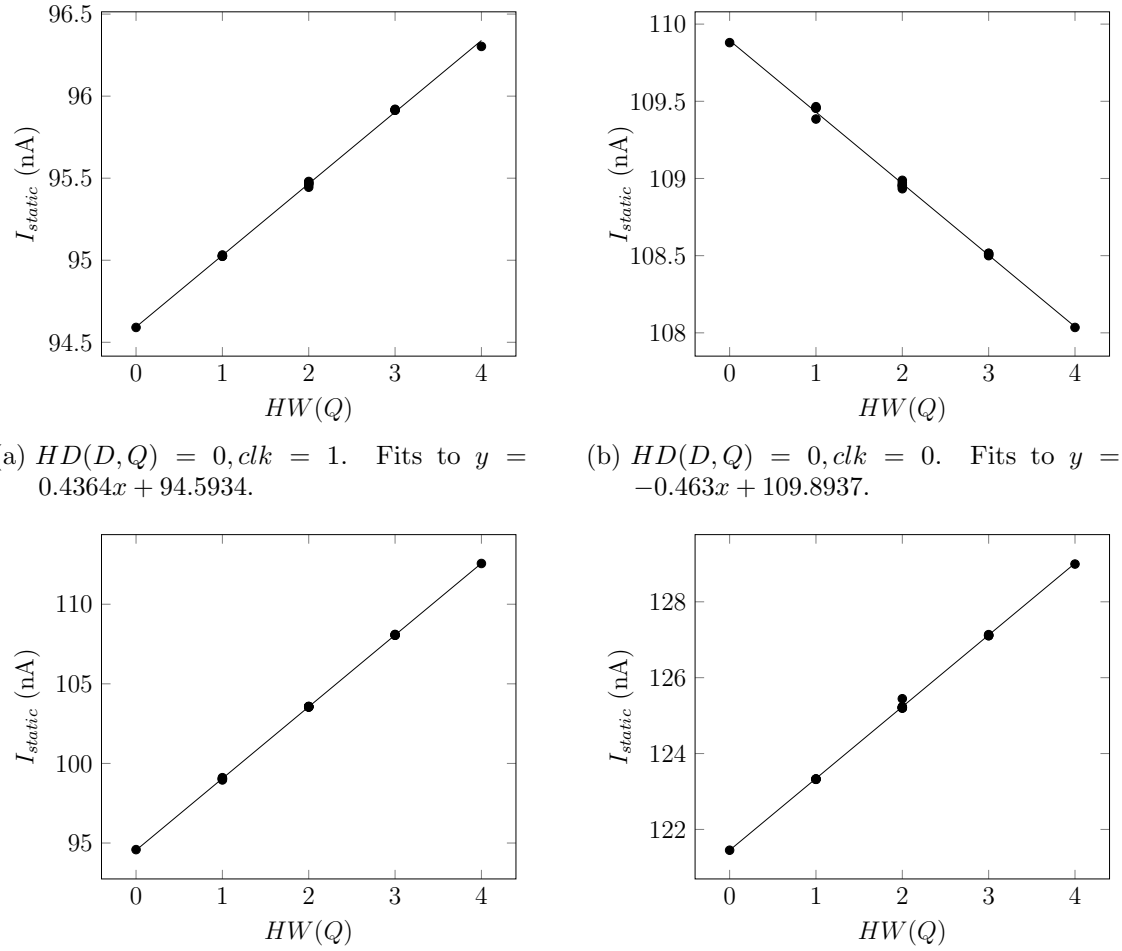
and measure the current value at this time point. After this, we change the clock input from 1 to 0, and measure the current after it settles again. These steps are repeated for different input values, and the time point is the same for different inputs.

Figure 4.3 shows our simulation results of the 4-bit register. D/Q refers to the input/output after the rising edge of clk. We use $HD(D, Q)$ to represent the Hamming distance between D and Q, and $HW(Q)$ represents the Hamming weight of Q. The X axis is the Hamming weight of Q; the Y axis is the measured static current leakage. We can see that the static current has very clear linear correlation with the output (in other words, the input before the rising edge of clock). This correlation is affected by the input value and clock. The case that $HD(D, Q) = 4$ and $clk = 1$ produces the largest difference in static current for different $HW(Q)$. In this case, the static current has strongest correlation with output.

We also repeat the simulation for 4-bit register in 45nm technology, and the similar linear correlation persists. It is clear that the inputs and outputs affect the static power leakage differently. Hence we conclude that, instead of the univariate relation stated in [13], the linear correlation model in registers is in fact multivariate.

In order to verify this, we model the simulation results of a 4-bit register in 45nm technology using multivariate linear regression. Instead of focusing on the cases that $HD(D, Q) = 0$ and $HD(D, Q) = 4$, we measure all the possible input and output combinations. Then we model the measurement results for $clk = 0$ and $clk = 1$ separately. The multivariate modeling parameters are: Hamming weight of the input, $HW(D)$; Hamming weight of the output, $HW(Q)$; Hamming distance between input and output, $HD(D, Q)$. The multivariate linear regression formula for this is

$$I_{static} = \theta_0 + \theta_1 \times HW(D) + \theta_2 \times HW(Q) + \theta_3 \times HD(D, Q). \tag{4.1}$$



(a) $HD(D,Q) = 0, clk = 1$. Fits to $y = 0.4364x + 94.5934$.

(b) $HD(D,Q) = 0, clk = 0$. Fits to $y = -0.463x + 109.8937$.

(c) $HD(D,Q) = 4, clk = 1$. Fits to $y = 4.5057x + 94.5467$.

(d) $HD(D,Q) = 4, clk = 0$. Fits to $y = 1.8908x + 121.4513$.

Figure 4.3: Simulation results of 4-bit register in 90nm technology.

After getting the linear regression results we use the root-mean-squared error (RMSE) and R^2 to evaluate the modeling results. Let there be n measured current values, I_i be the i-th measured current, $\hat{I}_i$ be the predicted current at this measure point, and $\overline{I}$ be the average current. $RMSE$ and R^2 are specifically defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{I}_i - I_i)^2}{n}}, \tag{4.2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{I}_i - I_i)^2}{\sum_{i=1}^{n}(I_i - \bar{I})}. \tag{4.3}$$

A model is good if $RMSE$ is close to 0 and R^2 is close to 1.

For $clk = 0$, we get

$$I_{static} = -1.519 + 0.050857 \times HW(D) + 0.05035 \times HW(Q) - 0.047143 \times HD(D,Q)(\mu A)$$

with $RMSE = 3.60 \times 10^{-6}$ and $R^2 = 1$. For $clk = 1$, we get

$$I_{static} = -1.7732 + 0.024981 \times HW(D) + 0.070557 \times HW(Q) - 0.046998 \times HD(D,Q)(\mu A)$$

with $RMSE = 3.67 \times 10^{-6}$ and $R^2 = 1$. To conclude, the multivariate linear model fits well for the static power leakage of 4-bit register.

4.2.2 Generalization of the Multivariate Linear Model

It is common in practice that registers are added to inputs and outputs of some modules, e.g. S-boxes. In such cases the adversary can use the S-box inputs and outputs as intermediate states in power analysis. However, it is also possible that in some highly compact designs, the register number is reduced. In these cases the registers may not be the dominant source of static power leakage. Hence we try to analyze other core components of cryptographic hardware.

There are possibilities that S-boxes can be designed as a bit-sliced structure. The Decoder-Switch-Encoder (DSE) structure [74] is known to be an energy-efficient structure [75]. We implement the 4-to-16 decoder and 16-to-4 encoder using the 90nm TSMC library in Virtuoso environment. The structures of the decoder and encoder are shown in Figure 4.4 and Figure 4.5, which satisfy the bit-sliced model. We carry out

simulations separately on the decoder and encoder using the methodology described in Section 4.2.1.

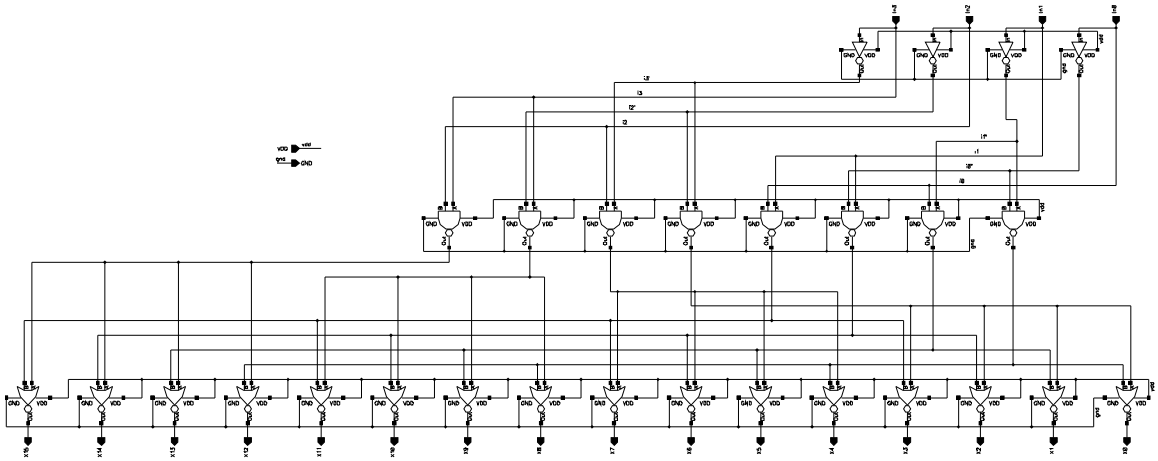


Figure 4.4: The structure of a 4-to-16 decoder.

The simulation results in Figure 4.6 show that the static current of the decoder/encoder is linearly correlated to its input/output. Thus we treat the DSE-structured the S-box as a combination of two bit-sliced components, and I_{static} of S-box should be approximately the summation of I_{static} for the decoder and the encoder. Using $HW(input)$, $HW(output)$, and $HD(input, output)$ as the multivariate linear regression parameters, the fitting result is

$$\begin{aligned} I_{static} =& 244.68 - 4.3153HW(input) - 8.5197HW(output) \\ &+ 0.28008HD(input, output)(nA) \end{aligned}$$

with $RMSE = 1.0066$ and $R^2 = 0.9910$. Here we use the p value, which is defined as the probability of obtaining a modeling result equal to or better than the current one when the selected parameter is removed [76], to evaluate the significance level of one parameter in the modeling result. In this case, we find that the corresponding p value for $HW(input)$, $HW(output)$, and $HD(input, output)$ are 9.749×10^{-10}, 1.117×10^{-12},

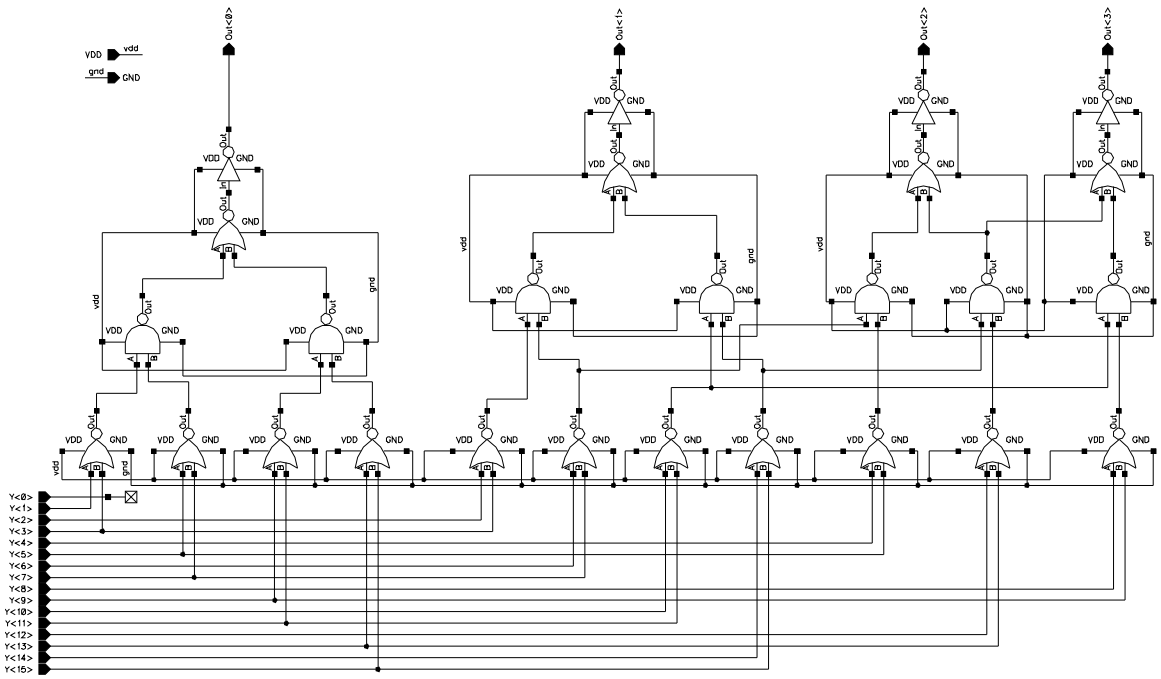


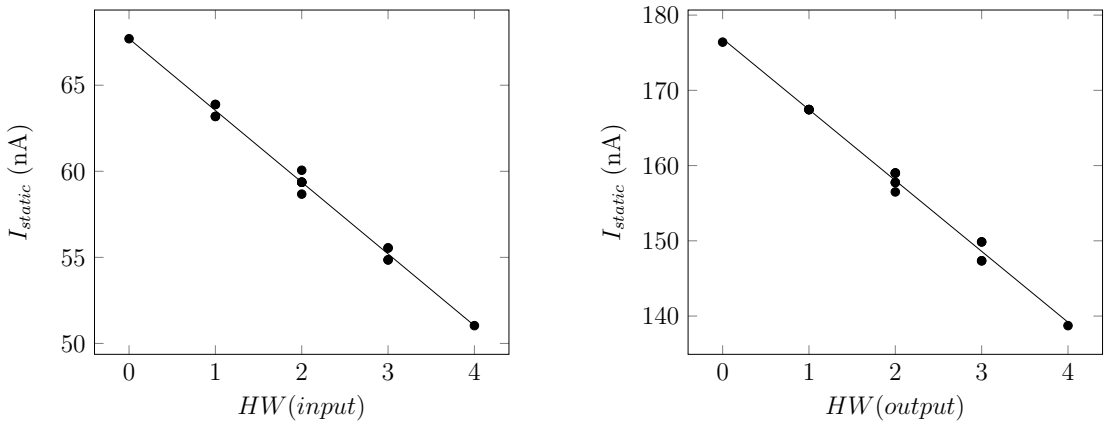
Figure 4.5: The structure of a 16-to-4 encoder.

and 0.4291. The p value for $HD(input, output)$ is greater than 0.05, which means that the contribution of this parameter is not significant in the fitting result. After removing $HD(input, output)$, we get a fitting result as

$$I_{static} = 244.68 - 4.2931HW(input) - 8.6219HW(output)(nA)$$

with $RMSE = 0.9937$ and $R^2 = 0.9905$. Thus our result shows strong linear correlation between I_{static} and the Hamming weight of input/output.

However, the DSE structure is only one specific structure. The linear relation may not exist in another structure. Even worse, circuit design tools usually provide optimization functionalities. There is no guarantee that an S-box designed using DSE structure in RTL level remains perfectly bit-sliced in the gate level after optimization. We redesign the DSE-structured S-box using minimization techniques, then model its



(a) Decoder fitting result. Fits to $y = -4.1667x + 67.7008$.

(b) Encoder fitting result. Fits to $y = -9.4201x + 176.8621$.

Figure 4.6: Simulation results of 4x16 decoder and 16x4 encoder in 90nm technology.

correlation using multivariate linear regression. The result is

$$I_{static} = 200.73 - 4.0348HW(input) - 5.1826HW(output)(nA)$$

with $RMSE = 2.82$ and $R^2 = 0.854$, which is worse than the previous DSE design. It is shown in [13] that due to the nonlinear transformation in the S-box, there is weak linear correlation in their simulation result. To verify their claim, we also use multivariate linear regression to model their data, and get

$$I_{static} = 118.96 + 3.255HW(input) - 1.08HW(output)(nA)$$

with $RMSE = 4.18$ and $R^2 = 0.488$.

We conclude that the bit-sliced structure is not a generic model for static power analysis. The bit-sliced structure and Hamming weight model does work for some circuit components, but if the whole circuit contains large scale of non-parallel gate logic, then it could contain noise which will affect the measurement of the leakage from

bit-sliced components. If we know that the bit-sliced structure exists in a given circuit, it is still non-trivial to apply this model, since we do not know exactly which parameter dominates the multivariate linear correlation model: different libraries would result in different leakage measurements. The structure also matters, since that contribution of Hamming distance may or may not be significant in different structures. Furthermore, the level of linearity in the bit-sliced model relies on specific structures which may easily be modified by design tools. With all these factors considered, the attacker would need detailed knowledge of the circuit structure in order to perform a successful static power analysis using the bit-sliced structure, and this may not be practical in many cases.

4.3 Template Attacks Using Static Power Leakage

As was mentioned in the previous section, the bit-sliced structure can be easily broken by adding nonlinear functions. In order to develop a method that can be applied to more complex circuits, we need a more generic statistical model. In other words, we need a robust model which can automatically distinguish different types of structures including the bit-sliced one.

It is commonly known that the same standard cell in the same design library would have the same electronic parameters, which means that if two circuits are designed using exactly the same structure and cells, then they would have the same static power leakage. To model this, say a circuit component performs an operation $\mathcal{O}(\cdot)$ which generates some kind of static power consumption, the behavior of $\mathcal{O}(\cdot)$ is determined by the library configuration *lib* and the circuit structure *struct*. For a limited set of

inputs $\mathbb{I}$ and outputs $\mathbb{O}$, a pair of input $in \in \mathbb{I}$ and output $out \in \mathbb{O}$ in this circuit can be mapped to a fixed static power leakage measurement result l. The set of leakage measurement results $\mathbb{L}$ is limited if the noise ε is ignored. In a real-world measurement, ε comes from different sources such as power consumption of other surrounding circuits, measurement noise caused by the probe, and manufacturing defects. In a simulation measurement, we can control the simulation environment to ensure that $\varepsilon = 0$. If we regard $\mathcal{O}(\cdot)$ as a function, then the result of this function is $l + \varepsilon$. This mapping relation can be represented as

$$\mathcal{O}_{lib,struct}(in, out) = l + \varepsilon, \tag{4.4}$$

where $in \in \mathbb{I}, out \in \mathbb{O}, l \in \mathbb{L}, \varepsilon \overset{R}{\mapsto} \mathbb{R}$.

In bit-sliced structure, $\mathcal{O}_{lib,struct}(\cdot)$ is a linear relation related to $HW(in)$, $HW(out)$, and $HD(in, out)$, while in other structures this mapping may result in some unknown relation which could be a mixture of linear and non-linear relations. Though the detail of $\mathcal{O}(\cdot)$ is unknown, the mapping relation is always fixed for the same *lib* and *struct* parameter. For the measurement of cryptographic devices, the input in can be seen as a tuple of plaintext $p \in \mathbb{P}$ and key $k \in \mathbb{K}$, that is $in = (p, k)$. The output is the encryption result of p and k, that is $out = enc(p, k)$. Hence $\mathcal{O}_{lib,struct}(\cdot)$ can be represented as

$$\mathcal{O}_{lib,struct}((p, k), enc(p, k)) = l + \varepsilon \tag{4.5}$$

For a given black-box target circuit, k is fixed for all the encryptions, and p is randomly chosen from $\mathbb{P}$, that is, p satisfies uniform distribution. $\mathcal{O}(\cdot)$ can be seen as a mapping from a uniform distribution of p to some unknown distribution of l.

