2016

Secure hardware design against side-channel attacks

Jungmin Park
Iowa State University

Secure hardware design against side-channel attacks

by

Jungmin Park

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:

Akhilesh Tyagi, Major Professor

Phillip Harrison Jones

Arun K. Somani

Diane T. Rover

Soma Chaudhuri

Iowa State University

Ames, Iowa

2016

DEDICATION

I would like to dedicate this thesis to my wife Mihyun and to my daughter Clare and to my son Kevin and Kaden without whose support I would not have been able to complete this work. I would also like to thank my friends and family for their loving guidance and financial assistance during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Akhilesh Tyagi for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education.

ABSTRACT

Embedded systems such as smart card or IoT devices should be protected from side-channel analysis (SCA) attacks. For the secure hardware implementation, SCA security metrics to quantify robustness of the implementation at the abstraction level from the logic level to the layout level against SCA attacks should be considered. In our design flow, the first security test is executed at the logic level. If the implementation does not satisfy the threshold of the SCA security metric based on Kullback-Leibler divergence, the module can be re-synthesized with secure logic styles such as WDDL or t-private logic circuits. At the final security test, we use the machine learning technique such as LDA, QDA, SVM and naive Bayes to check the distinguishability of the side-channel leakage depending on inputs or outputs. These techniques apply to an ASIC in characterizing the secret data leakage.

In this thesis, t-private logic circuits are implemented with the FreePDK45nm. The SCA security metric as well as the delay and power consumption is characterized. All this characterization data are stored in the standard liberty format(.lib) in order for general CAD tools to use this file. The t-private logic package including the general digital logics can be exploited for secure VLSI design. Also, various classifiers such as LDA, QDA, SVM or naive Bayes are used to emulate real SCA environment. Based on this SCA simulator, the threshold of the SCA security metric can be estimated and the security can be verified more accurately. The secure logic cell package and SCA simulator support the methodology of the secure hardware implementation.

CHAPTER 1. INTRODUCTION

Most of modern electrical devices are connected through the Internet. Private information and secret data go back and forth between devices and servers. If significant and secret information such as usernames, passwords and credit cards is controlled freely by adversaries, the vast monetary demage is caused. Information security has been an extensive issue of many IT field. For the secure communications and protection of information, cryptography has played a significant role and modern cryptography such as AES and RSA cannot be broken theoretically. In disregard of contribution of the modern cryptography, electrical devices can leak information through side-channels or physical channels that are unintended. Common side-channel attacks use power/current at V_{dd} pin [Kocher et al. (1999)] or electromagnetic radiation [Quisquater and Samyde (2001)] to reveal a secret key. The power side-channel attacks are based on the fact that the power consumption depends on the intermediate values which are correlated to both some controlled inputs and some secret data embedded in the crypto-block.

Differential power analysis (DPA) of side-channel attacks has been shown to be especially effective in finding the secret key by exploiting correlation between the power consumption and the processed data [Kocher et al. (1999)]. Since this attack needs little knowledge of the implementation of the cryptographic algorithm and can be performed with relatively cheap equipment, it poses a major threat to cryptographic devices such as smart cards or embedded systems. The hypothetical power consumption model (or leakage model) of an adversary based on the intermediate value which depends on the key is related to the measured power consumption if the key guess is correct. The attack can succeed or fail based on the selected leakage model. Assuming that power consumption depends solely on the number of switched bits or the number of 1's in the intermediate value, Hamming distance model or Hamming weight model is chosen respectively [Alioto et al. (2010), Mangard et al. (2007), Messerges et al. (2002)]. How-

ever, in a real circuit based on ASICs, the assumption that power is a function of Hamming distance or Hamming weight derived from the secret may not hold. Instead, model profiles (or templetes) may be a better choice at a higher cost for more complex modeling effort. An even more powerful adversary is the Bayesian side-channel adversary using the template that selects the key guess which maximizes the probability that key guess is correct given the leakage probability density ($argmax_{k^*}\mathbf{Pr}[k^*|l]$) [Standaert et al. (2009)]. Side-channel attacks of the Bayesian side-channel adversary should also be considered as major threats.

Many research efforts have targeted techniques to prevent side-channel attacks. The countermeasures of DPA attacks are categorized into two groups: *hiding* and *masking* [Mangard et al. (2007)]. The hiding countermeasures make the power consumption of cryptographic devices independent of the intermediate values by making the power consumption random or uniformly same for all data values. The masking countermeasures achieve the independence of power consumption from the intermediate values by randomizing the intermediate values. This also masks the logic behavior. But earlier countermeasures have been suitable for only specific hardware implementation. For example, the method to randomize logic behavior should be changed according to different hardware constraints such as the critical path, timing or power consumption. The ad hoc approach causes the productivity of the secure design to be low. Also, we do not know how much these countermeasures enforce DPA security. To the best of our knowledge, no CAD tools that integrate such a DPA resistance computation and suggest an appropriate hiding or masking countermeasure to improve a vulnerable design seem to exist. New paradigm should be needed to satisfy both productivity and security.

1.1 Contribution

There are four main threads for the unified secure design methodology. **First**, security against SCA attacks is included as a constrained resource along with delay and area for the secure hardware implementation of the cryptographic system. The SCA security is quantified using **(1)** the normalized variance metric (or the coefficient of variance) [Basel Halak (2013)], **(2)** Kullback-Leibler divergence and [S. Kullback and R. A. Leibler (1951)] **(3)** the information theoretic metric of the profiled power distribution [Mac et al. (2007)]. In our design flow, SCA

vulnerability should be verified with these metrics at all implementation abstraction levels from logic (or gate) to layout level. We estimate Kullback-Leibler divergence from the power distribution gathered from the approximate and quick renewel process based logic level simulation. The KL divergence is very related to vulnerability aginst side-channel attacks.