This mapping is controlled by parameters *lib*, *struct*, and k, among which k is the attacker's target. If another identical device is available to the attacker in which only k is chosen differently, then this relation results in a fixed distribution of l which is only related to k. If the attacker can distinguish different distributions of l in regards to different k parameters, then the attacker can recover the key.

This encourages us to apply template attacks to static power leakage, which uses multivariate Gaussian distributions to model the distribution of measured template traces. The templates can be built using different statistical tools to describe different types of distributions, which means this method can model different types of $\mathcal{O}_{lib,struct}(\cdot)$ functions besides linear correlation.

A common issue with template attacks on dynamic power is that it needs to compute high-dimensional matrices due to the large number of measurement points. Template attacks on static power leakage uses only one point-of-interest (POI) for each input/output state (which can be 2 POIs for each clock). The attacker can further shrink the size of POIs by choosing specific points. Figure 4.7 is an example of the current trace after the clock input changes. There is a rising edge in the clock input at time 40 ns. The figure shows significant sign of oscillation right after the clock change, which reflects the contribution of dynamic power leakage. After this, the power trace approaches a fixed value, which reflects the static power leakage. Note that the power trace does not go to a fixed value immediately, but instead it slowly decays to this value. This behavior is caused by the internal power, which is caused by the charging of internal loads and short-circuit current between transistors. The contribution of internal power becomes relatively small after all the inputs/outputs are fixed for a period of time. Note that this figure is only an illustration of the current behavior. In actual measurements we find that the spikes caused by dynamic power are rather high, and after the dynamic period the current trace decays much faster than shown

in this figure. For the accurate measurement of a simulated trace, we need to measure the point after the current trace settles.

For a trace measured on an actual chip, there is usually a large quantity of noise. In such real-world measurements, averaging is required to get only one point that can represent the static power leakage for the current period. This technique takes multiple points from the same period in the same trace, and averages them to the one point value.

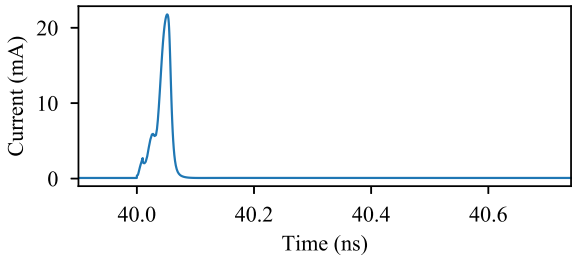


Figure 4.7: Part of a power leakage trace captured in simulation, a clock change happens at 40 ns.

4.4 Attacks on Different Cryptosystems

In this section we show the experiment results using static-power-based template attacks. We performed all our simulations using the 45nm environment described in Section 4.1. The PRESENT S-box circuit and the serialized SPN circuit are simulated using the AMS kit library, and the key schedule circuit of PRESENT is simulated using the FreePDK library.

4.4.1 Attack on the S-Box Circuit

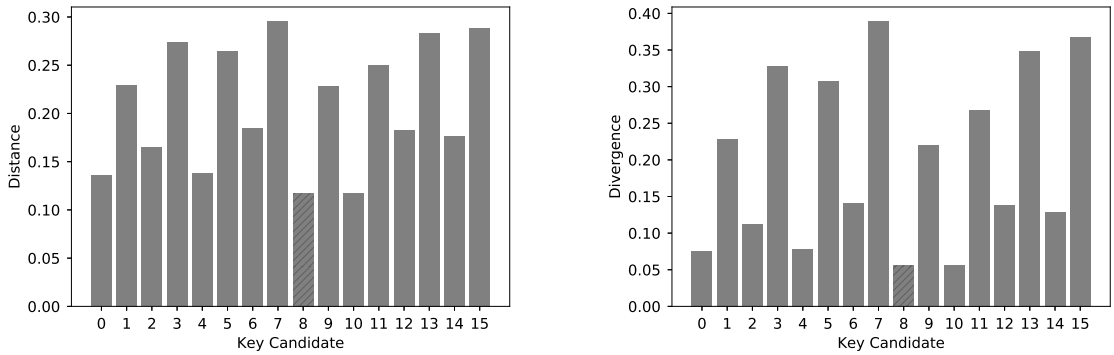
In this section we show the performance of static-power-based template attack on the S-box circuit which is introduced in Section 3.1.

In this cryptosystem three 4-bit registers are used to store the plaintext, key, and ciphertext. The plaintext is the input to an S-box component, which implements the PRESENT S-box [33] using the DSE structure we implemented in Section 4.2.2. The output of the S-box is XORed with the key to generate the ciphertext.

In this circuit, the encryption is done in 1 clock period, so 2 POIs (one for the clock high and the other for the clock low) are enough for static power analysis. According to our results in Section 4.2.1, the inputs and outputs of the register both affect the static power leakage. Since the input to the ciphertext register is deterministic on the output of the plaintext and key registers, the input to the ciphertext register is fixed given the output from the plaintext register when the output from the key register remains the same for all plaintext possibilities. Given a fixed value for the 4-bit input and output of the key register, there are $2^4 = 16$ possibilities for each of 1) plaintext register input, 2) plaintext output and ciphertext input, and 3) ciphertext output. Hence there are in total $16^3 = 4096$ possibilities of static power leakage ignoring the noise.

In order to perform a template attack on this circuit, we profile the templates for all 16 key guesses. For each key guess, we profile using a fixed sequence of different plaintext inputs. After this, we randomly pick two attack targets: key 1000 and key 0110. Setting these as the key input, we use the circuit to encrypt the same sequence of plaintext inputs for each target key. In other words, chosen plaintext attacks are performed in this scenario, in which the plaintext input set for profiling and attack phases are chosen to be the same.

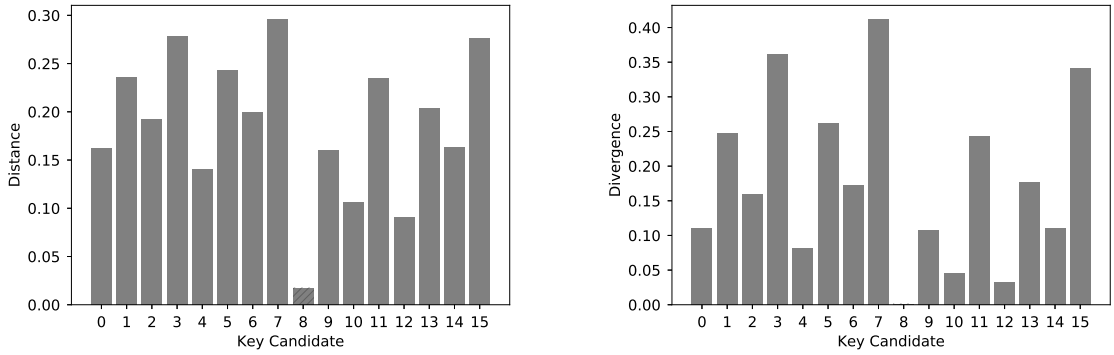
Then we perform template attacks by replacing the original distinguishers with the ones we introduced in Section 2.5.3. In this attack we use an identical circuit for profiling and attacking. Using 100 traces for both profiling and attack phases, the results of an attack on key 1000 are presented on Figure 4.8. It can be seen that the



(a) JS distance (b) J-divergence

Figure 4.8: Template attack results on key 1000 using 100 traces for template and 100 traces for attack.

key guess 1000 (8 in decimal) results in smallest distance/divergence values, thus the correct key is successfully detected, although that its distance/divergence value is only slightly smaller than that of the wrong key 1010 (10 in decimal). If we extend the number of template and target traces to 2000, this difference is more significant, as is shown in Figure 4.9. We observe that when using J-divergence, even though the size of the template sets and target set has been increased, the difference between correct key guess and some wrong key guesses (e.g. 1100) is still not very significant. This attack is also successful on target key 0110.



(a) JS distance (b) J-divergence

Figure 4.9: Template attack results on key 1000 using 2000 traces for template and 2000 traces for attack.

We conclude that the static power analysis is successful on this simple cryptosystem using template attacks with JS distance and J-divergence as the distinguishers. Our result shows that in this attack scenario JS distance is slightly better than J-divergence, for it gives more significant difference between the correct key and the wrong keys given the same template sets and attack set. This is a preliminary work that we did in the early period of our research.

4.4.2 Attack on a Serialized SPN Cipher

In this section, we attack a serialized SPN cipher which has four 4-bit blocks. Each block is encrypted using the same key input. The design is a 4-bit iterative circuit, which performs operations on 4 bits at a time. The design is done in RTL level and we perform our simulation on the circuit after synthesis and layout using the 45nm FreePDK library.

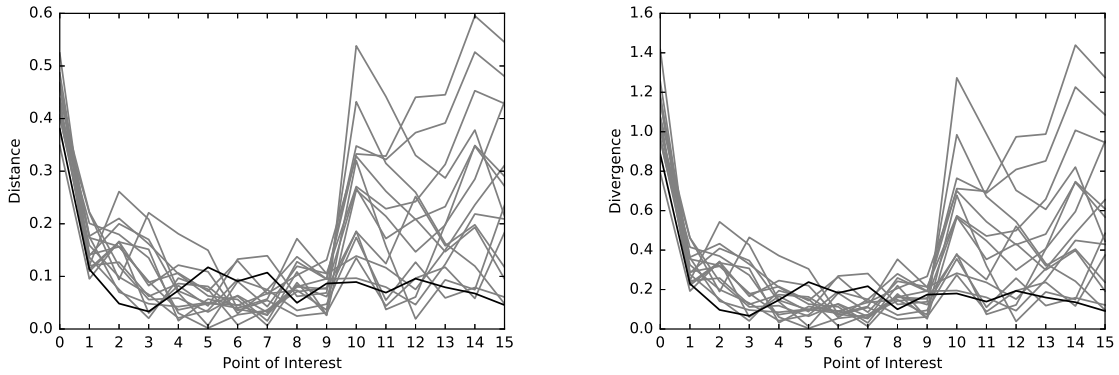
The design structure of this circuit is shown in Figure 3.4. This toy cipher accepts 4-bit plaintext input in each clock and XORs it with the fixed 4-bit key stored in the 4-bit key register. The result is processed by the S-box operation and the output is shifted into a 16-bit shift register. This register shifts 4 bits in each clock. After 4 clocks of loading the shift register stores the 16-bit intermediate value which is used for permutation. Multiplexers are used to select the bits from the shift register according to the permutation network shown in Figure 3.5 and output 4 bits of ciphertext in each clock period. We use the PRESENT S-box for the encryption. In this circuit we use an automatically generated S-box, hence the effect of bit-sliced structure is weakened. We also omit the register for the plaintext input to further weaken the effect of the bit-sliced structure. Similar serialized structures are commonly used in circuit design, especially in lightweight block ciphers, in order to shrink circuit size.

Registers are needed in this structure to store the intermediate values, such as in the implementation proposed in [68]. In order to mimic the usage of this register, we use a 16-bit register to store the encryption results, and output 4 bits in each clock period. Since we can gain enough randomness from the plaintext inputs, pipelining is not used in this circuit.

We use a similar attack strategy as we used in Section 4.4.1. However we pose a stronger assumption on the attack scenario that the attacker has no knowledge of the plaintext. Hence we collect the profile and target traces using different randomly generated plaintext inputs. This differs from the previous attack that the attacker is unable to choose a set of known plaintext inputs. In other words, the attacker has no knowledge of the plaintext. We have achieved success in attacks against all 16 possible key guesses. Here we are going to use key 1110 (14) as an example.

In this circuit, 4 clock periods are used for encryption and 4 clock periods are used for output. Hence there are at most 16 POIs according to our measurement method. We use univariate J-divergence and JS distance to find the points where the largest differences show, in order to get a better attack result. Using only 50 traces for the template sets and 50 for the attack set, we are able to achieve the univariate analysis result for all 16 keys shown in Figure 4.10. The black line refers to the univariate trace generated using the correct key.

Here we observe that large distances exist in POIs 0 and 9 to 15. The behavior of POI 0 is largely due to the randomness of the plaintext input, hence the encryption logic does not have large influence on the distance in this POI. The fact that all the traces in Figure 4.10 have relatively high distance/divergence value at POI 0 confirms this. The reason that the traces with wrong key guesses have large distances in POIs 9 to 15 is that the circuit is outputting the values in the 16-bit register, which stores the ciphertext generated using a given key guess. The distributions of static power



(a) Univariate JS distance

(b) Univariate J-divergence

Figure 4.10: Univariate analysis results of the serial block cipher using 50 traces for the templates and 50 traces for the attack set.

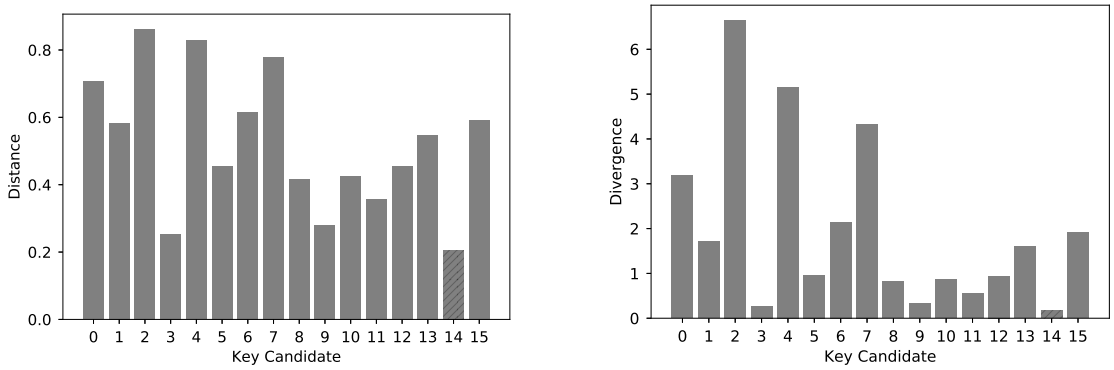
consumption at these POIs are related to the key guesses, while in other POIs the register contains random bits which have not been replaced by the ciphertext. Hence we pick POIs 9 to 15 for our attack.

Figure 4.11 shows one example of our attack results. The target key in this case is 1110 (14 in decimal). The distance/divergence between the template using the correct key guess and the attack trace is shown as shaded bars. As it is shown in the figure, the bars for 14 show least distance/divergence values of all, which means that the difference between the trace distribution of this key guess and the correct key is minimal of all key guesses. Thus the correct key has been successfully distinguished as 1110.

Similar to the attack in Section 4.4.1, this is also a preliminary work that we did in the early period of our research.

4.4.3 Attack on the Key Schedule Circuit of PRESENT

In order to provide flexibility in implementation, the key schedule is designed as a function that can be computed separately in some block ciphers such as AES and



(a) JS distance (b) J-divergence

Figure 4.11: Template attack results of the serial block cipher using 50 traces for the templates and 50 traces for the attack set.

PRESENT. However, if not protected properly, the key schedule circuit may introduce vulnerability to the cryptographic circuit.

In this section we perform static power analysis against the simplified key schedule circuit of the proposal in [68]. This circuit is designed in RTL level, and synthesis and layout of this circuit are performed automatically using 45nm FreePDK library.

We attack on the first key nibble loaded in the key register. This 4-bit key input is stored in the key register in clock periods from 0 to 5. After this, it is XORed with the round counter and then processed by the S-box operation.

Due to the bit-sliced structure of the registers, the power consumption in clock period 0 should be related to the Hamming weight of the target key, with some noise from the plaintext register; the power consumption in clock periods 1 and 2 contains noise from other key inputs; the power consumption in clock periods 3 to 5 would be noisier with more key inputs, but the non-linear function $f(key)$, which implements the S-box and XORing with round counters, should not worsen the attack result. Since $f(key)$ is affected only by the key values, as we described in (4.4), the static power leakage of the non-linear function is $\mathcal{O}_{lib,struct}(key, f(key)) = l + \epsilon$. Hence the

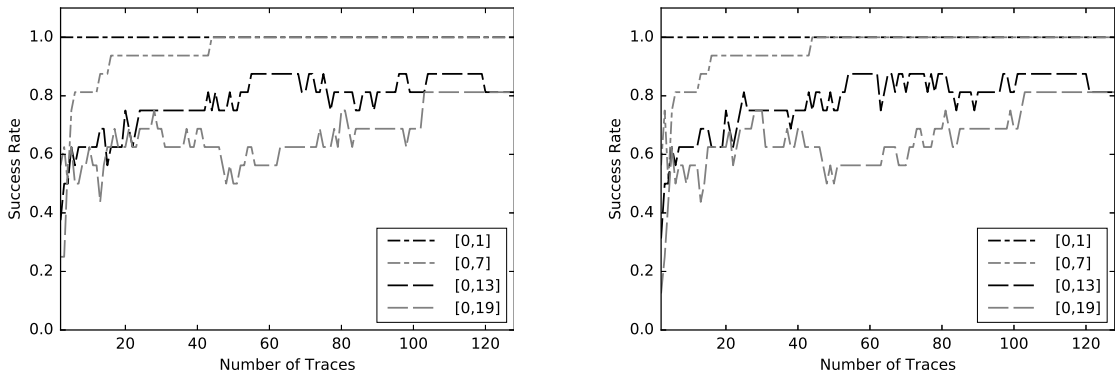
power consumption of the non-linear function is directly related to the key register, and no further noise is added. To conclude, the first clock period contains the least quantity of noise, the portion of noise in the power consumption increases with more POIs (recall that each clock period contains 2 POIs) from other clock periods added.

There are 16 possibilities for a 4-bit key nibble. We generate 16 sets of attack traces using all the 16 possible keys as the correct key, key inputs when the circuit is not in loading state and the plaintext input are set as random. Then we generate the template sets using 16 key possibilities and other random inputs. We perform our attack using different sets of POIs. For each set of POIs, we attack the 16 correct keys separately using the attack sets. Then we use the success rate introduced in Section 2.7.4 to evaluate the result of our attacks.

Here we are attacking the Hamming weight of the key. We find that in this circuit there is a large amount of registers. Many of these come from the shift register used to store the key bits, which does not have much relation with non-linear logic, but does consume a large amount of internal power which is measured in our traces. Hence, in this circuit the bit-sliced structure is dominant. As a result we are not able to directly recover the actual correct key, but we are able to distinguish the Hamming weight of the correct key.

Figure 4.12 shows the result of our attacks. The number of traces for both the attack set and the template sets go from 2 to 128. When only the first clock period (POIs 0 to 1) is used, we are able to successfully attack all the keys even using the minimal number of traces. More noise is introduced when more POIs are added, hence the success rate trace goes below 1.0.

This result shows that the static power analysis is successful against the loading period of PRESENT key schedule circuit. By choosing the POIs wisely, we can get a very satisfactory attack result. The relation in the loading period is straightforward



(a) Success rate using JS distance

(b) Success rate using J-divergence

Figure 4.12: Success rate of template attacks against the PRESENT key schedule circuit using different POI sets, e.g., [0,7] refers to POIs from 0 to 7.