Once the SCA metric at the higher logic abstraction level is within safe bounds, the design flow can enter the next abstraction level refinement. This abstraction refinement (as in logic level to netlist level) introduces details that may develop new SCA vulnerabilities. Hence an acceptable SCA metric value at higher abstraction layers still necessitates SCA metric computation at lower levels. The mutual information metric is computed at the layout level with multiple SPICE level circuit simulations. The acceptable thresholds for SCA security metric are defined theoretically. If any combinational module has a value larger than the threshold, it is flagged as a vulnerable module. Such a hierarchical filter not only results in more efficient assessment of SCA vulnerabilities, the countermeasures can also be of variable granularity to match the abstraction level (logic or netlist). Arguably, the corrective steps taken at logic level are more effective even though the accuracy of the metric at that level is lower.

The logic level filter uses the classical switching probability computation to estimate power which depends on the secret data (key) or a correlated intermediate result. Even though simulation based verification can be performed at the logic level, the probabilistic estimation method for power is more efficient. Statistical Monte Carlo power estimation techniques [Najm (1994)] are better suited than the BDD based power estimators [Sentovich et al. (1992), Monteiro et al. (1997)] due to the need for model parametrization with secret key. The statistical power estimation model is based on the fact that power consumption depends on the transition probability and capacitance of the output node of logic gates [Najm (1994)]. Since the transition probability of the output node is influenced by input transition patterns, it can be modeled as a normal distribution. The mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ can be estimated through sampling a large enough space of the input patterns and computing power over that input pattern. The more distinguishable and identifiable power consumption is according to different inputs, the more vulnerable is the SCA security. The SCA security metric can be computed as $\hat{\sigma}/\hat{\mu}$. This

analytical method can be applied to combinational circuits. Note that the SCA security metric for multiple implementations of the same behavior can vary even though the logic level boolean equations specifying the arithmetic function are the same. The normalized variance metric can be used to compare SCA vulnerability of multiple implementations but it does not provide a safety threshold to flag a vulnerable implementation. Instead, SCA metric using KL-divergence divergence plays a critical role to distinguish vulnerable implementations. If the SCA security metric of any computing block has a large value or is above a threshold, it should be reduced significantly, possibly to zero, by the proposed resynthesis at the logic level.

Once the logic level design has an acceptable variance metric, It can be synthesized into transistor level netlist. The information theoretic metric of mutual information can be computed both at the transistor netlist level and physical (or layout) level. Mutual information ($I(K;L)$) of the secret data (K) and the corresponding leakage (L) as the third SCA security metric quantifies amount of information about the secret data in the leakage channel (power). If the mutual information indicates that a significant fraction of n secret key bits are leaking through L (power), the design needs to be reinforced.

The **second** thread consists of a design schema to reduce the SCA vulnerability at the netlist level. This is done through the so called technology mapping or cell binding phase. SCA secure versions of the t-private [Ishai et al. (2003)] cells as well as the sense amplified based logic (SABL) and wave differential dynamic logic (WDDL) for AND, OR, NAND, NOR, NXOR and XOR logic gates are to be provided in the technology library. These t-private cell primitives are based on Ishai's t-private circuits which are robust against the t-th order side-channel (or probing) attacks [Ishai et al. (2003)]. They can be verified as SCA secure using our SCA security metrics at all design abstraction levels. The parts of the cryptographic system determined vulnerable by the KL divergence based SCA security metric at the logic level can be synthesized with these t-private cells, SABL or WDDL cells.

Third, a t-private logic synthesis method is proposed in order to prevent side-channel attacks at the logic (or gate) level. After logic synthesis, vulnerable sub-logic can be determined through SCA security metrics. It should be synthesized into the following reduced area version of t-private circuits. The boolean functions of insecure parts are represented by the exclusive-

Table 1.1: Proposed Security Metrics and Solution at each Design Abstraction level

	security metrics	leakage estimation method	solution
logic level	KL divergence	renewal process	t-private logic synthesis
transistor level	all	simulation	balance matching
physical(layout) level	all	simulation	balance matching

OR sum-of-products (ESOP) and then the products are masked with random bits. The masked products are replaced with t-private circuits. Exclusive-ORs are also replaced with t-private XOR circuits. We call this t-private logic synthesis. Since t-private XOR and NXOR primitives have significantly smaller area and better delay than the original t-private circuits, the ESOP representation may have both area and delay advantages. Table 1.1 summarizes the proposed security metrics and side-channel leakage estimation methods.

Finally, the **fourth** thread targets secure memory modules. Memories also leak information. Private data including cryptographic keys are committed to the memory. This data-at-rest is open to physical access based attacks. These attacks slice the silicon until individual transistors are exposed by a Focused Ion Beam (FIB). An electron microscope is used to examine the silicon. Halderman et al. [Halderman et al. (2008)] proposed "cold-boot attack" which is a method to measure a significant fraction of data stored in a powered-off memory (e.g. DRAM) by cooling the chip to around $-50°C$ at which temperature the data will persist for several minutes with minimal error. Ishai's [Ishai et al. (2003)] t-private coding can be used for memory as well. Recently, Valamehr et al. [Valamehr et al. (2012)] developed more general and more efficient masking methods to prevent such memory attacks. However, their more efficient memory coding methods require the private data-at-rest such as a key to be decoded before it can be used in computation. We propose coding methods that are as efficient as Valamehr et al. [Valamehr et al. (2012)] for memory coding, but at the same time can use the encoded data-at-rest for computing in flight as is. We call such coding systems t-private systems.

1.2 Summary

The thesis is orgaized as the following chapters. Chapter 1 gives an introduction to the background and contribution of the thesis.

Chapter 2 presents the overview of side-channel analysis attacks. As an example, side-channel based AVR diassembler is proposed.

Security metrics are proposed in Chapter 3. This chapter is based on the pulished papers in VLSID 2016 [Park and Tyagi (2016)] and ISVLSI 2014 [Park and Tyagi (2014b)].

Chapter 4 presents secure logic styles such as t-private logic circuits, SABL and WDDL. These secure logic cells are implemeted at the various abstract level (from the logic gate level to the layout level). Also, the SCA vulnerablility of these secure logic style is verified by simulating SCA attacks.