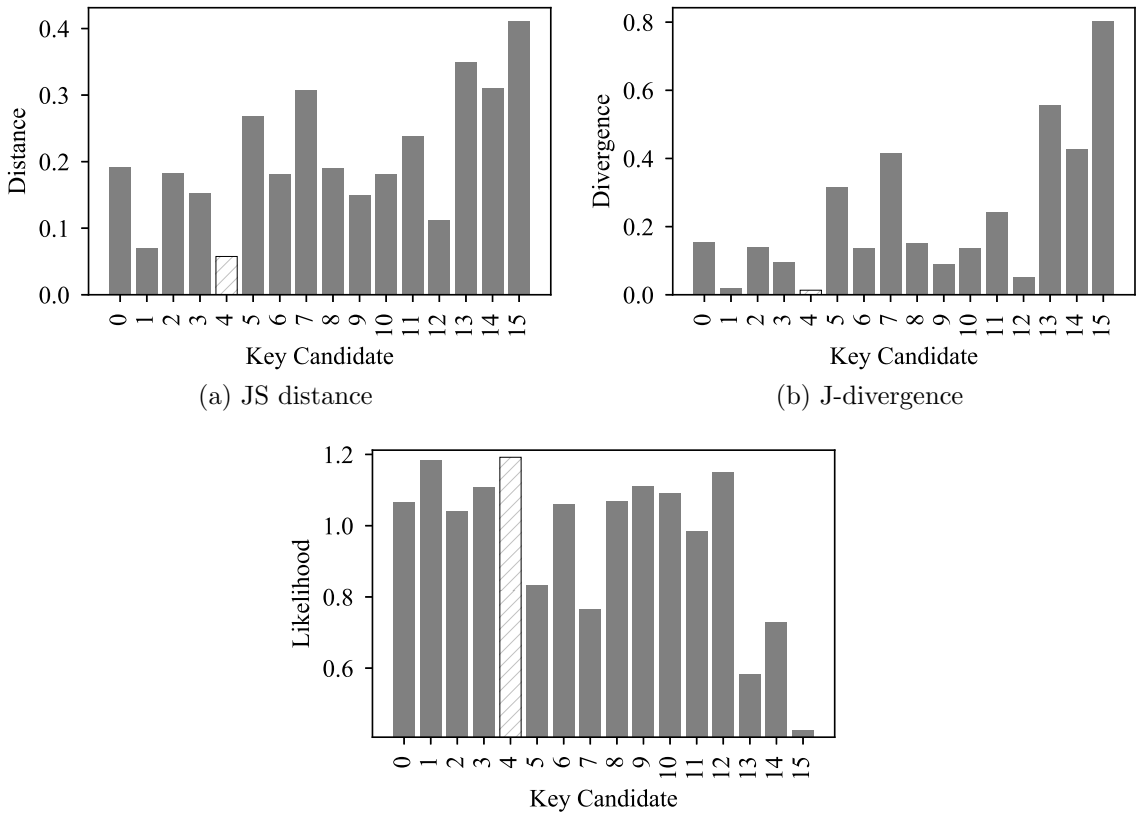
since there is no computation involving the target key. With more clock periods added, the input from other key registers and the plaintext resigters add noise to the measured traces, and the attack result is worsened.

4.4.4 The Effect of Different Distinguishers

In this section, we discuss a brief comparison of different distinguishers used in a template attack. We proposed to use JS distance and J-divergence to optimize the original Gaussian distinguisher in [39]. The PRESENT S-box and the key schedule circuit are used for verification.

In the PRESENT S-box circuit, the encryption is done in 1 clock period, so 2 POIs (one for the clock high and the other for the clock low) are enough for static power analysis. In order to perform a template attack on this circuit, we profile 16 template sets using 16 different key guesses. For each key guess, we profile using different sets of 50 random plaintext inputs. Next, assuming that we attack on key 0100 (4 in decimal), we use this key to encrypt another set of 50 random plaintext inputs. Hence, the attack is performed without knowledge of the plaintext. The result of this attack

is shown in Figure 4.13. All of the three distinguishers gain successful results as key 4 has the lowest JS distance and J-divergence, as well as highest Gaussian likelihood.



(a) JS distance

(b) J-divergence

(c) Gaussian likelihood

Figure 4.13: Template attack results on key 0100 using 50 traces for template and 50 traces for attack.

We also perform this attack using 1000 traces for the template and attack sets to verify the computation time of the distinguishers. Using Python with the Intel Math Kernel library on a computer with i7-4790 CPU, we are able to achieve our results in an elapsed time of around 0.023 s for J-divergence, 0.052 s for JS distance, and 0.599 s for Gaussian likelihood.

Next, we perform our attack on the simplified key schedule circuit of the PRESENT cipher. The structure of this circuit is introduced in Section 3.3. We attack on the first

key nibble loaded in the key register. A 4-bit state register is used to store plaintext inputs, and this component adds noise to the power consumption of the key schedule circuit. We perform our attack following a similar attack process that we apply to the toy cipher, except that in this attack we try to recover the Hamming weight of the correct key. Registers are the major source for power consumption in this circuit, and as is shown in [13], the static power consumption of registers follow the Hamming weight model, thereby making it difficult to distinguish between keys of the same Hamming weight.

We perform 16 attacks using a different correct key nibble (from 0000 to 1111) for each attack. We use the same number of traces used for profiling and attacking, and this number ranged from 2 to 128. Then we use the success rate defined in [66] to evaluate the result of our attacks. The result of this attack is shown in Figure 4.14. The success rates using different distinguishers increase as the number of traces increases, and approaches 100% with less than 60 traces. As is shown the new distinguishers gain the same level of performance as the Gaussian distinguisher. Using 128 traces for profiling and 128 traces for attack, the time for a single attack is around 0.019 s for J-divergence, 0.030 s for JS distance, and 0.051 s for Gaussian likelihood.

In this work we conclude that using the proposed JS distance and J-divergence we are able to achieve the same level of performance as the original distinguisher using less computing time.

4.5 Conclusion

In the earlier stage of our research we tried to model the behavior of static power leakage by measuring the power consumption of some components in a specific circuit structure. This provides us with some basic understanding of the factors that affect

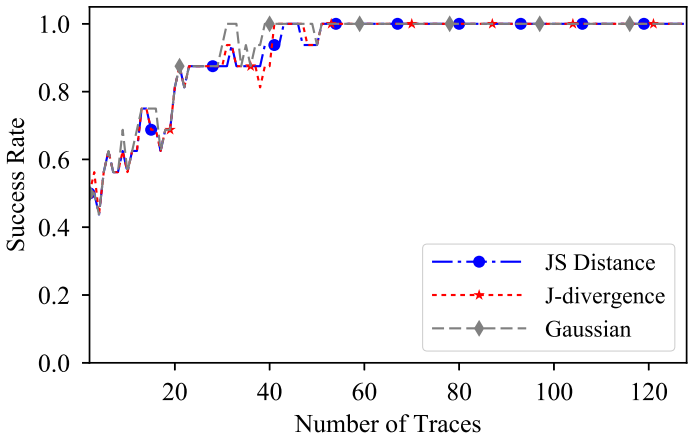


Figure 4.14: Success rate of template attacks on the key schedule circuit. The success rate is the number of successful attacks in all the 16 attacks.

the static power leakage. However, later we decided to use a more generic statistical method to model the behavior of static power. As a result our research target became to profile attacks and particularly template attacks.

We have gained successful results in the static-power-based template attacks against the S-box circuit, the serialized SPN cipher and the key schedule circuit of PRESENT. As we have observed in the attack against the key schedule circuit, the attack becomes less successful when there is more noise from other components of the circuit.

Note that we use random plaintext inputs in the attack against the serialized SPN cipher and the key schedule circuit. This assumption indicates that the attacker has no knowledge of the plaintext inputs, which could happen in scenarios where the attacker is targeting part of a cryptographic circuit, and the input is some unknown intermediate value. As is described in [77], as long as the attacker has access to the input to the target device, it is reasonable to assume that the attacker may have control to the plaintext inputs.

In the next chapter, we shall introduce our work of applying kernel-based template attacks to static power, which applies to more generic non-Gaussian trace

distributions.

Chapter 5

Kernel-Based Template Attacks Using Static Power

In this chapter we propose non-parametric template attacks which use a kernel methodology to improve the accuracy of modeling static power consumption. The proposed template attacks are tested using transistor-level simulations of circuits designed with a 45-nm standard cell library. Our test results show that our approach improves the success rate of template attacks using static power in cases where the distribution of static power consumption cannot be accurately modeled by Gaussian models. The work in this chapter has been presented in [78].

5.1 Simulation Workflow

We use simulated CMOS circuits to perform template attacks in our research. All the circuits are designed and simulated using the FreePDK 45-nm library [72].

The circuits we use to verify our proposal are designed using a formal design workflow of digital circuits. We firstly design the register-transfer level (RTL) logic of a target device, and then the RTL logic is synthesized using Synopsys Design Compiler with standard cells provided by the FreePDK library to generate a pre-layout netlist. Next the pre-layout netlist is imported to Cadence Encounter to perform placement and routing and in this step we generate the post-layout netlist and a standard delay

file (sdf). Finally, the post-layout netlist and sdf file are imported to Cadence Virtuoso to perform spice-level simulations.

The inputs to the target devices are idealized digital signals generated by VerilogAMS modules. The working voltage for the targets is set as 1.1 V. In order to ensure maximum generality of our targets, we build our circuits using the default parameters for the standard cells and CAD tools. The frequency constraint for placement and routing is set as 100 MHz, which is a commonly used clock frequency in microcontrollers. The simulation is carried out using a clock frequency of 5 MHz.

Recall that it is demonstrated in Figure 4.7 that the trace can be separated into two parts. In our research we are interested in exploiting the static power, hence we take measurements in the static period. To minimize the effect of capacitive charging, we take measurements at the same time point for each clock period. That is, we wait for a fixed amount of time after the clock changes to measure the current leakage. For example, when the clock is set to change every 100 ns, we measure the current value when it is 95 ns after the clock edges.

5.2 Proposed Attacks

In this section, we describe the kernel-based template attacks that apply to static power, which is based on the original template attack algorithm described in Section 2.5.1.

5.2.1 Approach

Recall that there are two phases in template attacks: a profiling phase and an attack phase. During the profiling phase, the profiling device is used to build templates corresponding to each possible key before the actual attack is performed. Each

template represents the statistical properties of a set of side-channel measurement results, and in our case it refers to the static power traces. Following the profiling phase, the target device is attacked by obtaining a set of traces generated with a fixed unknown key. The attacker compares the target traces with all the templates and tries to find a template that is statistically closest to the target traces, and the key of this template is presumed to be the key of the target device.

Here we give a detailed description of our proposal to perform the kernel-based template attacks using static power:

1. Assuming that the unknown key in the target device has K possibilities, random plaintext input is used to generate M_p power traces for each possible key using the profiling device. Hence there are K sets of traces, each having M_p traces of encryption using random plaintexts and a corresponding known key value. Let there be N measurement points in each trace. For attacks using static power, ideally only 1 measurement is needed to represent the static power consumption corresponding to each different input/output pair. That is, only 2 measurements are needed in each clock period, one when the clock is high and the other when the clock is low.

2. The profiling traces are processed, using statistical techniques such as PCA and KL divergence to select n ($n \leq N$) points-of-interest (POIs) where larger statistical differences between sets show up. This step compresses the dimensionality of the raw data. Some noise can be removed in this step, and reduced dimensionality also helps improve the efficiency of matrix arithmetic in step 3. Another benefit from this step is that it can address the issue of singular covariance matrix when using a multivariate Gaussian distribution to model the data. In order to differentiate from the POIs in the original traces, we refer to

each point in the PCA-processed trace as a *dimension*.

3. KDE is applied to each of the K sets of traces separately. For each possible subkey $k \in [0, K-1]$, let the trace vector generated using subkey k be $\mathbf{t}_{k,i}, 1 \leq i \leq M_p$. Note that the traces only contain chosen POIs now. This step results in the PDF for each template set, represented by KDE functions:

$$\hat{f}(\mathbf{t}_k|k) = \frac{1}{M_p} \sum_{i=1}^{M_p} K_{\mathbf{H}} \left(\mathbf{t}_k - \mathbf{t}_{k,i}\right), \tag{5.1}$$

where $\mathbf{t}_k$ is the random variable that represents the set of profiling traces generated using k.

4. A set of M_t traces is obtained from the target device using random plaintexts and the unknown key. The measurement points and the POIs are chosen to be the same as those in the profiling phase. If PCA is applied, then we need to apply the projection matrix to the attack traces as well. Given trace $\mathbf{t}'_i$ $(1 \leq i \leq M_t)$ in the trace set, the probability that this trace is derived from subkey k, according to Bayes' rule, follows

$$Pr(k|\mathbf{t}'_i) = \frac{\hat{f}(\mathbf{t}'_i|k)Pr(k)}{f(\mathbf{t}'_i)}, \tag{5.2}$$

where $Pr(k)$ is the probability that key k is the correct key and $f(\mathbf{t}'_i)$ is the PDF function of $\mathbf{t}'_i$. Ideally the distribution of k and $\mathbf{t}'_i$ are unbiased, which means that $Pr(k)$ and $f(\mathbf{t}'_i)$ are fixed values. Hence $f(\mathbf{t}'_i|k)$ reflects the likelihood that key k is used given $\mathbf{t}'_i$, and we define the likelihood to be:

$$\mathcal{L}(k|\mathbf{t}'_i) = \hat{f}(\mathbf{t}'_i|k) \propto Pr(k|\mathbf{t}'_i). \tag{5.3}$$

We could use $\hat{f}(\mathbf{t}'_i|k)$ as a simple metric for $Pr(k|\mathbf{t}'_i)$. The subkey with the highest probability is assumed to be the correct key used in the target device.

Steps 1-3 are defined as the *profiling phase*, while step 4 is defined as the *attack phase*.

The original proposal in [11] applied template attacks to stream ciphers, in which they assume that only 1 attacking trace can be used to perform a template attack. In our scenario targeted at block ciphers, we are able to acquire multiple target traces. Since all the traces are acquired independently, by applying the maximum likelihood estimation, we define the overall likelihood, $\mathcal{L}_k$, given all the traces as

$$\mathcal{L}_k = \prod_{i=1}^{M_t} \mathcal{L}(k|\mathbf{t}'_i) \propto \prod_{i=1}^{M_t} Pr(k|\mathbf{t}'_i). \tag{5.4}$$

We find that $\mathcal{L}_k$ may exceed the data range limit of programming languages if the number of traces is too large. Hence, in our experiments, we use the normalized log-likelihood, which is

$$\mathrm{NLL}_k = \frac{\ln \mathcal{L}_k}{M_t} = \frac{\sum_{i=1}^{M_t} \ln \mathcal{L}(k|\mathbf{t}'_i)}{M_t} \tag{5.5}$$

Note that applying the logarithm and dividing by a constant will not alter the relation of one likelihood value being greater than another.

5.2.2 Discussions on Profiling Phase

The original proposal of template attacks in [11] suggested using the Gaussian distribution model to identify the points where the Gaussian fits well. In our case we are dealing with distributions which might be more complex than Gaussian (as we will show later), hence we use non-parametric methods to pick the POIs with

larger difference. More specifically, we use KL divergence to measure the univariate divergence from the target set to each template set for each POI, then we choose the POIs where the divergence is significantly larger.

In a noise-free simulation environment some of the measurement points may be highly correlated, which would cause the covariance matrix in step 3 to be singular. PCA can be applied in step 2 to address this issue. Alternatively we could diagonalize the covariance matrix in step 3, by which we simply remove any possible correlation between the measurement points at the cost of removing all the information about the relation between different measurement points.

Note that PCA can also be used for dimensionality reduction in step 2. We will use our simulation results to discuss the performance of PCA versus KL divergence for this purpose in later sections.

5.2.3 Statistical Methodology

If we abstract template attacks to a high level, it can be seen that the approach falls into a category of statistical problems known as *classification*. Hence rather than a single algorithm, template attacks build a framework where we can switch out any component and plug in new statistical tools.

In this chapter, we propose to use kernel methods to model the static power in step 3, but it is easy to switch to multivariate Gaussian. We believe that, like in other classification problems, there is no ultimately optimal universal solution for all potential data sets. For example, the kernel methods may not be superior to multivariate Gaussian methods when considering the factors such as accuracy of modeling, computational speed etc. We will discuss this more thoroughly in the later sections of this chapter using the data from our simulations and compare kernel

methods with the Gaussian approach.

Besides the modeling method, the so-called distinguisher used in step 4 can also be changed with other statistical tools. We have demonstrated how to replace (5.4) with other tools such as KL divergence and its application to static power in [39]. Our work in [39] was focused on parametric methods, while KDE is a non-parametric method. We believe that the advantage of KL divergence in computational speed will not be so significant when it comes to non-parametric PDFs, hence we straightforwardly use (5.4) as the distinguisher in our work.

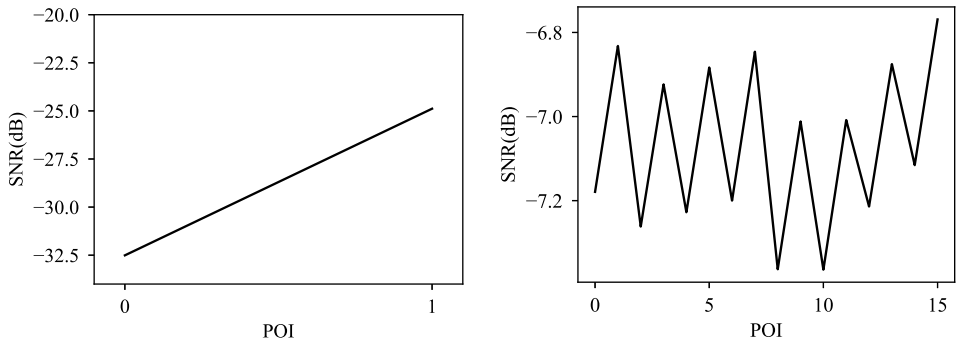
5.3 Attack Results and Discussions

In this section we demonstrate the attack results using different setups.

5.3.1 SNR of Circuits

Before performing the actual attacks, we analyze the SNR of the traces achieved using the two targets. We take all the generated traces and compute the estimated SNR following (2.40). Figure 5.1 demonstrates our estimation result, giving the SNR for each POI in the data. Our SNR result can be compared to the estimations given by [17], where an AES S-box designed using 65-nm technology is used as the target. The SNR of static power leakage in [17] is between $-10\,\mathrm{dB}$ and $-7\,\mathrm{dB}$. The SNR of our S-box circuit is much smaller than that of the circuit used in [17], while the SNR of our serialized SPN circuit is at about the same level. For both of our targets, the SNR of static power leakage is much smaller than the dynamic power leakage in [17], which is as high as $3\,\mathrm{dB}$.

The estimation in (2.40) invokes only the first and second-order statistical moments, which is also used by conventional template attacks [11] to build Gaussian estimations.



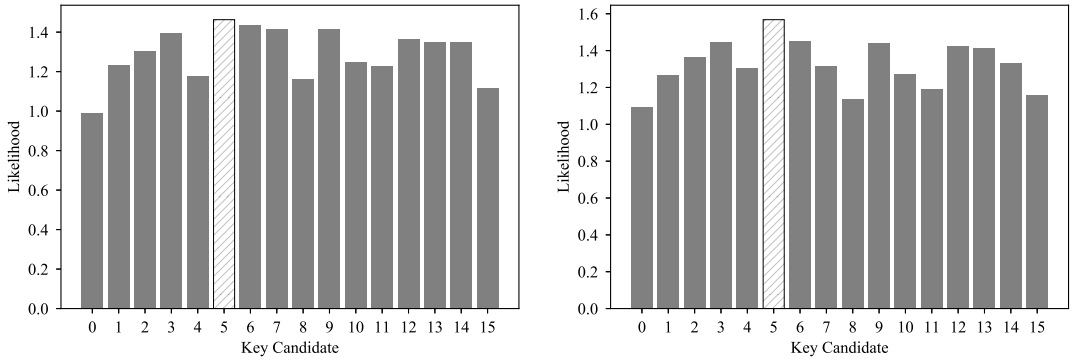
(a) S-box (b) Serialized SPN

Figure 5.1: SNR of the S-box circuit and the serialized SPN cipher.

Theoretically the first two moments do not guarantee a complete representation of the data's statistical characteristics, and it is proven in [79] that this incompleteness applies to SCA. However, the SNR estimation highlights that the difficulty in performing template attacks using conventional Gaussian estimation of the static power of our targets is comparable to the difficulty of previous works using a more complex circuit.