Chapter 5 presents the methodology of SCA secure FPGA and ASIC implementation. This chapter is based on the published paper in HOST 2012 [Park and Tyagi (2012)]

Chapter 6 presents t-private memory and systems. Probing-resistant memories are focuced on. This chapter is based on the published paper in SPACE 2014 [Park and Tyagi (2014a)]

In the final chapter 7, the thesis is concluded with a discussion on future work.

CHAPTER 2. SIDE-CHANNEL ANALYSIS ATTACKS

2.1 Introduction

Common side-channel analysis attacks use a current path at V_{dd} or *gnd* pin or electromagnetic radiation of a specific location in the chip to reveal a secret key. Power based side-channel attacks are based on the observation of general CMOS switching characteristic that the power consumption depends on input signals. Simple power analysis (SPA) attack [Kocher et al. (1999)] is a technique to directly interpret power consumption measurements collected during cryptographic operations. SPA attack requires detailed knowledge about the implementation of the cryptographic algorithm executed by the device under attack. A skilled adversary monitors only one trace or a few traces of power consumption during cryptographic operations and then reveals the secret key. This scenario is not practical since it is very difficult to obtain detailed information of the modern complex hardware implementation such as effective capacitance and resistance of internal nodes.

But profiling makes the scenario practical. In the profiling phase, an adversary can estimate probability distribution of power consumption given any secret key by recoding many power traces at the specific times when cryptographic operations with intermediate values related to the secret key are performing. The more power traces are exploited for the profiling, the more accurately the probability distributions are estimated. The correct secret key can be extracted with various classifiers (or distinguishers) based on the estimated probability distributions and a maximum-likelihood (ML) decision rule. Machine learning techniques such as linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic regression classifier or support vector machine (SVM) can be utilized.

As a non-profiling attack, differential power analysis (DPA) attack has been shown to

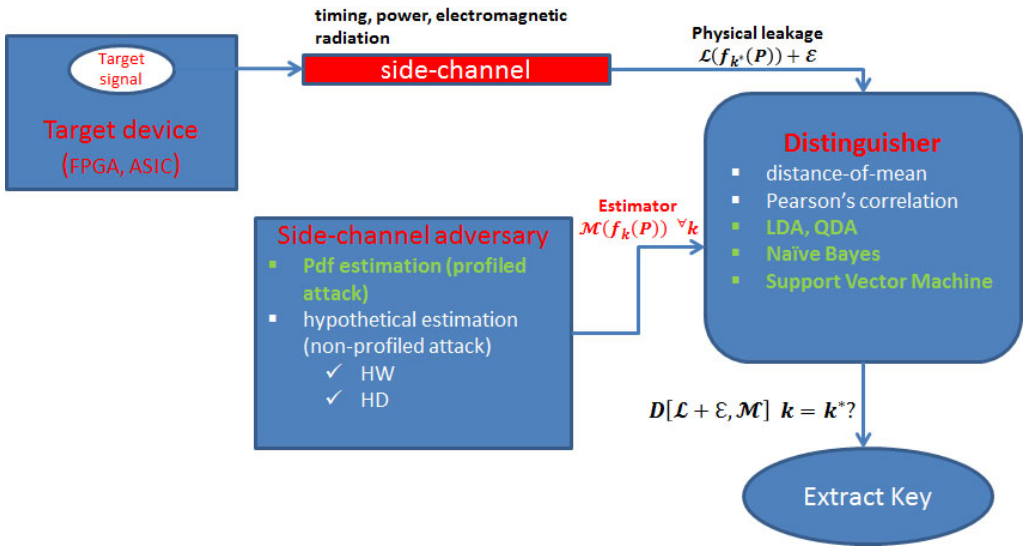


Figure 2.1: Side-channel analysis attacks

be especially effective in finding the secret key by exploiting correlation between estimated power consumption and the processed data. Since this attack needs little knowledge of the implementation of the cryptographic algorithm and can be performed with relatively cheap equipment, it is known to be a major threat to cryptographic devices such as smart cards or embedded systems. The hypothetical power consumption model (or leakage model) of an adversary based on the intermediate value which depends on the key is related to the measured power consumption if the key guess is correct. The adversary's leakage model and the classifier affect success of failure of attack. Fig. 2.1 shows the diagram of the side-channel analysis attacks.

The chapter is organized as follows. The next section presents the general method of differential power analysis attack. Section 2.3 describes profiling attacks with various machine learning classifiers such as LDA, QDA, naïve Bayes classifier and SVM. As an example of SCA application, SCA based disassembler of AVR is proposed in Section 2.4.

2.2 Differential Power Analysis (DPA) Attack

There exists a general attack strategy that is used by all DPA attacks. The first step of the DPA attack is to determine the intermediate value of the cryptographic algorithm executed by the device under attack, which is denoted by $v_i = f(d_i, k^*)$, where d_i is the ith plain text or cipher text and k^* is the secret key.

The second step is to measure the power consumption of the cryptographic device while it encrypts or decrypts D different data blocks including the seleted function at the first step. We denote the power trace as $\vec{t_i} = (t_{i,1}, t_{i,2}, \ldots, t_{i,t^*}, \ldots, t_{i,P})^T$ corresponding to data block d_i, where P denotes the length of the trace and t_{i,t^*} is the power consumption when the selected function at the first step is performed. An adversary measures a trace for each of the D data blocks, and hence, the traces can be written as matrix $\mathbf{T}$ of size $D \times P$: $\mathbf{T} = (\vec{t_1}, \vec{t_2}, \ldots, \vec{t_{t^*}}, \ldots, \vec{t_P})$, where $\vec{t_j}$ for $j = 1, \ldots, P$ is a column vector of size $D \times 1$.

The third step is to calculate a hypothetical intermediate value for all possible k : $v_{i,j} = f(d_i, k_j)$ for $i = 1, \ldots, D$ and $j = 1, \ldots, K$.

The forth step is to map the hypothetical intermediate values to the hypothetical power consumption values: $h_{i,j} = g(v_{i,j}) = g(f(d_i, k_j))$ for $i = 1, \ldots, D$ and $j = 1, \ldots, K$. The most commonly used power consumption models are the Hamming-distance and the Hamming-weight model. The $D \times K$ matrix $\mathbf{H}$ is made at this step : $\mathbf{H} = (\vec{h_1}, \ldots, \vec{h_K})$, where $\vec{h_i}$ for $i = 1, \ldots, K$ is a vector of size $D \times 1$.

The fifth step is to compare the hypothetical power consumption model with the measured power traces. In order to measure the linear relationships between two vectors $\vec{h_i}$ and $\vec{t_j}$ for $i = 1, \ldots, K$ and $j = 1, \ldots, T$, the correlation coefficient is calculated :

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i} - \overline{h_i})(t_{d,j} - \overline{t_j})}{\sqrt{\sum_{i=1}^{D}(h_{d,i} - \overline{h_i})^2 \sum_{i=1}^{D}(t_{d,j} - \overline{t_j})^2}}$$

where $\overline{h_i}$ and $\overline{t_j}$ denote the mean values of the vector $\vec{h_i}$ and $\vec{t_j}$, respectively. If r_{k^*,t^*} of the correct key k^* and the specific time t^* has the distinct peak value, the DPA attack is successful.

2.3 Profiling Attacks

Assuming that the adversary performs a Bayesian attack, s/he first carries out many experiments to measure power consumption in order to model the conditional probability distribution of side-channel power given all possible keys k for $k = 1, \ldots, K$, denoted by $\Pr[\vec{l}|k]$. We call this process the profiling step. After the profiling step, the posterior probability that the secret key is equal to k given any measured power ($\vec{l_j}$) can be computed using Bayes' theorem :

$$\Pr[k|\vec{l_j}] = \frac{\Pr[\vec{l_j}|k]\Pr[k]}{\sum_{k=1}^{K} \Pr[\vec{l_j}|k]\Pr[k]}.$$

Using the maximum-likelihood estimation, the best guess key is the key k that leads to the maximum probability:

$$k = \arg\max_{k \in \mathcal{K}} \prod_{j=1}^{D} \Pr[k|\vec{l_j}]. \tag{2.1}$$

If the prior probability $\Pr[k]$ for $k = 1, \ldots, K$ is uniformly distributed, Eq. (2.1) is equal to the following:

$$k = \arg\max_{k \in \mathcal{K}} \prod_{j=1}^{D} \Pr[\vec{l_j}|k] \tag{2.2}$$

The likelihood probability $\Pr[\vec{l_j}|k]$ at Eq. (2.2) determines the kind of the classifier. The successful classifier selects the correct key : $k = k^*$.

2.3.1 Naïve Bayes classifier

Assuming that $\vec{l_j} \in \mathbb{R}^t$ with $\vec{l_j} = (l_{j,1}, \ldots, l_{j,t})^T$ where $1 \leq t \leq P$ and each $l_{j,i}$ is conditionally independent of every other $l_{j,m}$ for $i \neq m$ given the key k, the classifier is defined as

$$k = \arg\max_{k \in \mathcal{K}} = \prod_{j=1}^{D} \prod_{i=1}^{t} \hat{f}(l_{j,i}|k), \tag{2.3}$$

where $\hat{f}(l_{j,i}|k)$ is a kernel density estimator. The kernel density estimator is written as

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

A kernel is a non-negative real-valued integrable function satisfying the following two requirements:

$$\int_{-\infty}^{\infty} K(u)du = 1$$
$$K(-u) = K(u) \quad \text{for all values of } u.$$

If the Gaussian kernel is used,

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right).$$

A possible estimated bandwidth for kernel density estimation [Wasserman (2006)] is given by

$$\hat{h} = \left[\frac{8\sqrt{\pi}\int K^2(u)du}{3(\int u^2 K(u)du)^2}\right]^{1/5} \min(S, S_{rob})n^{-1/5},$$

where $S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2}$ and $S_{rod} = \frac{\text{median}\{|X_i - F_n^{-1}(\frac{1}{2})|\}}{0.6745}$, $F_n^{-1}(\frac{1}{2})$ denotes the median of the sample. The classifier of Eq. (2.3) is called naïve Bayes classifier.

2.3.2 Linear discriminant analysis

If the likelihood probability $\Pr[\vec{l_j}|k]$ for $k = 1, \ldots, K$ is the multivariate Gaussian density fuction with the mean vector $\vec{\mu_k}$ and common covariance matrix $\mathbf{\Sigma}$ of size $t \times t$, that is,

$$\Pr[\vec{l_j}|k] = \frac{1}{(2\pi)^{t/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{l_j} - \vec{\mu_k})^T \mathbf{\Sigma}^{-1}(\vec{l_j} - \vec{\mu_k})\right),$$

then the classifier is the following:

$$\begin{aligned} k &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \log \Pr[\vec{l_j}|k] \\ &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \log \frac{1}{(2\pi)^{t/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{l_j} - \vec{\mu_k})^T \mathbf{\Sigma}^{-1}(\vec{l_j} - \vec{\mu_k})\right) \\ &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \left[\vec{l_j}^T \mathbf{\Sigma}^{-1}\vec{\mu_k} - \frac{1}{2}\vec{\mu_k}^T \mathbf{\Sigma}^{-1}\vec{\mu_k}\right] \end{aligned}$$

This classifier is the linear discriminant analysis(LDA) classifier.

2.3.3 Quadratic discriminant analysis

The quadratic discriminant analysis (QDA) classifier results from the assumption that each class is drawn from a multivarite Gaussian distribution with a class specific mean row vector $\vec{\mu_k}$ and class specific covariance matrix $\mathbf{\Sigma}_k$. The QDA classifier is the following:

$$\begin{aligned} k &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \log \Pr[\vec{l_j}|k] \\ &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \log \frac{1}{(2\pi)^{t/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\vec{l_j}-\vec{\mu_k})^T \mathbf{\Sigma}_k^{-1}(\vec{l_j}-\vec{\mu_k})\right) \\ &= \arg\max_{k\in\mathcal{K}} \prod_{j=1}^{D} \left[-\frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}\vec{l_j}^T\mathbf{\Sigma}_k^{-1}\vec{l_j} + \vec{l_j}^T\mathbf{\Sigma}_k^{-1}\vec{\mu_k} - \frac{1}{2}\vec{\mu_k}^T\mathbf{\Sigma}_k^{-1}\vec{\mu_k}\right]. \end{aligned}$$

2.3.4 Support vector machine

Support vector machines have been introduced by Vapnik [Vapnik (1995)]. It became more important and popular in recent years when extensions to general nonlinear SVMs have been made [Vapnik (1995), Vapnik (1998)].