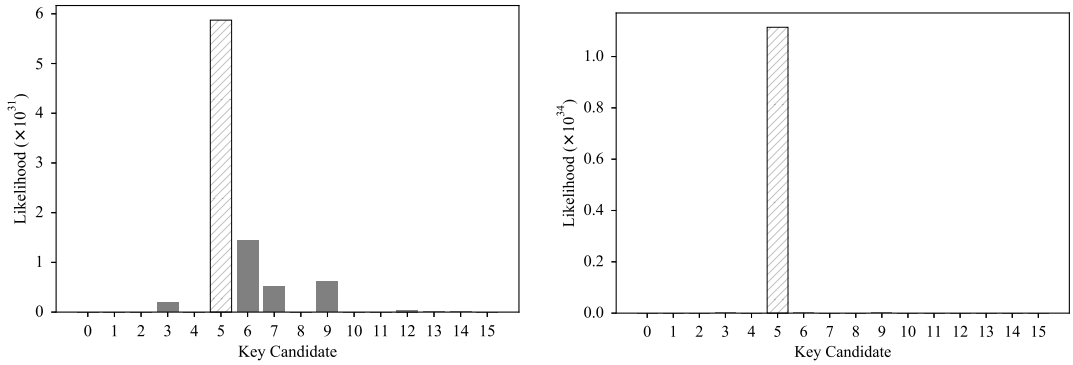
5.3.2 Attacking the S-Box Circuit

We now show a simple attack of the S-box circuit using the static power. For each key guess, we profile using different sets of 50 random plaintext inputs. Next, assuming that we attack key 0101 (5 in decimal), we use this key to encrypt another set of 50 different random plaintext inputs. After performing PCA to project the original data to a new 2-D space, we model the projected static power traces using two methods: the original multivariate Gaussian and the kernel method. Then we project the target set to the corresponding dimensions and compute NLL_k for each template set. The result of this attack is shown in Figure 5.2. The highest likelihood is represented by white bars, and in this attack both the Gaussian and the kernel methods correctly recover the key. Since the difference between log-likelihoods may not be easily visible,



(a) Gaussian (b) Kernel

Figure 5.2: Template attack results on key 0101 using 50 traces for template and 50 traces for attack. The likelihoods in this figure are computed using (5.5).



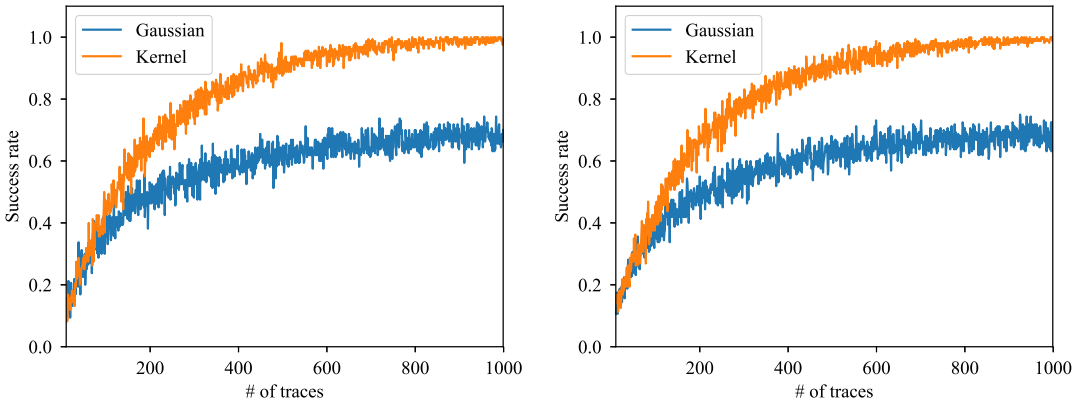
(a) Gaussian (b) Kernel

Figure 5.3: Template attack results on key 0101 using 50 traces for template and 50 traces for attack. The likelihoods in this figure are computed using (5.4).

we also provide the results computed using (5.4) in Figure 5.3.

The success rate is generated by performing 10 attacks for each of 16 different target keys using a fixed number of randomly chosen traces for profiling and attack ($M_p = M_t$). The traces are re-selected for each different attack to ensure maximum generality.

From Figure 5.4 we clearly see that the kernel-based attacks achieve a much higher success rate than the conventional Gaussian estimation, with or without PCA. In this scenario we do not reduce the number of dimensions by applying PCA. Instead its



(a) Without PCA (b) With PCA

Figure 5.4: Success rate of attacking the S-box circuit.

purpose is only to remove the linear dependencies between POIs and avoid a singular covariance matrix for the multivariate Gaussian model. This could be also achieved by empirically increasing the number of traces [41] or diagonalizing the covariance matrix [39]. However, using PCA provides a deterministic mathematical process that does not need to empirically estimate the number of traces needed to avoid singularity and does not remove correlation information from the original covariance matrix, and hence can be expected to have a better result.

5.3.3 Analysis of S-Box Attack Results

Since histograms are non-parametric and do not pose any assumption on characteristics of the observed PDF, we use histograms to characterize the trace data and plot the results in order to numerically analyze the difference between Gaussian and kernel modeling. We reproduce both phases of the template attacks. That is, we take the profiling traces and compute the MISE between the histograms and the profiling results (Gaussian and kernel), then we model the target traces using Gaussian and kernel, and compute the MISE between the modeled target traces and the modeled profiling

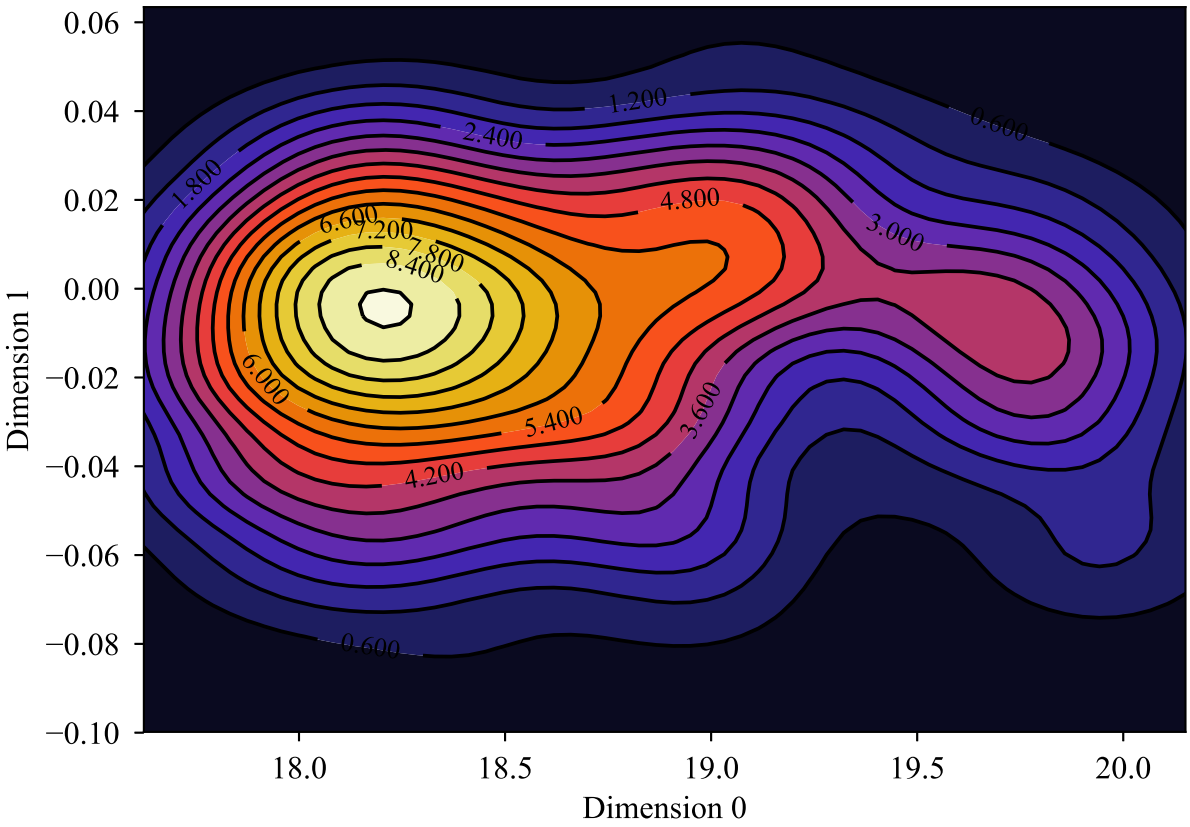
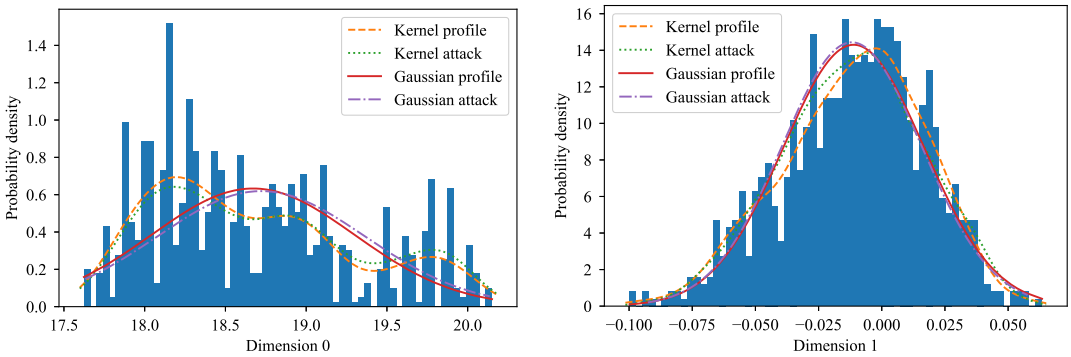


Figure 5.5: Distribution of the projected S-box traces in the projected 2-d space.

traces. We do this to determine how close the derived Gaussian and kernel model are from the histogram in the profiling phase. Note that strictly computing the MISE requires a closed-form equation of functions f and g. However in a non-parametric fitting problem, we are not able to derive the a priori PDF without any assumption of the original distribution, hence we are using histograms to estimate the MISE. To ease the process of our analysis, we work on the projected traces of one specific key, but the conclusion holds for all other keys since the distribution of different key sets are very similar.

Here we choose the key 0000 and the singular vector for the profiling set is $[0.39739919, 0.00077894]$. Figure 5.5 shows the general distribution of the projected data, and Figure 5.6 demonstrates the fitting results of each dimension. Table 5.1



(a) Probability density fitting results for dimension 0.

(b) Probability density fitting results for dimension 1.

Figure 5.6: Fitting results for each projected dimension of the S-box traces.

Table 5.1: MISE analysis results of S-box circuit

	Kernel		Gaussian	
	Profile	Attack	Profile	Attack
2-d	41.365154	0.208276	43.660649	0.042259
Dim 0	0.060989	0.001125	0.297575	0.000615
Dim 1	1.945094	0.774198	1.730859	0.035433

shows the MISE for each dimension and the combined multi-dimensional case.

Overall the kernel method outperforms the Gaussian approach in fitting the data since the general distribution is not Gaussian-like, despite the fact that Gaussian has less error in fitting the target set. Breaking down to each dimensionality, we can clearly observe that dimension 0 is far more dominant than dimension 1. The fitting of dimension 0 using the kernel is significantly better than fitting using the conventional Gaussian, although the fitting of attacking sets using kernel is slightly worse than using Gaussian. The fitting of dimension 1 using Gaussian is actually better than using the kernel approach, however this dimension contributes little to the general PDF.

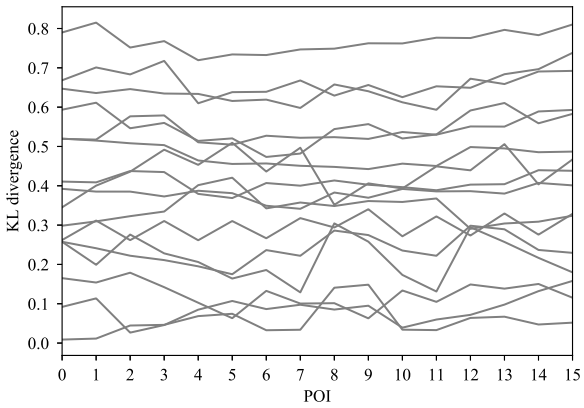


Figure 5.7: KL divergence between profiling sets and the target set, $M_p = M_t = 1000$.

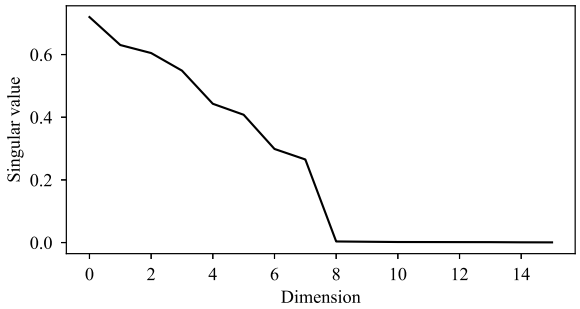
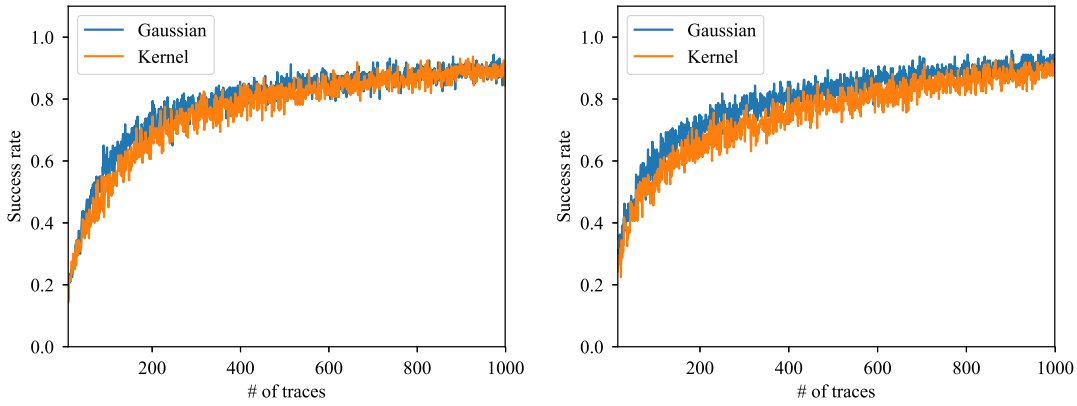


Figure 5.8: Singular values of a profiling set for the SPN cipher.

5.3.4 Attacking the Serialized SPN Circuit

We attack the SPN circuit following a similar routine to the S-box case. The difference in this attack is that originally there are 16 POIs in the traces. Hence we could reduce the dimensionality in the profiling phase.

Figure 5.7 shows the KL divergence results between the target set and all the templates, and it is hard to find a specific set of POIs in which the KL divergence is relatively larger. When we apply PCA to the template sets, the result is Figure 5.8, which shows the singular values for one of the templates. It is clear that the first 8 dimensions contribute more to the overall distribution. The ratio of the sum of the first 8 singular values to the overall sum of singular values is greater than 99.6%,



(a) Using the threshold of 99%

(b) Using the first two dimensions

Figure 5.9: Success rates of attacking the serialized SPN circuit. The PCA-processed dimensions are chosen by different strategies.

and the ratio of the first 7 singular values is around 92.9%. We could empirically choose a threshold of 99%, which means that we pick the first M dimensions, and these dimensions should satisfy that the ratio of the sum of their singular values to the overall sum is greater than 99%. In fact, we could get use an even smaller threshold without sacrificing the attack results too much.

Figure 5.9 is generated using the same methodology as the attack of the S-box circuit. We observe that the difference between choosing the threshold of 99% and using the first two dimensions is not very significant. However, in this case we observe that the performance of the kernel method is slightly worse than Gaussian. We are interested in this difference and shall discuss this in the next section.

5.3.5 Analysis of Serialized SPN Attack Results

We analyze the trace data using a similar methodology to the S-box attack. In order to better visualize the data, we focus on the first 2 dimensions.

Figure 5.10 demonstrates the combined distribution and we observe that it is more Gaussian-like than the S-box PDF. The MISE analysis results are shown in Table

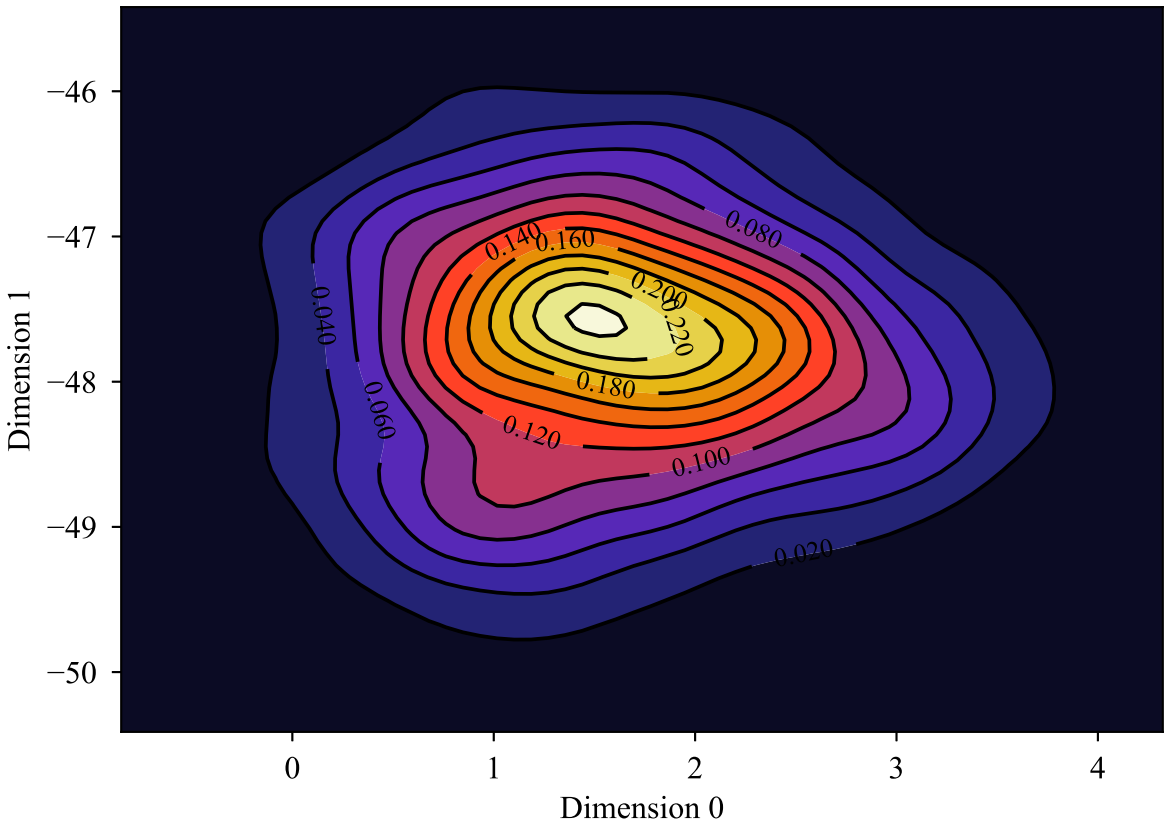
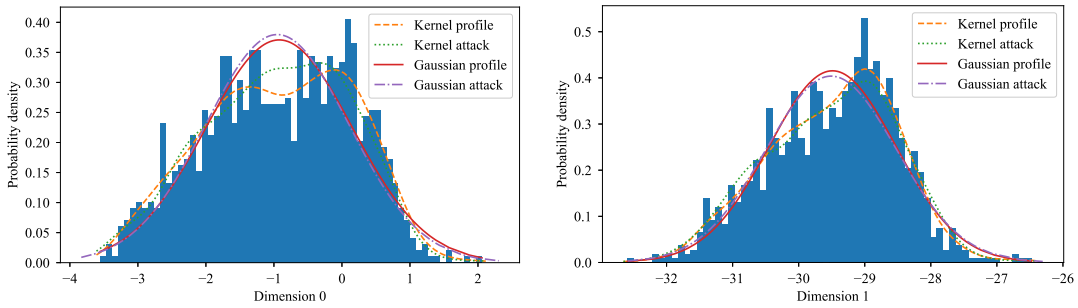


Figure 5.10: Distribution in the projected 2-d space.