2.3.4.1 Linear SVM classifier : separable case

Consider a given training set $\{\vec{x_i}', y_i\}_{i=1}^N$, input patterns $\vec{x_j} \in \mathbb{R}^d$ and output patterns $y_i \in \mathbb{R}$ with class labels $y_i \in \{+1, -1\}$. We define a unique separting hyperplane. We would like to find $\vec{w}$ and b such that

$$\begin{cases} \vec{w}^T\vec{x_i} + b \geq +1 & \text{if } y_i = +1 \\ \vec{w}^T\vec{x_i} + b \leq -1 & \text{if } y_i = -1 \end{cases}$$

which can be rewritten as

$$y_i(\vec{w}^T\vec{x_i} + b) \geq 1, \quad i = 1, \ldots, N. \tag{2.4}$$

The optimal searching hyperplane is the one that maximize the distance between the hyperplane and the nearest points on either side. The distance of $\vec{x_i}$ to the discriminant is

$$\frac{|\vec{w}^T\vec{x_i} + b|}{\|\vec{w}\|} = \frac{y_i(\vec{w}^T\vec{x_i} + b)}{\|\vec{w}\|}$$

,which we would like to be at least some value ρ which is called margin:

$$\frac{y_i(\vec{w}^T\vec{x_i}+b)}{\|\vec{w}\|} \geq \frac{\rho}{2} \quad \forall i$$

By scaling $\vec{w}$ such that $\min_i |\vec{w}^T\vec{x_i}+b| = 1$, the problem is equal to the following optimazation problem :

$$\min_{\vec{w}} \frac{1}{2}\vec{w}^T\vec{w} \quad \text{subject to} \quad y_i(\vec{w}^T\vec{x_i}+b) \geq 1 \quad \text{for} \quad i = 1,\ldots,N.$$

This is a standard quadratic optimization problem, whose complexity depends on d, the dimensionality of the training data. We can convert the optimization problem to a form whose complexity depends on N, the number of training instances, and not on d. The advantage of this new formulation is that it will allow us to rewrite the basic functions in terms of kernel functions [Alpaydin (2010)].

The Lagrangian for this problem is

$$\mathcal{L}_p(\vec{w},b;\vec{\alpha}) = \frac{1}{2}\vec{w}^T\vec{w} - \sum_{i=1}^{N}\alpha_i\{y_i(\vec{w}^T\vec{x_i}+b)-1\}$$

with Lagrange multipliers $\alpha_i \geq 0$ for $i = 1,\ldots,N$. Since the main term is convex and the linear constraints are also convex, this is a convex quadratic optimization problem. Therefore, we can equivalently solve the dual problem, making use of the Karush-Kuhn-Tucher condition. The dual is to maximize $\mathcal{L}_p$ with respect to $\vec{\alpha}$, subject to the constraints that the gradient of $\mathcal{L}_p$ with respect to $\vec{w}$ and b are 0 and also that $\alpha_i \geq 0$. The solution is given by the saddle point of the Lagrangian

$$\max_{\vec{\alpha}} \min_{\vec{w},b} \mathcal{L}(\vec{w},b;\vec{\alpha}),$$

$$\begin{cases} \dfrac{\partial \mathcal{L}_p}{\partial \vec{w}} = 0 \rightarrow \vec{w} = \displaystyle\sum_{i=1}^{N}\alpha_i y_i \vec{x_i} \\ \dfrac{\partial \mathcal{L}_p}{\partial b} = 0 \rightarrow \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0. \end{cases}$$

The resulting classifier is the following:

$$y(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i \vec{x_i}^T\vec{x}+b\right). \tag{2.5}$$

Note that this problem is solved in $\vec{\alpha}$, not in $\vec{w}$. Once we solve for $\vec{\alpha}$, most elements of $\vec{\alpha}$ vanish with $\alpha_i = 0$ and only a few elements have greater than 0. The data related to nonzero α_i are called support vectors and these data points contribute to the sum in the classifier model at Eq. (2.5).

2.3.4.2 Linear SVM classifier : non-separable case

If the two classes are not linearly separable such that there is no hyperplane to perfectly separate the data, the hyperplane that incurs the least error should be searched. The inequality of Eq. (2.4) is modified into the following:

$$y_i(\vec{w}^T\vec{x_i} + b) \geq 1 - \xi_i \quad \text{for} \;\; i = 1, \ldots, N$$

with slack variables $\xi_i > 0$ such that the original inequalities can be violated for certain points if needed. The optimization problem becomes

$$\min_{\vec{w},\vec{\xi}} \mathscr{T}(\vec{w}, \vec{\xi}) = \min_{\vec{w},\vec{\xi}} \frac{1}{2}\vec{w}^T\vec{w} + c\sum_{i=1}^{N}\xi_i$$

subject to

$$\begin{cases} y_i(\vec{w}^T\vec{x_i} + b) \geq 1 - \xi_i \quad \text{for} \;\; i = 1, \ldots, N \\ \xi_i \geq 0 \quad \text{for} \;\; i = 1, \ldots, N. \end{cases}$$

The Lagrangian for this problem is

$$\mathcal{L}_p(\vec{w}, b, \vec{\xi}; \vec{\alpha}, \vec{\nu}) = \mathscr{T}(\vec{w}, \vec{\xi}) - \sum_{i=1}^{N}\alpha_i\{y_i(\vec{w}^T\vec{x_i} + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\nu_i\xi_i$$

and Lagrange multipliers $\alpha_i \geq 0, \nu_i \geq 0$ for $i = 1, \ldots, N$. The solution is given by the saddle point of Lagrangian :

$$\max_{\vec{\alpha},\vec{\nu}} \min_{\vec{w},b,\vec{\xi}} \mathcal{L}(\vec{w}, b, \vec{\xi}; \vec{\alpha}, \vec{\nu}),$$

$$\begin{cases} \dfrac{\partial \mathcal{L}_p}{\partial \vec{w}} = 0 \rightarrow \vec{w} = \displaystyle\sum_{i=1}^{N}\alpha_i y_i \vec{x_i} \\ \dfrac{\partial \mathcal{L}_p}{\partial b} = 0 \rightarrow \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \dfrac{\partial \mathcal{L}_p}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq c, \quad i = 1, \ldots, N. \end{cases}$$