(a) Probability density fitting results for dimension 0.

(b) Probability density fitting results for dimension 1.

Figure 5.11: Fitting results for each projected dimension of the SPN traces.

5.2. We can see that for the 2-dimensional distribution case, similar to the S-box case, the kernel method fits the profiling set better than the Gaussian method. However, the kernel approach results in larger MISE for the target set, which means that it is

Table 5.2: MISE analysis results of SPN circuit

	Kernel		Gaussian	
	Profile	Attack	Profile	Attack
2-d	3.158×10^{-3}	3.156×10^{-5}	3.515×10^{-3}	2.838×10^{-6}
Dim 0	1.315×10^{-3}	5.435×10^{-4}	5.987×10^{-2}	4.629×10^{-5}
Dim 1	1.180×10^{-3}	2.847×10^{-4}	5.469×10^{-2}	6.613×10^{-5}

overfitting compared with the Gaussian method.

From Figure 5.11 we observe that the fitting result of the profiling set using Gaussian method is not so much worse than the kernel approach, since the distribution is more Gaussian (compared with the S-box traces) in the SPN case. In this scenario, the error caused by overfitting would result in the kernel method recognizing the target set more poorly than the Gaussian approach. As a result of these effects, the kernel and Gaussian methods perform similarly in the SPN case, with the kernel approach being only slightly worse than the Gaussian method, as is shown in Figure 5.9b.

5.4 Effect of Gaussian Noise

In the attacks in Section 5.3, the only noise source is the random plaintext input. Although we have shown that the traces we acquire from the simulated circuits have the same level of SNR compared with the real-world chips used in [17], there still exists other noise sources we did not take into consideration in our simulations. These include the activity of other on-board components and noise from measurement devices. Commonly it is assumed that the combination of all the different sources of noise is Gaussian white noise. The existence of Gaussian white noise will result in the overall distribution being more Gaussian-like. In the attacks on the SPN circuit we observe that the kernel method would perform slightly worse than the Gaussian approach when the distribution of traces is Gaussian-like. To further explore the performance of the kernel method and the original Gaussian approach, as well as to better reflect

attacks on realistic circuits, we add Gaussian white noise to the original traces.

We know from the previous section that the distribution of the traces acquired from the SPN circuit are already quite close to Gaussian, while the distribution of S-box traces are less Gaussian-like. Hence it makes more sense to add Gaussian white noise to the S-box data and observe the changes in the attack results.

In this section we use MISE to analyze the modeling errors. Similar to the attacks in previous sections, we acquire 16 profiling sets using all key possibilities and one target set. Then we measure MISE between the profiling set generated using the correct key and the target set. This represents the modeling error between the profiling and attack phase, which is caused by overfitting. We also measure the MISE generated between the profiling sets generated using wrong keys and the target set, and record the MISE which is closest to that of the correct key. This represents the smallest statistical difference between profiling sets generated using wrong key guesses and the target set. If this difference is too close to the error caused by overfitting, then it would be likely that the distinguisher chooses the closest wrong key as the key used in the target set. Finally, we repeat this process by generating 16 different target sets using different keys (0000 to 1111), and compute the average of MISE across the 16 tests. The standard deviations of the additive Gaussian white noise are 0.01 A, 0.03 A, 0.05 A, 0.07 A and 0.1 A.

Figure 5.12 shows the results of this analysis using the S-box traces, where the label "Kernel, correct key" represents the MISE between the target set and the profiling set generated using the correct key, the label "Kernel, wrong key" represents the closest MISE between the target set and a profiling set which is generated using a wrong key, and likewise for the Gaussian labels. An obvious observation is that the MISE decreases with more Gaussian white noise added. This means that with more noise added the difference between each set becomes smaller. There would be less

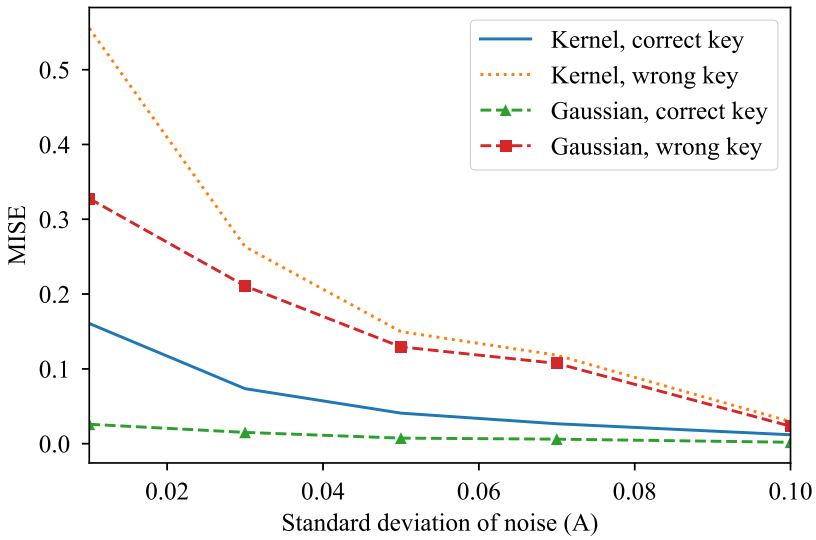
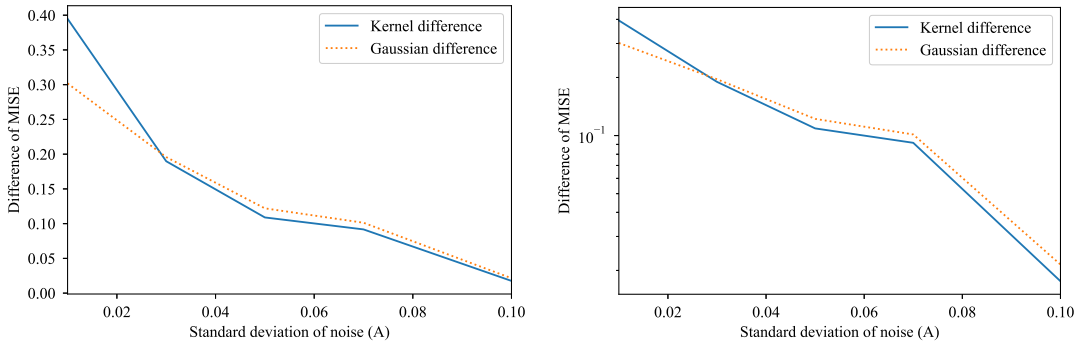


Figure 5.12: MISE with relation to the standard deviation of the additive noise, using the S-box traces. The success rates of attacks using the data with additive noise are shown in Figure 5.16.



(a) Difference of MISE is represented as a linear axis

(b) Difference of MISE is represented as a logarithmic axis

Figure 5.13: Difference of MISE with relation to the standard deviation of the additive noise, using the S-box traces.

overfitting of the correct profiling set using the kernel method, but the wrong keys are even harder to distinguish. This observation holds with the knowledge that more noise means lower SNR, which results in lower success rate. This can be verified by the success rates shown in Section 5.5.

Next we compare the difference between the MISE of the correct key and the

closest wrong key using MISE results from Figure 5.12. Larger difference means that it is easier to distinguish the correct key, while smaller difference means it is harder. Figure 5.13 shows the difference for the kernel method and the Gaussian approach. We see that when the noise amount is small, the kernel method results in significantly larger difference than the Gaussian approach, which means that the kernel method can better distinguish the correct key, hence the success rate would be higher. With more noise added, the difference of using the kernel method becomes slightly smaller than the difference of using the Gaussian approach. In such scenarios the kernel method would not outperform the Gaussian approach. Note that although the kernel method has smaller difference, it is still very close to the difference of the Gaussian method, hence the kernel method will not be significantly worse than the Gaussian approach in the sense of success rate. This is shown in Figure 5.16 for the S-box circuit.

There are two questions concerning these observations: 1) Do we need to further optimize the kernel method to match up with the performance of conventional Gaussian? 2) Does the kernel model still apply to static power leakage?

The answer to the first question could be the famous "no free lunch" theorem [80]. We should note that the advantage of Gaussian over kernel in a Gaussian-like scenario is very limited. How to optimize the kernel algorithm to fit a given set of data is a problem that requires a great deal of trial and error. Given the condition that the attacker has access to the template sets, it would be straightforward for him/her to directly observe the templates and determine whether it is Gaussian-like. Based on this, the attacker can easily choose between the conventional Gaussian and kernel methods.

The answer to the second question is not deterministic. Conventionally, the Gaussian model is widely accepted for SCAs using dynamic power, because there are many POIs in a dynamic power trace (10s to 100s) and according to the central

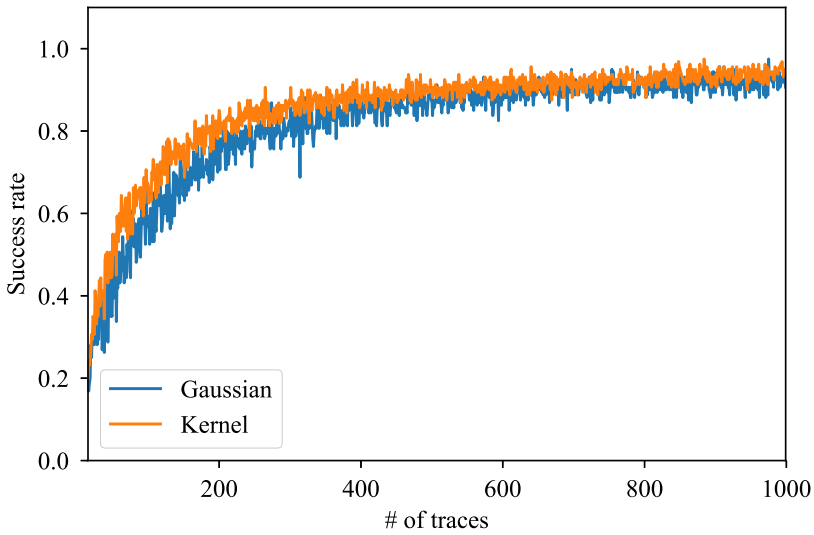


Figure 5.14: Success rate of attacking serialized SPN using POIs 14 and 15.

limit theorem, it is reasonable that the combined distribution is Gaussian. However in static power there are only 2 POIs for each clock period and the overall number of useful POIs can be far less than a dynamic power attack. Hence we cannot always guarantee that the amount of dimensionality is high enough for the central limit theorem to apply. For example, if we attack the serialized SPN circuit using its last two POIs, the success rate would be as shown in Figure 5.14, showing that the kernel method actually slightly outperforms the Gaussian model. Also, the characteristics of the target device also affects the attack results. Figure 5.15 shows the attack results using a circuit designed with the Cadence AMS kit library [71]. Contradictory to the success rates shown in Figure 5.9, the kernel method performs slightly better than Gaussian model. In both of these cases the kernel result has a better success rate than the Gaussian. The attacker needs to carefully observe the PDF of the templates to determine whether the Gaussian model fits better than the kernel.

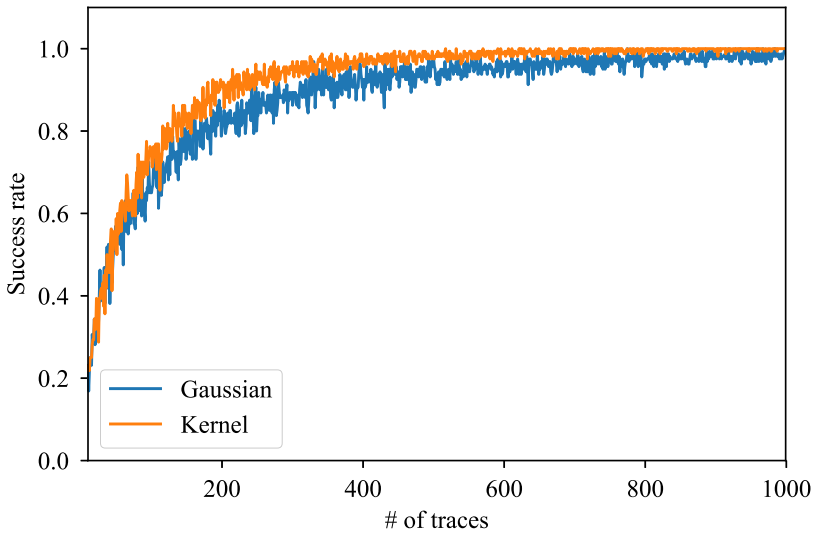


Figure 5.15: Success rate of attacking serialized SPN generated using the AMS kit, the attack uses POIs from 9 to 15.

5.5 Attack Results With Additive Noise

In this section, we demonstrate the success rates of attacks against the S-box circuit and the serialized SPN circuit with extra Gaussian white noise added to the traces. By varying the standard deviation of additive noise, we acquire the success rates shown in Figure 5.16 and Figure 5.17. Note that the success rates of attacks without additive noise are shown in Figure 5.4 and 5.9. As expected, as the noise standard deviation increases, the success rate decreases and the advantage of the kernel approach in the S-box case diminishes.

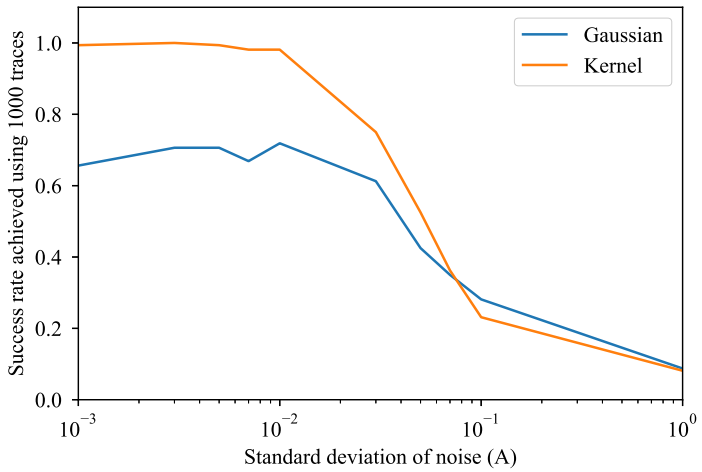


Figure 5.16: Success rate of attacking the S-box cipher using 1000 traces. 10 attacks are performed for each of the 16 key possibilities.

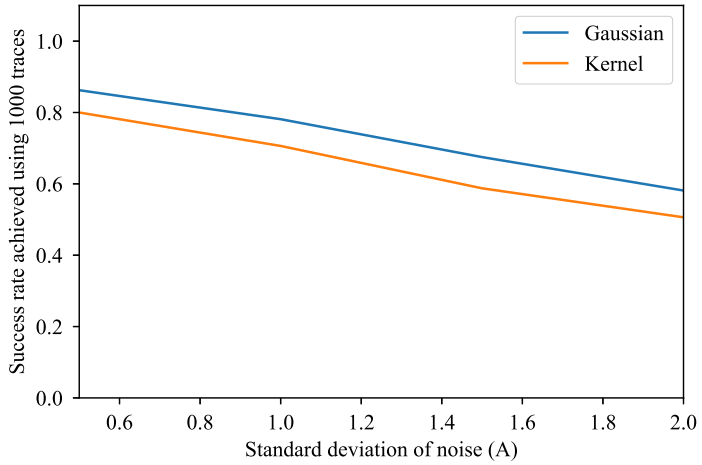


Figure 5.17: Success rate of attacking the serialized SPN cipher using 1000 traces. 10 attacks are performed for each of the 16 key possibilities.

5.6 Conclusion

In this chapter we propose a new profiled attack for static power analysis – the kernel-based template attack. This method is non-parametric and does not rely on any assumption on the a priori statistical properties of the attack target. We demonstrate that this method results in significantly better success rate than the conventional

Gaussian for certain types of static power leakage where the PDF is not Gaussian. For scenarios where the PDF is Gaussian, this method might result in slightly worse, but still very close performance compared with the Gaussian method.

Our work brings the conventional template attacks [11] into a new application scenario where we do not have to accurately sample the shape of the waveform caused by dynamic power. We demonstrate that in the context of static power leakage, PCA pre-processing outperforms manually choosing the POIs using KL divergence. As for static-power-based template attacks, we are the first to bring kernel methods into this scenario compared with conventional works [17, 18], and we have discussed about the cases in static-power-based attacks where kernel methods may be applicable and outperform the Gaussian approach.

Though we have achieved success using this new attack method, we note that many of our analyses given on this method are based on numerical or empirical observations. A more concrete mathematical proof of its performance is to be finished. We leave this as future work. As well, in the long term, analysis of implementation of realistically sized circuits with real measurements should be considered for future work. This would include different target devices designed using different CMOS technologies

In the next chapter we will look into the effect of profiled static power analysis on protected circuits, and compare the effect of static versus dynamic power as the leakage source.

Chapter 6

Template Attacks of a Masked S-Box Circuit: A Comparison Between Static and Dynamic Power Analyses

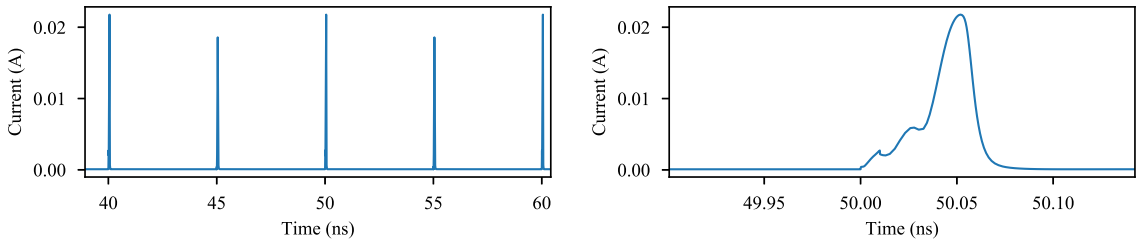
In this chapter we perform template attacks on the masked S-box circuit introduced in Section 3.4. We are the first to compare template attacks using static power and dynamic power in the context of masked S-box implementations. This countermeasure [69] is proposed to be secure against dynamic power analysis attacks such as DPA and CPA. It would be of interest to explore its security against profiled attacks and static-power-based attacks. We are able to achieve successful results using both types of power leakage. However, we observe that, in the 45-nm environment, dynamic power analysis requires a high sampling rate for the oscilloscopes used to collect data, while the results of static power analysis are more sensitive to additive noise. The work presented in this chapter has been published in [81].

6.1 Attack Methodology

In this section, we introduce the environment and steps we take to perform the attacks.

The target circuit is the masked S-box circuit introduced in Section 3.4.In the simulated attack, the target circuit is synthesized using Synopsys design compiler and the FreePDK 45-nm standard cell library, followed by placement and routing using Cadence Encounter. The final netlist and delay file are imported and simulated using Cadence Virtuoso with Spectre simulator.

The circuit is designed and simulated at the clock frequency of 100 MHz. During the simulation, we measure the current at V_{dd} at each different time. The measurement result for one full encryption is known as a trace. Figure 6.1 shows an example of a trace captured in simulation. We observe from Figure 6.1a that there are spikes every 5 ns, at times when clock changes happen. In our attacks, these spikes are used to represent the dynamic power consumption, while the remaining flat part of the traces are used to represent the static power consumption. In order to perform a successful attack using dynamic power, we need to accurately capture the shape of these spikes, while only one measurement is needed for static power after each clock change, since theoretically static power remains constant if the inputs/outputs are not changed.



(a) Part of a full trace

(b) Zoomed in around 50 ns

Figure 6.1: An example captured current trace.

Figure 6.1b shows the zoomed in trace, in which we observe that the spike happens

from 50 ns to around 50.08 ns. In order to capture the shape of these spikes, we sample from 0.05 ns before the clock change to 0.15 ns after the clock change with an interval of 0.01 ns, which is a sampling rate of 100 GS/s. As for static power, we sample at 4 ns after the clock change. This sampling rate is 200 MS/s, and it can be even smaller if the attacker has control over the clock, which is practical in some environments (as is shown in [17, 82]). This measurement setup shows one of the advantages of static power analysis: it requires a smaller sampling rate, and hence it can be carried out using relatively lower-end oscilloscopes compared with dynamic power analysis.