2.3.4.3 Nonlinear SVM classifiers

If the problem is nonlinear, we can map the problem to a high dimensional feature space ($\mathbb{R}^{n_h}$) by doing a nonlinear transformation using suitably chosen basic function. After the nonlinear mapping $\varphi(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^{n_h}$, a construction of the linear separating hyperplane is done in this high dimensional feature space. The optimization problem becomes

$$\min_{\vec{w},\vec{\xi}} \mathscr{T}(\vec{w},\vec{\xi}) = \min_{\vec{w},\vec{\xi}} \frac{1}{2}\vec{w}^T\vec{w} + c\sum_{i=1}^{N}\xi_i$$

subject to

$$\begin{cases} y_i(\vec{w}^T\varphi(\vec{x_i}) + b) \geq 1 - \xi_i \quad \text{for } \ i = 1,\ldots,N \\ \xi_i \geq 0 \quad \text{for } \ i = 1,\ldots,N. \end{cases}$$

One constructs the Lagrangian :

$$\mathcal{L}_p(\vec{w},b,\vec{\xi};\vec{\alpha},\vec{\nu}) = \mathscr{T}(\vec{w},\vec{\xi}) - \sum_{i=1}^{N}\alpha_i\{y_i(\vec{w}^T\varphi(\vec{x_i}) + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\nu_i\xi_i$$

and Lagrange multipliers $\alpha_i \geq 0, \nu_i \geq 0$ for $i = 1,\ldots,N$. The solution is given by the saddle point of Lagrangian :

$$\max_{\vec{\alpha},\vec{\nu}} \min_{\vec{w},b,\vec{\xi}} \mathcal{L}(\vec{w},b,\vec{\xi};\vec{\alpha},\vec{\nu}),$$

$$\begin{cases} \dfrac{\partial\mathcal{L}_p}{\partial\vec{w}} = 0 \to \vec{w} = \displaystyle\sum_{i=1}^{N}\alpha_i y_i\varphi(\vec{x_i}) \\ \dfrac{\partial\mathcal{L}_p}{\partial b} = 0 \to \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \dfrac{\partial\mathcal{L}_p}{\partial\xi_i} = 0 \to 0 \leq \alpha_i \leq c, \quad i = 1,\ldots,N. \end{cases}$$

We make use of the Mercer condition by choosing a kernel

$$K(\vec{x_k},\vec{x_l}) = \varphi(\vec{x_k})^T\varphi(\vec{x_l}).$$

By applying this theorem one can avoid computations in the huge dimensional feature spce. The nonlinear SVM classifier takes the form

$$\begin{aligned} y(\vec{x}) &= \text{sign}\left[\sum_{i=1}^{N}\alpha_i y_i K(\vec{x},\vec{x_i}) + b\right] \\ &= \text{sign}\left[\sum_{i=1}^{\#SV}\alpha_i y_i K(\vec{x},\vec{x_i}) + b\right] \end{aligned}$$

with $\#SV$ denotes the number of support vectors.

Several kernels $K(\cdot,\cdot)$ are the followings:

$$K(\vec{x},\vec{x_i}) = \vec{x_i}^T\vec{x} \quad \text{(linear kernel)}$$

$$K(\vec{x},\vec{x_i}) = (\vec{x_i}^T\vec{x}+1)^d \quad \text{(polynomial kernel of degree } d\text{)}$$

$$K(\vec{x},\vec{x_i}) = \exp(-\|\vec{x}-\vec{x_i}\|^2/\sigma^2) \quad \text{(RBF kernel)}$$

$$K(\vec{x},\vec{x_i}) = \tanh(\kappa\vec{x_i}^T\vec{x}+\theta) \quad \text{(MLP kernel)}.$$

2.4 Side-channel Based Disassembler of AVR microcontroller

The main focus of the side-channel based disassembler is to extract assembly level code along with the control flow graph from the side-channel leakage. The significant difference between side-channel analysis attacks and side-channel based disassembler is the number of required power sample traces to succeed assuming that both use profiled templates. Side-channel analysis attacks for secret data leakage have more flexibility in the number of required sample traces because the adversary can control the plaintext input of the target device. But side-channel disassembler does not have similar controllability of the target device. It should recognize a power or EM trace of each executed instruction. In other words, side-channel disassembler should estimate which instruction is executing, which register is used, or what value is processed with only one sample. This makes side-channel disassembler a more challenging problem. It requires more advanced estimation techniques.

There exist many challenging problems in complete disassembly. Identification of destination register, Rd and source register, Rs for register transfer instructions or data for load or store instructions is difficult. For a more complete monitoring of programs, many variables such as register names, register data, memory address and values for load/store instructions should be estimated. Moreover, recent embedded microcontrollers such as ARM Cortex-M or Cortex-A series have more complex architectures with deeper pipeline stages and larger in-

struction sets. Their system clock frequency also approaches a few hundred MHz or about $1GHz$. It becomes more difficult to disassemble programs on the recent embedded devices. Lastly, the acquisition methods of power or EM emanations with oscilloscopes would be significantly stretched because of higher system clock frequencies of the target devices (of the order of 1GHz). High sampling rate oscilloscopes (over $5GS/s$) are needed to collect power or EM leakage information generated at $1GHz$ frequency to prevent loss of fidelity. For profiling, multiple data samples are needed, which may be a few billion (2^{32}) in case of 32-bit instruction sets. This can make the profiling process significantly time consuming. Fast bandwidth between the oscilloscope and the desktop or laptop to store the sampled data is also required. The oscilloscopes with high sampling, high vertical resolution and fast bandwidth are fairly expensive (over \$ 20 K).