We perform template attacks using the Gaussian model and conventional distinguisher as is described in Section 2.5.1. The practicality of conventional template attacks is limited by the complexity of computing high dimensional distributions and the occurrence of a singular covariance matrix [41]. In our scenario it takes 4 clock cycles for the target circuit to perform a full encryption, resulting in 160 measurement points for dynamic power analysis and 8 measurement points for static power analysis. We perform principal component analysis (PCA) [42] to reduce the dimensionality and remove singularity in the covariance matrix. In our practice we use a threshold of 99% when performing PCA. That is, we pick the n dimensions with highest eigenvalues and ensure that the sum of the eigenvalues of these n dimensions is greater than 99% of the sum of all the eigenvalues. After performing PCA, the dimensionality is reduced from 160 to 9 for dynamic power traces and from 8 to around 3 for static power traces. Here we see another advantage of static power analysis: there is less dimensionality in the distribution model, hence computationally it is faster to perform a static power analysis than a dynamic power analysis.

6.2 Experimental Results

In this section we discuss the results of our attacks on the target circuit.

For all the attacks in this chapter, the profiling sets and target sets are generated using the same set of randomly generated plaintext inputs and different keys. In other words, we are performing known plaintext attacks in this chapter. It is also possible to perform unknown plaintext attacks by means of template attacks. However, we observe that the superiority of one statistical method does not change whether it is applied to known plaintexts or unknown plaintexts. The reason for this is that when using known plaintexts, we choose the plaintexts to be uniformly distributed. It is reasonable to assume that the plaintexts also follow uniform distribution if the size of plaintext sets are large enough when using unknown plaintexts, since in this case the plaintexts are randomly chosen. Hence in our scenario we choose known plaintext attacks for simplicity.

The quality of attacks are evaluated by the first-order success rate defined in [66], which refers to the ratio of attacks that the correct key is properly chosen versus all the attacks. We set the number of traces in the profiling sets and target sets to be the same, and this number increases from 15 to 300. For each different trace number, 10 attacks using different plaintext sets are performed for each correct key. In other words, for a 4-bit key setting there are 160 attacks performed for each different trace number.

Figure 6.2 shows the success rates of static- and dynamic-power-based template attacks on the target circuit. Both methods are able to reach a success rate of 90% with less than 100 traces. Hence this masking scheme designed for CPA may not be secure against both types of template attacks. Besides, we observe that when there is a low number of power traces, static power analysis outperforms dynamic power traces.

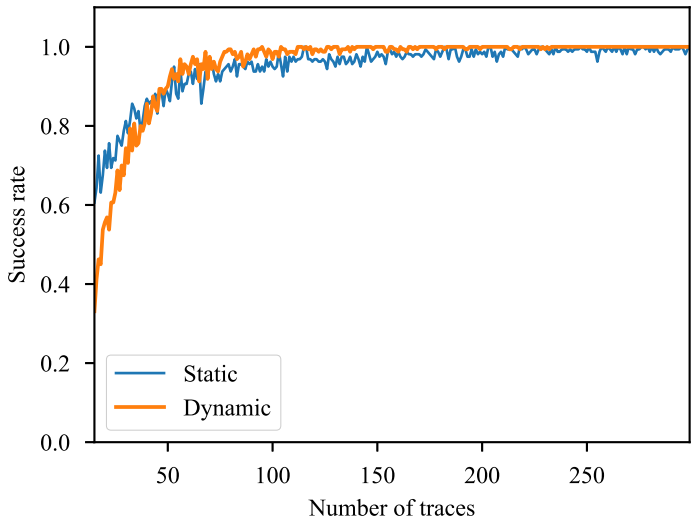


Figure 6.2: Success rates of static and dynamic power attacks on the target circuit.

When there are enough traces, dynamic power analysis performs slightly better.

As we have mentioned before, the dynamic success rate in Figure 6.2 is based on the traces sampled at the rate of 100 GS/s, which requires high-end equipment. Now we reduce the sampling rate to 20 GS/s, and the success rates are shown in Figure 6.3. In this scenario dynamic-power-based template attacks perform worse than static-power-based template attacks, although the sampling rate used for the dynamic power traces is still much higher than the 200 MS/s used for static power traces. For a real-world scenario, it is possible that attacker would be able to successfully perform static power analysis using lower-end equipment compared with dynamic power analysis.

Comparing our work with the results from [17] and [82], one might notice that in our case the static-power-based attacks are far more successful. One of the reasons to this is that in simulation there is no environmental noise considered, and static-power-based SCAs are actually more sensitive to environmental noise compared with dynamic-power-based SCAs.

The signal-to-noise ratio (SNR) of the static and dynamic power traces used for

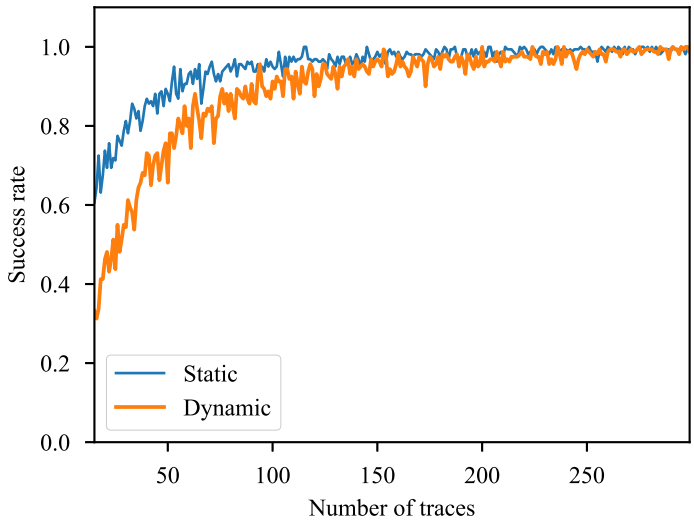
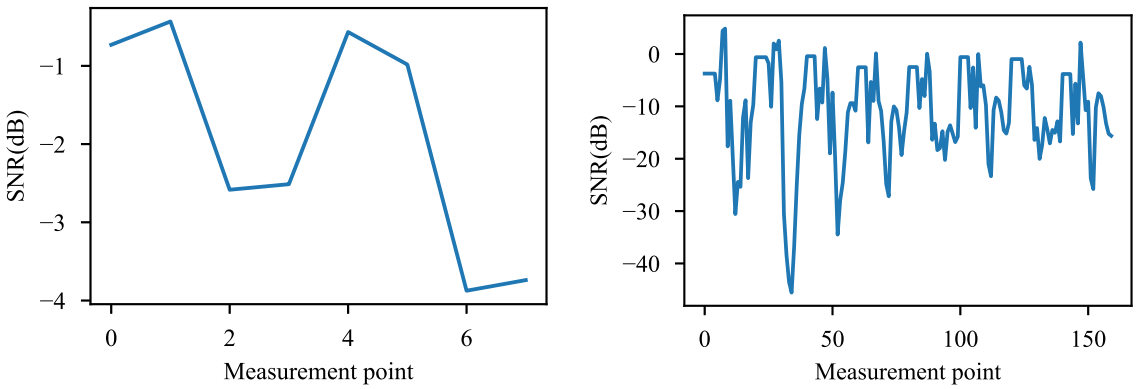


Figure 6.3: Success rates of static and dynamic power attacks on the target circuit. The sampling rate for dynamic power traces is 20 GS/s

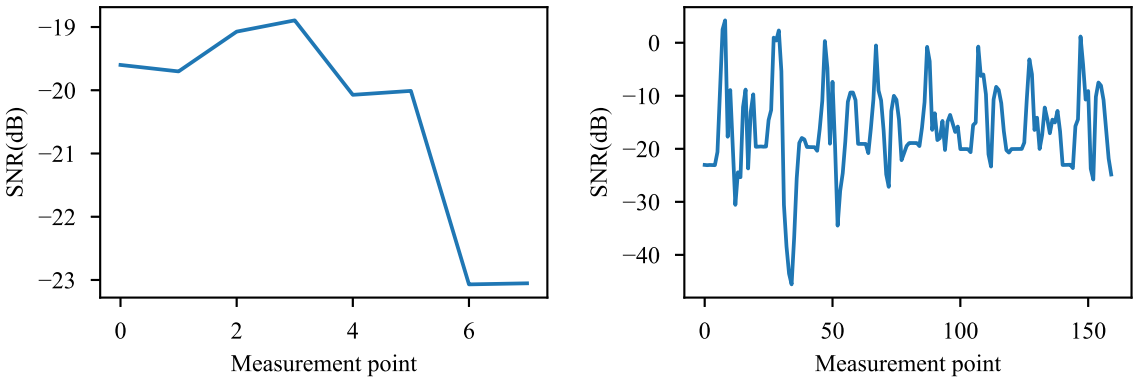
Figure 6.2 are shown in Figure 6.4. Now we add Gaussian white noise with a standard deviation of 0.007 to the traces and the results are shown in Figure 6.5. We see that the SNR for static power drops from around −2 dB to around −21 dB, while the measurement points with higher SNR in the dynamic power traces do not change very much after the noise is added. The success rates of attacks on the traces with additive noise are shown in Figure 6.6. In this scenario static-power-based template attacks become significantly worse than dynamic-power-based template attacks, while the performance of dynamic-power-based attacks do not change much compared with Figure 6.2.

An interesting advantage of static power analysis is that it can easily average out environmental noise. Since theoretically static power consumption is a fixed value, all the changes we capture when the circuit is static can be regarded as noise. The attacker can freeze the clock for a period of time and take a number of measurements. Since we assume the environmental noise to be Gaussian white noise, the average of these measurements would be close to the real static power leakage. In our scenario, we take



(a) Static power traces (b) Dynamic power traces

Figure 6.4: SNR of static and dynamic power traces. The SNR here is computed using the method proposed in [64].



(a) Static power traces (b) Dynamic power traces

Figure 6.5: SNR of static and dynamic power traces with additive Gaussian white noise.

25 measurements of the traces with additive Gaussian white noise, the measurements start from 2 ns after each clock change with a timing interval of 0.1 ns, and regard the average of the noisy measurements as a single measurement point. Then we perform static power analysis using the averaged measurements and the results are shown in Figure 6.7. The success rate of static power analysis has been greatly improved compared with Figure 6.6. This verifies that the claims in [82] for CPA also hold for template attacks.

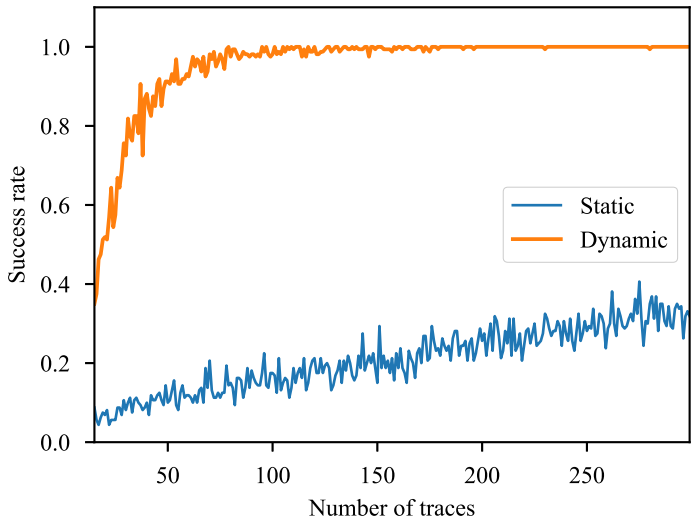


Figure 6.6: Success rates of static and dynamic power attacks on the target circuit. Gaussian white noise with a standard deviation of 0.007 is added to the traces.

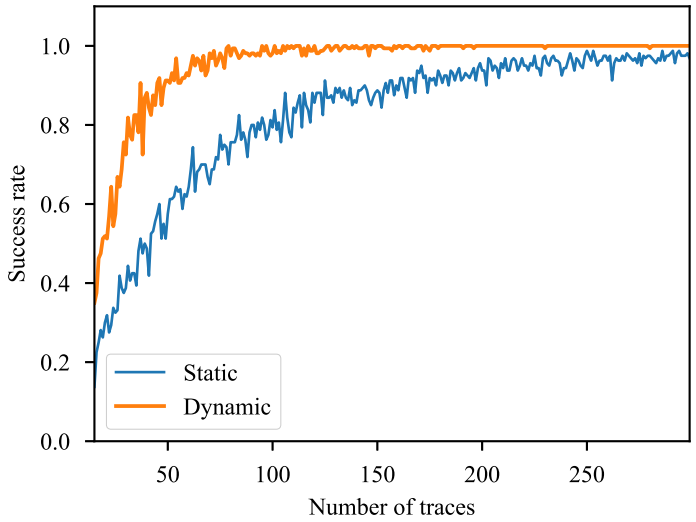


Figure 6.7: Success rates of static and dynamic power attacks on the target circuit. Gaussian white noise with a standard deviation of 0.007 is added to the traces. The static power traces are the average of 25 measurements.

6.3 Conclusion

In this chapter, we perform static- and dynamic-power-based template attacks on a PRESENT S-box masked using the threshold implementation method [69]. We observe

that the countermeasures designed for CPA may not be secure against template attacks exploiting either dynamic or static power.

Template attacks based on dynamic power consumption are less sensitive to environmental noise. However, in 45-nm technology environment the width of spikes generate by dynamic power are very small, hence requiring high-end oscilloscopes to accurately capture the shape of the spikes. Static-power-based template attacks do not rely on high sampling rates, but the current caused by static power leakage is very small, hence it is more sensitive to noise. Due to the small measurement values of static power leakage, high-end probes with amplifiers may be necessary to measure static power. If the attacker is able to take multiple measurements of the static power leakage after each clock change, it would be possible for the attacker to average out the noise affecting static power measurements.

In the next chapter, we will take a step further and explore the possibilities of combining static and dynamic power leakage in the implementation of a power analysis SCA.

Chapter 7

Using Deep Learning to Combine Static and Dynamic Power Analyses of Cryptographic Circuits

In previous chapters, we have explored the potential of static power as a new source for side-channel leakage and have compared it with dynamic power. Both types of power leakage have their advantages and disadvantages. In this chapter, we propose to use the deep neural network technique to combine the benefits of both static and dynamic power. This approach replaces the classifier in template attacks with our proposed long short-term memory network schemes. Hence, instead of deriving a specific probability density model for one particular type of power leakage, we gain the ability of combining different leakage sources using a structural algorithm. In this chapter we propose three schemes to combine the static and dynamic power leakage. The performance of these schemes are compared using the masked S-box circuit introduced in Section 3.4. The work presented in this chapter has been presented in [83].

7.1 Proposed Attacks

In this section, we present our proposed attacks. There are multiple deep learning algorithms that can be used to combine the leakage sources. In this chapter we focus

on the LSTM introduced in Section 2.6.1. There are three different schemes that employ different network structures to combine static and dynamic power traces. We refer to these as combined schemes, and specifically as: a) combined traces (CT), b) combined LSTM (cLSTM) and c) hierarchical LSTM (hLSTM).

For all the following attacks, we consider the scenarios that the attacker is able to measure a static power trace and a dynamic power trace for the same plaintext and key inputs. Due to the fact that the measurement of static and dynamic power may have different requirements on the specification of the oscilloscope, it is possible that the static and dynamic traces are measured separately using different oscilloscopes.

7.1.1 Combined Traces

The first scheme is a straightforward method. We combine the static and dynamic traces for the same input by concatenating two traces into one trace, then use the combined traces as the input to an LSTM network. The concatenation here simply concatenates two time-sequences together in time domain to get one time-sequence. This straightforward method has been used by many existing works, such as [28, 29].

The network we are using contains two LSTM hidden layers followed by a dropout layer, then a dense layer with softmax activation function is used for the output. The prediction result for a single trace is the probability that it is generated using each possible key. For a set of traces, we use the maximum likelihood method introduced in (5.4) to determine the key used to generate this set. A brief illustration of this structure is shown in Figure 7.1.

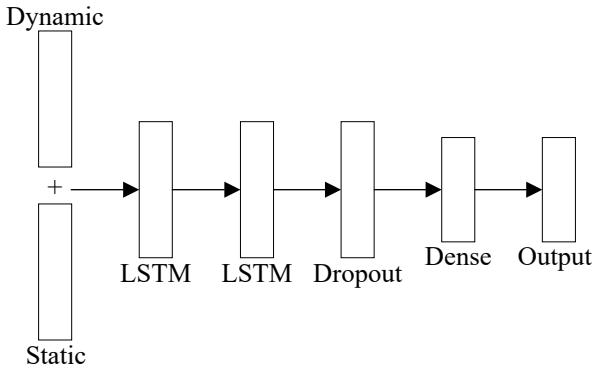


Figure 7.1: The layer structure of the deep neural network used in the combined traces scheme.

7.1.2 Combined LSTM

The second scheme is slightly different from the first one. In this case, we train the static and dynamic traces separately using two LSTM networks. Then the hidden state outputs from these two sub-networks are fed to a special hidden layer that concatenates the two parts of hidden states. The concatenation is done similar to the concatenation done in the CT scheme. Then similar to the first scheme, a dense layer is used to generate the output. This method takes advantage of the property of neural network structures, in which we can easily customize the shape of the network.

The two sub-networks have similar structure to the network in the first scheme. Both of them contain two LSTM hidden layers followed by a dropout layer. The overall structure of this scheme is shown in Figure 7.2.

7.1.3 Hierarchical LSTM

The third scheme makes use of the necessary environment for a successful static power attack. A necessary condition to perform SCA using static power leakage is that the attacker is able to identify the clock changes in the traces, so as to distinguish the part dominated by static power from the part dominated by the dynamic power. This

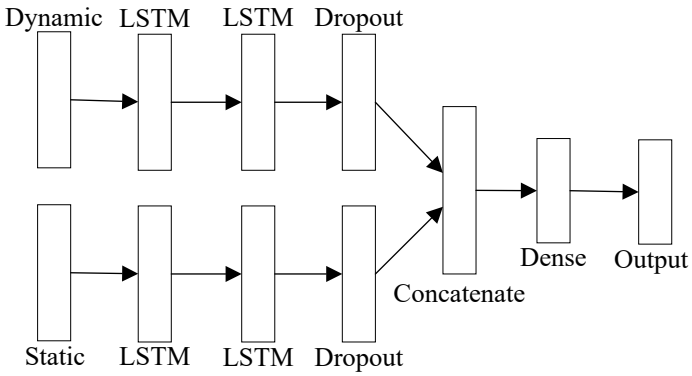


Figure 7.2: The layer structure of the deep neural network used in the combined LSTM scheme.

is usually achieved by using a slower clock or using an external input to control the clock. The requirement of being able to distinguish the static part from the dynamic part indicates that the attacker may be able to measure both parts in a specific clock period, which leads to the proposal of our third attack scheme. Proposed by Chung et. al. in [84], the hierarchical structure has become popular nowadays.