In this section, we propose power side-channel based disassembler of AVR using hierarchical quadratic discriminant analysis (QDA) classifier and SVM classifier. Even though AVR microcontroller is not state-of-the-art devices, we believe that our method can be a starting point to disassemble recent embedded microcontroller. Our disassembler includes estimating which registers are used and what value the registers have as well as which instructions are executed. Also, we compare QDA classifier with other classifiers such as naïve Bayes classifier, LDA classifier and the SVM classifier.

2.4.1 Preliminary Experiments

We conducted preliminary experiments to check if similar style instructions of AVR ATmega328p μC can be disassembled through power analysis. We considered 6 data transfer instructions (`add, sub, and, mov, or, eor`) from the source register (`Rs16` $\sim$ `Rs25`) to the destination register (`Rd16` $\sim$ `Rs25`). The goal of this experiment is to identify which instruction is executed and which `Rd` and `Rd` are exploited.

The AVR μC has 2 pipeline stages and with a clock frequency of 16 MHz. Tektronix DPO-4032 oscilloscope is used to sample the power pin at 1.25GS/s, 20MHz bandwidth, 1000 sample points and 128 average mode. Using this oscilloscope, the voltage of the shunt resistor

between the GND pin and ground is measured. Each power trace is measured with the following program segment template: `sbi, 5 nops, targeted profiled instruction, 5 nops`. The `sbi` instruction is executed for the trigger signal. In order to remove power consumption of `sbi` instruction and electrical noise, we compute the difference between each power trace and the reference power traces of `sbi` and 10 nops sequence. For profiling, 3000 power traces per each instruction with randomly selected `Rs` and `Rd` (the values of the `Rs` and `Rd` also are randomly distributed) are sampled. We also measures 3000 power traces per each `Rd` with randomly selected instruction and `Rs` and 3000 power traces per each `Rs` with randomly selected instruction and `Rd`. These training data will be used for the classification. There exist 3 different class groups. The first class group represents the instruction : $C_{int} = \{c_{add}, c_{sub}, c_{and}, c_{mov}, c_{or}, c_{eor}\}$. The second class group and third class group represents the source register and the destination register, respectively : $C_{Rd} = \{c_{rd16}, \ldots, c_{rd25}\}, C_{Rs} = \{c_{rs16}, \ldots, c_{rs25}\}$.

Before the training, the measured traces should be preprocessed in order to remove noise and to make different classes be more distinguishable. The continuous wavelet transform (CWT) to extract distinct features among all classes in both the frequency and the time domain is used. Principal components (time and frequency) are extracted from the wavelet transform of the collected traces. Only the principal time and frequency region features are kept, and all the other time and frequency domain signals are zeroed. An inverse CWT of the shaped time and frequency signal contains only the principal features in the time domain. The next step is the feature selection to look for which any specific times are significant. The total number of sampling point per each inverse-CWT power trace is 160. Assuming that each sampling point has normal distribution with the mean μ_i and the variance σ_i^2 for $i = 1, \ldots, 160$, the probability distribution of each class has the multivariate(160-dimensional) normal distribution. The computation complexity is very expensive and not practical. Thus, the dimensionality reduction or feature selection is required.

The Kullback-Leibler divergence is useful metric for the feature selection. The more the KL divergence between two random variable, the more distinguishable two random variables. The specific sampling points should have large KL-divergence value. Also, the specific sampling points does not have dependency (or collinearity). To satisfy two conditions, the specific

sampling points have locally maximum value. As a result, 160 dimensionality can reduce to about 10. Fig. 2.2 shows the preprocessing for the separation of power traces of `and` and `sub`. The scatter plots of the raw power traces are overlapped. After CWT analysis and the feature selection, the scatter plots does not have the overlapping region.

3000 power traces with the specific sample points per each class are used for the training depending on the classifier. Linear discriminant analysis (LDA), quadratic discriminant analysis(QDA) and naïve Bayes method are executed. Each classifier has different assumption. LDA assumes that the distribution of each class has multivariate normal distribution with the same covariance matrix (Σ). QDA has more flexibility than LDA since they assumes that the distribution of each class has multivariate normal distribution with the different covariance matrix ($\boldsymbol{\Sigma}_i \neq \boldsymbol{\Sigma}_j \quad \forall i \neq j$). Naïve Bayes classifier assumes that the probability distribution of each specific sampling point of each class can be various distribution independently. The marginal probability distribution of power traces at a specific sampling point resembles the normal distribution and the marginal probability distribution of each class has different variance. Fig. 2.3 shows the kernel density estimation of each instruction at a specific sampling point. Since the characteristic of power traces satisfies the assumption of QDA, the QDA classifier has the best performance among three classifiers (LDA, QDA, naïve Bayes classifier). The successful recognition rates (SR) of instructions (`add, sub, and, mov, or, eor`) according to classifiers are shown in Table 2.1.

The registers from `Rd16` (or `Rs 16`) to `Rd25` (or `Rs25`) can be grouped into 4 classes depending on the Hamming weight of the binary address of the register. The Hamming weight of the register address is very related to the power consumption during the fetch and decoding of the instruction since the address of registers occupies 10-bit length of the 16-bit instruction code. The classification of registers (`Rd, Rs`) can be executed hierarchically. The Hamming weight of the address of the register is identified and then the address of the register in the Hamming weight class is recognized. Fig. 2.4 shows the hierarchical classification of the register (`Rd, Rs`) using the QDA classifer and the successful recognition rate of the Hamming weight class and the address. The successful recognition rates of the Hamming weight of `Rd` and `Rs` are 80% and 69.6%, respectively. The address of the register `Rd` and `Rs` with the 2-Hamming weight is