Based on the knowledge that the logical changes in a digital circuit only occurs when there is a clock change, the static and dynamic power in the same clock period can be seen as the side-channel leakage of the same logical operation. Hence we concatenate the static and dynamic power traces in each clock period separately. Together the combined trace is used to represent the logical state in a clock period. In order to take advantage of the combined traces, we use a hierarchical LSTM network in this scenario. This network structure first uses a LSTM hidden layer to encode the combined traces in each clock period. The dynamic power and static power traces in each clock period are concatenated to generate the input to the encoder LSTM layer. The encoded hidden state is then sent to 2 stacked LSTM hidden layers, which are used to classify the encoded traces. Finally, similar to the previous two schemes, a dense layer with a softmax activation function is used to generate the probability

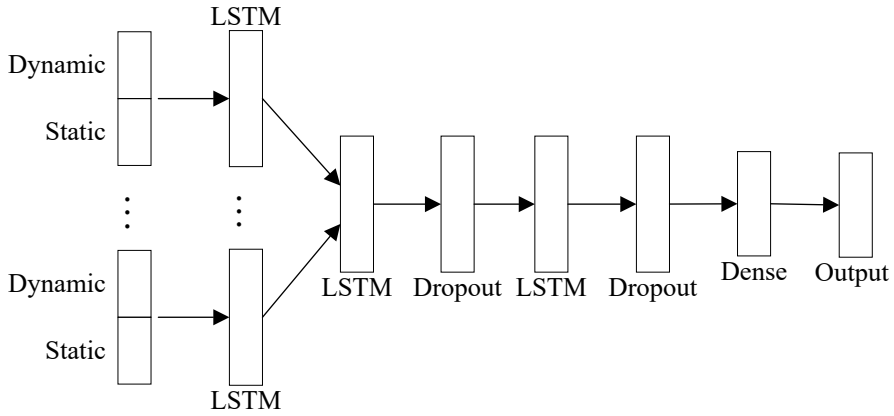


Figure 7.3: The layer structure of the deep neural network used in the hierarchical LSTM scheme.

output. In order to prevent overfitting, dropout layers are added to the outputs of both LSTM layers. The structure of the network used in this scheme is shown in Figure 7.3.

7.2 Simulation Workflow

We perform our proposed attacks on a masked S-box of the PRESENT cipher [33] introduced in Section 3.4. The circuit is designed using the FreePDK 45-nm standard cell library [72]. After performing placement and routing in Cadence Encounter on the synthesized circuit, we carry out SPICE-level simulation on the circuit in Cadence Virtuoso environment.

In our simulation, the clock input to the circuit has a frequency of 100 MHz. We capture the power behavior of the circuit by measuring the current at V_{dd}. Since we perform our attacks in a simulated environment, we capture and store the entire raw trace of each encryption using Virtuoso, then we sample the different parts in the trace with different sampling rates, in order to represent the different configurations used by the oscilloscopes when measuring the static and dynamic power. The example in

Figure 6.1 shows part of a captured trace. Spikes occur when there is a clock change. A successful attack based on the dynamic power requires the accurate acquisition of the shape of these spikes. Conversely static power leakage is supposed to be a static value that occurs when there is no power change. If the static period is sufficiently long, then the requirement of sampling rate of the oscilloscope can be lowered.

This circuit requires 4 clock cycles to execute, which means 8 different static power leakage occur in the process. Theoretically 8 measurements would be enough to perform a SCA using the static power leakage. However, considering the existence of environmental noise, it is suggested in [17] that the attacker may take multiple measurements in a static period and take the average to filter out the noise. This technique has been demonstrated to be effective to simulated circuits in Chapter 6. In this work, we take multiple measurements in a time frame of 5 ns, that is clearly in the window of a clock period associated with static power only. For the convenience of our experiments we do not extend the clock length. We observe that if we arbitrarily extend the clock length of the simulated circuit and take multiple measurements, the measurement results for the noise-free static power leakage in the same clock period would be quite identical, disregarding of the sampling rate. In other words, in the context of simulated circuits, the sampling rate does not have a significant effect on the attack results using static power. The measurement of real-world chips would be more complex and some discussions on the sampling interval are discussed in [82].

Similar 4-bit S-box circuits have been commonly used by other papers analyzing static power leakage, such as [13, 14, 18]. In order to evaluate the performance of the proposed LSTM method in different scenarios, we also add white noise to the captured traces, in order to represent the environmental noise.

7.3 Attack Results and Discussions

In this section we demonstrate the attack results using different setups. The target circuit is the masked PRESENT S-box introduced in Section 3.4.

With the help of the simulated environment, we are able to change a few parameters in our attack, so as to demonstrate the performance of each attack scheme in different scenarios. These scenarios include: 1) similar level of static and dynamic leakage; 2) more informative static leakage; and 3) more informative dynamic leakage. The level of leakage in our context is judged by performing LSTM-based profiled attacks using solely static or dynamic leakage and evaluating the success rates. If the success rates using static and dynamic leakages are close then we consider the two sources have similar level of leakage, while one source is more informative if the success rate using this leakage source is significantly larger than the other. We shall explain the method used to generate the success rates in the following sections.

In our experiments we collect 65,536 raw traces for each attack, then sample the static and dynamic parts separately. The plaintext and masking inputs are randomly chosen for each key. After collecting the leakage traces in simulation, we normalize the values to the range of [0, 1], then randomly separated these traces into three sets. The ratio of 1) training set to 2) validation set to 3) test set is 6:2:2. The training set is used to train the LSTM networks. Then the validation set is used to validate the performance of the networks in each epoch. Finally the prediction set is used to validate the performance of the final networks in attacking a new set of traces. For the combined traces and combined LSTM schemes, we also apply LDA to the traces, since dimensionality reduction may be necessary for longer traces, which is common in more complex circuits. The configuration of the networks we use in the experiments are described in Section 7.3.4.

7.3.1 Similar Level of Leakage

In this scenario, we investigate based on the presumption that static and dynamic power leakage have similar level of information contained. The dynamic traces are taken using a sampling rate of 20 GS/s. The static traces are taken by picking 25 measurement points in each half clock period (clock high and clock low). Subsequently, we add Gaussian noise with 0 mean and 0.007 A standard deviation to both the static and the dynamic traces. Then the 25 static power measurements are averaged to filter out much of the noise.

Table 7.1 shows the training results of all the different methods. Three different metrics are demonstrated in this table: *time per epoch* refers to the average time used to train the networks in each epoch; *loss* refers to the loss function we introduced in Section 2.6.1; and *accuracy* is the percentage of accurate predictions of the key used in each single trace. In each metric, the scheme that performs the best is marked by bold font. It is easy to identify that for loss and accuracy, all the three proposed combined schemes outperform the cases where solely static or dynamic traces are used. Due to the reason that the input layers contain more dimensions when combining static power and dynamic power traces, more time will be needed to train the networks. The performance of the three combined schemes are close, while the hierarchical LSTM scheme manages to get the best accuracy with the least amount of training time.

While Table 7.1 demonstrates the accuracy of predicting the key used in a single trace, it does not directly relate to the success rate metric, which involves the prediction result of multiple traces. In this experiment, we generate 16 target sets by randomly picking M_t traces from the validation set for each possible key, and use the trained networks to guess which key this set uses. Then we repeat this process 10 times, that is, for each M_t there are 160 attacks performed to get the empirical success rate. The

Table 7.1: Training results with similar level of leakage

	Time per epoch (s)	Loss	Accuracy
Sta.	16	2.3712	0.2055
Sta. w/LDA	14	2.3744	0.2036
Dyn.	25	1.9467	0.3208
Dyn. w/LDA	26	1.9931	0.3080
CT	45	1.5983	0.4375
CT w/LDA	28	1.6453	0.4301
cLSTM	40	1.5880	0.4462
cLSTM w/LDA	39	1.6454	0.4274
hLSTM	**23**	**1.5657**	**0.4517**

number of target traces, M_t, is varied from 1 to 200.

The success rates of the attacks are shown in Figure 7.4. Similar to Table 7.1, we observe that the proposed schemes all get significantly better success rates than the attacks that are solely using one leakage source. While the difference between the three combined schemes is small, we do observe that the hierarchical LSTM gets better performance than the other two schemes, and the combined trace scheme performs worst of the three.

7.3.2 More Informative Static Leakage

In this scenario we change the sampling rate of the dynamic traces to 10 GS/s. The other configurations remain the same as that used in the previous scenario. By decreasing the sampling rate we are making the dynamic leakage less informative compared to the static leakage.

The training results shown in Table 7.2 are evaluated using the same metrics as the previous case. For the cases with sole leakage sources, static power leads to significantly better accuracy than dynamic power. When combining the two leakage sources together using our proposed scheme, we find that the hierarchical LSTM outperforms the others in all metrics, but the difference is negligible compared with

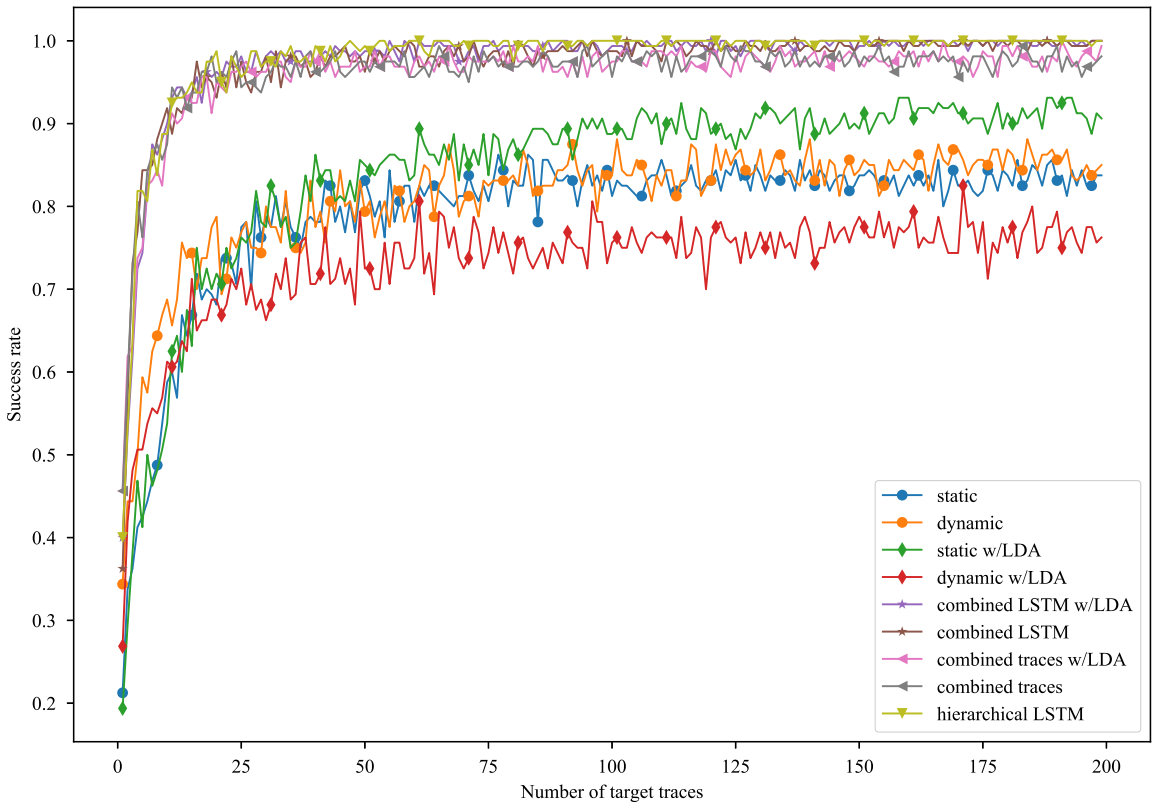


Figure 7.4: Success rates of LSTM-based attacks in a scenario that has similar level of static and dynamic power leakage.

Table 7.1. The only significant advantage is that the training time for the hierarchical LSTM is still much smaller than the other schemes. The performance of the combined schemes is only slightly better than the static power case. Due to the reason that little information can be gained from the dynamic traces, combining the two leakage sources does not significantly improve the accuracy of prediction compared with using only static leakage.

The success rates of attacks using these training results are shown in Figure 7.5, generated using the same method as in Section 7.3.1. The success rate traces are clearly divided into two groups: the less informative dynamic part and the more informative static part together with the combined schemes. The combined LSTM and hierarchical LSTM get slightly better performance than the static power leakage, while

Table 7.2: Training results with more informative static leakage

	Time per epoch (s)	Loss	Accuracy
Sta.	16	2.3789	0.2002
Sta. w/LDA	15	2.3731	0.2035
Dyn.	15	2.7478	0.0876
Dyn. w/LDA	15	2.7464	0.0877
CT	27	2.3724	0.2012
CT w/LDA	25	2.3700	0.2044
cLSTM	27	2.3658	0.2017
cLSTM w/LDA	27	2.3668	0.2058
hLSTM	**19**	**2.3656**	**0.2065**

the results for combined traces is unstable – achieving both the best and the worst in the proposed schemes. A significant conclusion to be drawn from the numerical results is that, with the hierarchical LSTM scheme we are able to achieve a success rate that is as good as the performance of an approach using the static power leakage, when the dynamic part is significantly less informative.

7.3.3 More Informative Dynamic Leakage

In this scenario we use the sampling rate of 50 GS/s for the dynamic traces, and only 2 measurement points are used to average out the noise in the static traces. The same additive Gaussian white noise with 0.007 A standard deviation is also added to both traces. The dynamic power leakage is more informative and static power leakage is less informative compared to the first scenario.

The training results are shown in Table 7.3. Similar to the other scenarios, the hierarchical LSTM gets the best validation results using the least amount of training time. The combined traces without LDA overfits the training set, though it gets slightly better training results. Since there is little information in the static power leakage, the proposed combined schemes achieve performances close to the dynamic-power-based attacks.

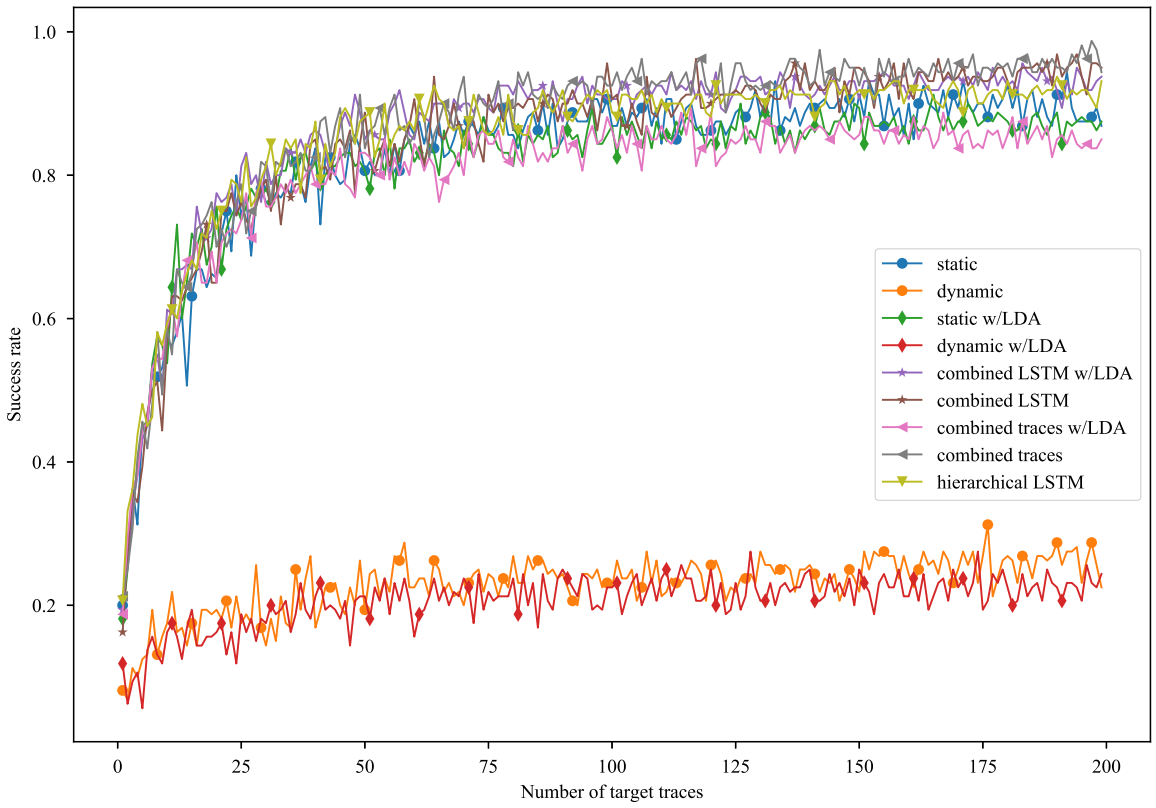


Figure 7.5: Success rates of LSTM-based attacks in a scenario that the static power leakage is more informative than the dynamic power leakage.

The success rate results are shown in Figure 7.6. We can observe that the combined schemes achieve similar success rates as the dynamic power attacks. Combined LSTM without LDA performs worst of all combined schemes, and even worse than the dynamic power with LDA attack. The hierarchical LSTM performs slightly better than the other schemes and achieves the best performance of all attacks.

Similar to the previous scenario, we can draw the conclusion that when the static power leakage is significantly less informative, the hierarchical LSTM scheme is able to guarantee performances which are close to the dynamic-power-based attacks. This conclusion is verified by both the machine learning metrics and the success rates.

Table 7.3: Training results with more informative dynamic leakage

	Time per epoch (s)	Loss	Accuracy
Sta.	16	2.7430	0.0888
Sta. w/LDA	15	2.7458	0.0888
Dyn.	49	1.6616	0.4054
Dyn. w/LDA	25	1.6476	0.4034
CT	73	1.6122	0.4241
CT w/LDA	30	1.6147	0.4188
cLSTM	63	1.6467	0.4087
cLSTM w/LDA	41	1.6214	0.4092
hLSTM	**24**	**1.5435**	**0.4339**

7.3.4 Attack Configuration

The proposed LSTM network methods are implemented using the Keras [85] library with Theano [86] backend, and executed on a desktop computer equipped with an Intel Core i7-4790 CPU, an NVIDIA GeForce GTX 745 GPU and 16 GB of RAM.

Since the dimensionality of the target data in our scenario is not huge – hundreds in total – the required size of the LSTM network does not have to be extremely large. We are able to iterate through the possible parameters for the network size and choose the relatively optimal one, which should be able to train fast and not overfit in the given number of epochs.

We list the parameters we use in Table 7.4. For all the attacks we use a fixed number of 50 epochs for training, and the batch size used in each iteration is set to 32.

Even in a realistic attack using a more complex data set, it is still possible to get some outcome using a relatively small LSTM network. For example, the data set from DPA Contest V2 [34] is used by [25], in which traces with thousands of dimensions are used to attack an AES roundkey byte, and the authors manage to demonstrate some successful attack results using a 2-layer LSTM with 26 units in each hidden layer.

If indeed a large LSTM network is needed and it is hard to determine the structure parameters for the network, a *genetic algorithm* [87] can be employed to find the

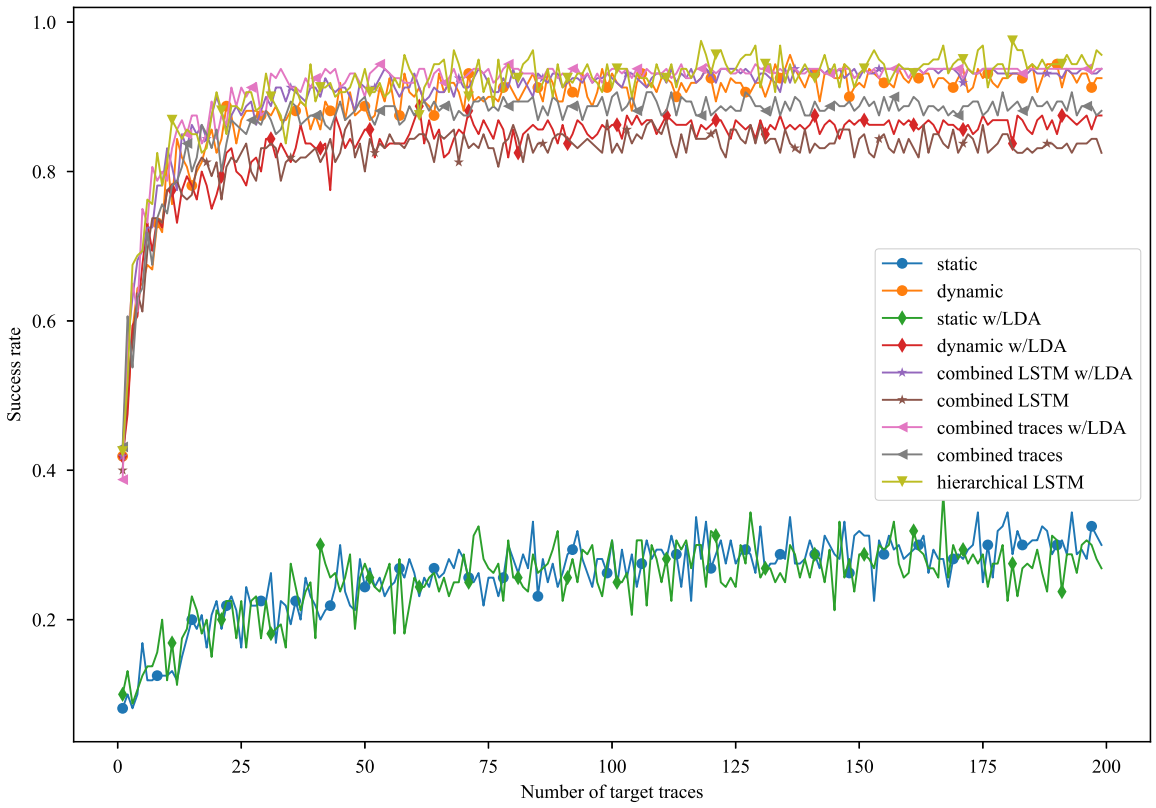


Figure 7.6: Success rates of LSTM-based attacks in a scenario that the dynamic power leakage is more informative than the static power leakage.

optimal parameters.

7.4 Conclusion

In this chapter, we propose three different schemes that combine the static power leakage and dynamic power leakage by means of an LSTM network. The performance of these schemes are numerically evaluated via the traces acquired from a simulated masked S-box circuit. We are able to demonstrate that when the static part and the dynamic part have similar level of leakage information, the proposed combined schemes are able to get a better result compared to using only one leakage source. When one leakage source is significantly less informative, a proposed combined scheme

Table 7.4: The parameters used to train the networks

			Similar	More Static	More Dynamic
Sta.		LSTM	20	20	20
		LSTM	20	20	20
		Dropout	0.2	0.2	0.2
Dyn.		LSTM	28	8	28
		LSTM	28	8	28
		Dropout	0.2	0.2	0.2
CT		LSTM	48	28	48
		LSTM	48	28	48
		Dropout	0.2	0.2	0.2
cLSTM	Sta.	LSTM	20	20	20
		LSTM	20	20	20
		Dropout	0.2	0.2	0.2
	Dyn.	LSTM	28	8	28
		LSTM	28	8	28
		Dropout	0.2	0.2	0.2
hLSTM		Encode LSTM	40	24	40
		LSTM	40	20	40
		Dropout	0.2	0.1	0.2
		LSTM	40	20	40
		Dropout	0.2	0.1	0.2

can still guarantee that the combined performance is as good as the performance of an attack targeting only the more informative leakage.

Generally the hierarchical LSTM performs better than the other two judging from machine learning metrics. This method trains the network quickly, but it requires extra preprocessing to format the data for different clock cycles. Also, we notice that the single-trace prediction accuracy is not exactly the same as the final success rate, hence we observe some randomness in the success rate results. We recommend the hierarchical LSTM scheme, since it is the best or close to the best in all cases. In order to facilitate our comparisons we try to keep the structure of the networks simple, but some additional optimizations are possible. For example, more dropout layers can be added, and we can add extra LSTM layers after the concatenation layer in the combined LSTM scheme, to get a combination of the first two schemes.

The proposed schemes are verified using a simple simulated cryptographic circuit

to explore the benefit of *combining* the two leakage sources. Using a simple circuit ensures the attacks with single leakage source can be easily implemented, hence we are able to focus on the schemes used to combine the leakage sources. Using a simulated circuit also brings us the convenience of easily changing the parameters, in order to demonstrate the performance of the proposed schemes in different scenarios. For future work, verifying with real-world cryptographic circuits will be considered.

Finally, we have demonstrated how to combine the static power and dynamic power leakage in this work, but the possible leakage sources to be considered should not be limited to these. With the help of deep learning techniques, we might able to choose from a wider range of leakage sources. For example, electromagnetic leakage [29] and domain knowledge [27]. We hope that our research will inspire more related works.

Chapter 8

Conclusion

8.1 Summary and Contribution

The research of this thesis targets static-power-based profiled attacks. This research starts with examining the behavior of static power leakage, then some improvements are proposed to improve the performance of template attacks when attacking using static power. The performance of static power and dynamic power are compared, then a method is proposed to combine static and dynamic power, in order to achieve better performance.

Part of the initial motivation of this research comes from the prediction made by ITRS that the static power consumption can be significant in newer processing technologies, especially in sub-90nm technologies. Driven by this, we have performed a few evaluations of the overall static power consumption of some circuit components.

The early period of our research is influenced by papers such as [13, 38], in which a specific circuit structure is targeted. In our research we tried to analyze the static power leakage of circuits employing a given circuit structure, such as the bit-sliced structure suggested in [13, 38]. However we find this method limiting when dealing with synthesized circuits – if the design structure of target circuit is a black box then it is hard to determine the structure affecting the leakage. Besides, the synthesized circuits are more random compared to hand-wired bit-sliced structure, which makes

them harder to model and the bit-sliced structure less realistic in our scenario. Hence instead of a specific structure, we decided to use generic statistical methods to model the behavior of circuits, which is the motivation for the major part of our research being focused on profiled attacks.

Here we summarize the contributions throughout the author's PhD program.

In the early stage of research, the leakage model proposed by [13] was verified and improved, taking both input and output into consideration. Then the focus of research shifted to applications of profiled attacks using static power. When applying conventional template attacks to static power, J-divergence and JS distance were proposed to replace the original Gaussian likelihood, and it was numerically shown that the proposed replacement reduces the elapsed time of attacks without sacrificing the effectiveness of the attacks.

Later, motivated by the discovery that the distribution of static power traces may not satisfy multivariate Gaussian distribution, we were the first in the literature to propose to use kernel density estimation in static-power-based attacks. We demonstrated with simulated circuits that when the static power traces are not Gaussian, the kernel method outperforms the conventional Gaussian distinguisher. Since there are less POIs in a static power trace than in a dynamic power trace, the combined distribution is more likely to be non-Gaussian in the static power scenario.

Then we carried out template attacks against protected cryptographic circuits. A masked PRESENT S-box was attacked using both static power and dynamic power leakage. Multiple conclusions can be drawn from these attacks. Firstly we prove that the threshold implementation masking scheme is not secure against profiled attacks, and we are able to demonstrate different advantages of using dynamic and static power as the leakage source.

Motivated by the previous outcome we started exploring the possibility of combining

static-power-based attacks and dynamic-power-based attacks in order to ensure better performance in both scenarios. With the help of the LSTM network we propose three different schemes to combine static and dynamic power leakage. All these schemes can ensure that the performance of attacking using the combined traces is at least as good as solely using the better one in the two leakage sources. In our experiments we find that the hierarchical LSTM proposal outperforms the other two in different scenarios. We are the first in literature to propose to combine dynamic and static power leakage by means of deep learning techniques.

8.2 Future Work

Due to the limitation in time and equipment, some parts of our work were not extended as far as possible. Here we list some suggested future work.

Up till now, all the experiments are carried out using simulated circuits. Due to the reason that SPICE-level simulations are time consuming, we are not able to investigate the static power leakage of more complex and completely-functional cryptographic circuits. Carrying out the proposed attacks using real-world chips will greatly improve the value of our proposal.

Using simulated circuits does benefit our research in the sense that we can easily control the amount of environmental noise. A potential future work would be investigating the influence of noise on the proposed attacks. The outcome of this could be some generic modeling of the success rate or a guidance on how much noise has to be added in order to defend against static-power-based attacks.

Our work in combining the static and dynamic power leakage can be further extended. Applying deep learning to profiled attacks does provide us with many possibilities. In our work only LSTM is used, while other deep learning techniques

can also be applied to similar scenarios, such as convolutional neural network and attention network. In our research only static and dynamic power are considered, both taking the format of power traces. However, with the help of deep learning we are capable of dealing with more complex data, hence different formats of leakage sources can all be considered for combination, such as timing, EM leakage and heat.

Bibliography

[1] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1998. 1

[2] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology - CRYPTO 96*, ser. Lecture Notes in Computer Science, N. Koblitz, Ed. Springer Berlin Heidelberg, 1996, vol. 1109, pp. 104–113. 2, 3

[3] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99. London, UK, UK: Springer-Verlag, 1999, pp. 388–397. 2, 3

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, ser. Lecture Notes in Computer Science, M. Joye and J.-J. Quisquater, Eds. Springer Berlin Heidelberg, 2004, vol. 3156, pp. 16–29. 2, 3

[5] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 426–442. 2, 3, 18

[6] M. Hutter and J.-M. Schmidt, "The temperature side channel and heating fault attacks," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013, pp. 219–235. 2

[7] J. Longo, E. De Mulder, D. Page, and M. Tunstall, "Soc it to em: Electromagnetic side-channel attacks on a complex system-on-chip," in *Cryptographic Hardware*

and Embedded Systems – CHES 2015, T. Güneysu and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 620–640. 2

[8] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic side-channel attacks on printers," in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 20–20. 2

[9] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full AES," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 344–371. 2

[10] C. Clavier, J.-L. Danger, G. Duc, M. A. Elaabid, B. Gérard, S. Guilley, A. Heuser, M. Kasper, Y. Li, V. Lomné *et al.*, "Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest," *Journal of Cryptographic Engineering*, vol. 4, no. 4, pp. 259–274, 2014. 2, 18

[11] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2002, pp. 13–28. 3, 18, 21, 84, 86, 102

[12] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated, 2007. 3, 12, 13

[13] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti, "Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 2, pp. 355–367, Feb 2010. 3, 4, 14, 18, 38, 52, 56, 58, 63, 77, 117, 128, 129

[14] M. Alioto, S. Bongiovanni, M. Djukanovic, G. Scotti, and A. Trifiletti, "Effectiveness of leakage power analysis attacks on DPA-resistant logic styles under process variations," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, no. 2, pp. 429–442, Feb 2014. 3, 4, 18, 38, 117

[15] L. Lin and W. Burleson, "Leakage-based differential power analysis (LDPA) on sub-90nm cmos cryptosystems," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 252–255. 4, 18

[16] A. Moradi, "Side-channel leakage through static power," in *Cryptographic Hardware and Embedded Systems - CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds. Springer Berlin Heidelberg, 2014, vol. 8731, pp. 562–579. 4, 18, 53

[17] S. M. Del Pozo, F.-X. Standaert, D. Kamel, and A. Moradi, "Side-channel attacks from static power: when should we care?" in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 145–150. 4, 14, 51, 53, 86, 95, 102, 105, 107, 117

[18] D. Bellizia, M. Djukanovic, G. Scotti, and A. Trifiletti, "Template attacks exploiting static power and application to cmos lightweight crypto-hardware," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 229–241, 2017. 4, 38, 102, 117

[19] T. Bartkewitz and K. Lemke-Rust, "Efficient template attacks based on probabilistic multi-class support vector machines," in *Smart Card Research and Advanced Applications*, S. Mangard, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 263–276. 4

[20] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, p. 293, Oct 2011. [Online]. Available: https://doi.org/10.1007/s13389-011-0023-x 4

[21] L. Lerman, G. Bontempi, and O. Markowitch, "Side channel attack: An approach based on machine learning," in *in Constructive Side-Channel Analysis and Secure Design*. Springer, 2011, pp. 29–41. 4

[22] H. Patel and R. O. Baldwin, "Random forest profiling attack on advanced encryption standard," *Int. J. Appl. Cryptol.*, vol. 3, no. 2, pp. 181–194, Jun. 2014. 4

[23] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, Jun 2015. 4

[24] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert, "Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)," in *Constructive Side-Channel Analysis and Secure Design*, S. Mangard and A. Y. Poschmann, Eds. Cham: Springer International Publishing, 2015, pp. 20–33. 4

[25] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering*, C. Carlet, M. A. Hasan, and V. Saraswat, Eds. Cham: Springer International Publishing, 2016, pp. 3–26. 4, 34, 124

[26] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but don't touch me! contactless control flow monitoring via electromagnetic

emanations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1095–1108. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134081 5, 34

[27] B. Hettwer, S. Gehrer, and T. Güneysu, "Profiled power analysis attacks using convolutional neural networks with domain knowledge," in *2018 Selected Areas in Cryptography (SAC)*, August 2018. 5, 127

[28] D. Agrawal, J. R. Rao, and P. Rohatgi, "Multi-channel attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 2–16. 5, 113

[29] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 411–425. 5, 113, 127

[30] D. Stinson, "Cryptography: Theory and practice," 2002. 8, 9

[31] A. Kahate, *Cryptography and network security*. Tata McGraw-Hill Education, 2013. 8

[32] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949. 9

[33] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher,"

in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 450–466. 10, 38, 68, 116

[34] DPA contest v2. [Online]. Available: https://www.dpacontest.org/v2/ x, 16, 17, 124

[35] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 109–128. 16

[36] S. Tiran, S. Ordas, Y. Teglia, M. Agoyan, and P. Maurine, "A model of the leakage in the frequency domain and its application to CPA and DPA," *Journal of Cryptographic Engineering*, vol. 4, no. 3, pp. 197–212, 2014. 18

[37] N. Debande, Y. Souissi, M. Aabid, S. Guilley, and J.-L. Danger, "Wavelet transform based pre-processing for side channel analysis," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture Workshops*. IEEE Computer Society, 2012, pp. 32–38. 18

[38] M. Alioto, S. Bongiovanni, G. Scotti, and A. Trifiletti, "Leakage power analysis attacks against a bit slice implementation of the serpent block cipher," in *2014 Proceedings of the 21st International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, June 2014, pp. 241–246. 18, 128

[39] J. Xu and H. M. Heys, "Template attacks based on static power analysis of block ciphers in 45-nm cmos environment," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2017, pp. 1256–1259. 20, 51, 75, 86, 89

[40] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2008, pp. 411–425. 20, 21

[41] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013, pp. 253–270. 20, 21, 22, 28, 89, 105

[42] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 1–14. 21, 27, 28, 105

[43] T. Schneider, A. Moradi, F.-X. Standaert, and T. Güneysu, "Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms," in *Selected Areas in Cryptography – SAC 2016*, R. Avanzi and H. Heys, Eds. Cham: Springer International Publishing, 2017, pp. 58–78. 21

[44] C. Meyer, "On the rank of the sum of two rectangular matrices," *Canad. Math. Bull*, vol. 12, p. 508, 1969. 22

[45] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951. 23

[46] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information theory*, 2003. 23, 26

[47] K. T. Abou-Moustafa and F. P. Ferrie, "A note on metric properties for some divergence measures: The gaussian case." in *ACML*, 2012, pp. 1–15. 25

[48] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 4. IEEE, 2007, pp. IV–317. 25, 36

[49] J. Duchi, "Derivations for linear algebra and optimization," *Berkeley, California*, 2007. 25

[50] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991. 26

[51] D. Johnson and S. Sinanovic, "Symmetrizing the kullback-leibler distance," *IEEE Transactions on Information Theory*, 2001. 26

[52] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x 27, 28

[53] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 411–425. 28

[54] T. Eisenbarth, C. Paar, and B. Weghenkel, *Building a Side Channel Based Disassembler*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 78–99. [Online]. Available: https://doi.org/10.1007/978-3-642-17499-5_4 28

[55] D. W. Scott and S. R. Sain, "9 - multidimensional density estimation," in *Data Mining and Data Visualization*, ser. Handbook of Statistics, C. Rao, E. Wegman, and J. Solka, Eds. Elsevier, 2005, vol. 24, no. Supplement C, pp. 229 – 261. 29, 30, 35

[56] T. Schneider, A. Moradi, F.-X. Standaert, and T. Güneysu, *Bridging the Gap: Advanced Tools for Side-Channel Leakage Estimation Beyond Gaussian Templates and Histograms*. Springer International Publishing, 2017, pp. 58–78. 30

[57] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015. 30

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735 31

[59] M. Hermans and B. Schrauwen, "Training and analyzing deep recurrent neural networks," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 190–198. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999611.2999633 31

[60] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994. [Online]. Available: http://dx.doi.org/10.1109/72.279181 32

[61] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. 32, 33

[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html 34

[63] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 03 1951. 35

[64] S. Mangard, *Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 222–235. xii, 36, 109

[65] F. Durvaux, F.-X. Standaert, and N. Veyrat-Charvillon, *How to Certify the Leakage of a Chip?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 459–476. 36

[66] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2009, pp. 443–461. 36, 37, 77, 106

[67] M. Rivain, *On the Exact Success Rate of Side Channel Analysis in the Gaussian Model.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 165–183. 37

[68] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, "Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents," in *International Conference on Smart Card Research and Advanced Applications.* Springer, 2008, pp. 89–103. 44, 71, 73

[69] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling, "Side-channel resistant crypto for less than 2,300 ge," *Journal of Cryptology*, vol. 24, no. 2, pp. 322–345, Apr 2011. 46, 103, 110

[70] J. Xu and H. M. Heys, "Introduction to static power analysis of cryptographic devices," in *Newfoundland Electrical and Computer Engineering Conference (NECEC 2015)*, St. John's, Newfoundland and Labrador, Nov. 2015. 51

[71] Cadence AMS kit. [Online]. Available: https://www.cmc.ca/en/WhatWeOffer/Products/CMC-00200-01509.aspx 53, 99

[72] FreePDK. [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK 53, 80, 116

[73] CD4013BC datasheet. [Online]. Available: https://www.fairchildsemi.com/products/legacy/CD4013BC.html 57

[74] L. Xiao and H. M. Heys, "Hardware design and analysis of block cipher components," in *Information Security and Cryptology - ICISC 2002*. Springer, 2002, pp. 164–181. 60

[75] S. Banik, A. Bogdanov, and F. Regazzoni, "Exploring energy efficiency of lightweight block ciphers," in *International Conference on Selected Areas in Cryptography*. Springer, 2015, pp. 178–194. 60

[76] D. J. Biau, B. M. Jolles, and R. Porcher, "P value and the theory of hypothesis testing: an explanation for new researchers," *Clinical Orthopaedics and Related Research®*, vol. 468, no. 3, pp. 885–892, 2010. 61

[77] N. Hanley, M. Tunstall, and W. P. Marnane, "Unknown plaintext template attacks," in *Information Security Applications*. Springer, 2009, pp. 148–162. 78

[78] J. Xu and H. M. Heys, "Kernel-Based Template Attacks of Cryptographic Circuits Using Static Power," *Under review.* 80

[79] F. Durvaux, F.-X. Standaert, and S. M. Del Pozo, *Towards Easy Leakage Certification.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 40–60. 87

[80] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr 1997. 98

[81] J. Xu and H. M. Heys, "Template attacks of a masked s-box circuit: A comparison between static and dynamic power analyses," in *2018 IEEE 16th International New Circuits and Systems Conference (NEWCAS)*, July 2018. 103

[82] T. Moos, A. Moradi, and B. Richter, "Static power side-channel analysis of a threshold implementation prototype chip," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1324–1329. 105, 107, 109, 117

[83] J. Xu and H. M. Heys, "Using Deep Learning to Combine Static and Dynamic Power Analyses of Cryptographic Circuits," *Under review.* 112

[84] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *CoRR*, vol. abs/1609.01704, 2016. [Online]. Available: http://arxiv.org/abs/1609.01704 115

[85] F. Chollet *et al.*, "Keras," https://keras.io, 2015. 124

[86] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688 124

[87] J. Shapiro, *Genetic Algorithms in Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 146–168. [Online]. Available: https://doi.org/10.1007/3-540-44673-7_7 124