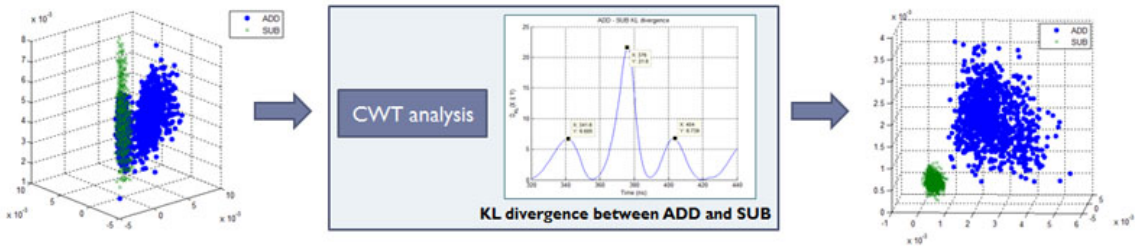


Figure 2.2: Separation of power traces of `ADD` and `SUB`

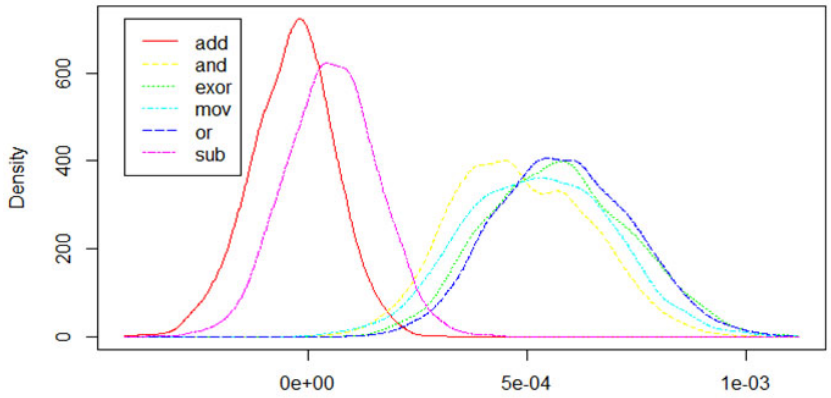


Figure 2.3: Kernal density estimation denpending on instructions at a specific sampling point

recognized at the rate of 77.8% and 67.5%, respectively. The address of the register `Rd` and `Rs` with the 3-Hamming weight is recognized at the rate of 83% and 73.6%, respectively. Fig. 2.4 shows the hierarchical classification of registers `Rd` and `Rs` and successful recognition rates.

2.4.2 SVM

LS-SVM(Least Squares Support Vector Machine) [Leuven (2011)] is used to classify instructions. Fig. 2.5 shows the successful recognition rates of LS-SVM and QDA to classify measured power traces into two classes. LS-SVM mostly overcomes QDA classifier in terms of

Table 2.1: Successful recognition rate(SR) of instructions according to classifiers

Classifier	SR
LDA	37 %
QDA	70.1 %
naïve Bayes	37.1 %

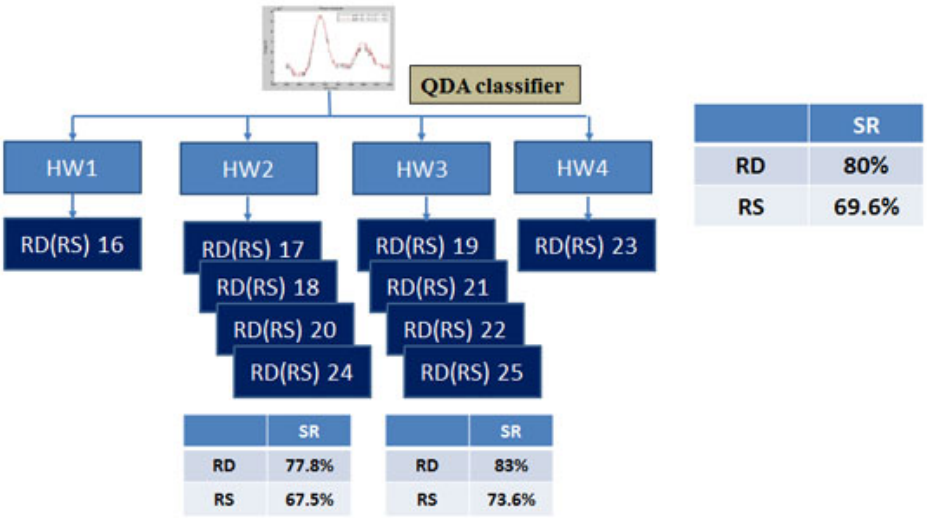


Figure 2.4: Hierarchical classification of registers and successful recognition rate

the successful recognition rate. In case of $C = \{c_{exor}, c_{mov}\}$, LS-SVM results in 8.5 % better performance than QDA classifier. Table 2.2 shows successful recognition rates of LS-SVM and QDA classifier depending on various classes. LS-SVM increases 12 % successful recognition rates of 6 instructions (`add, sub, and, mov, or, eor`) compared with QDA result.

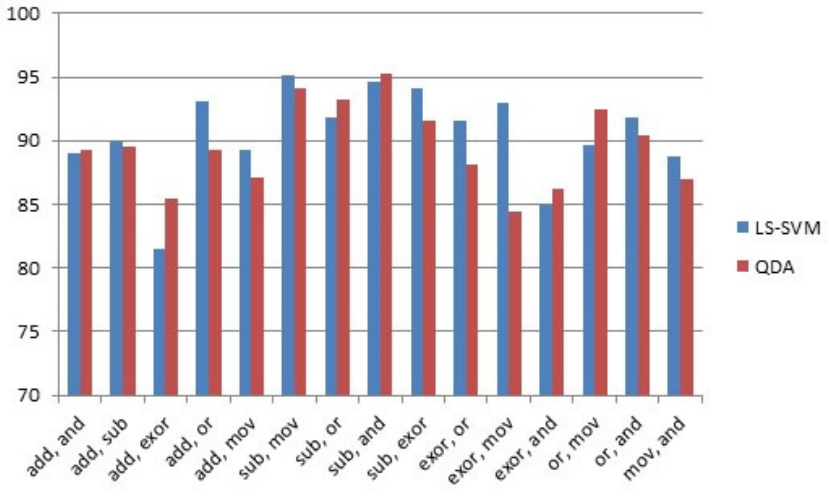


Figure 2.5: LS-SVM vs QDA

Table 2.2: SR of instructions using LS-SVM and QDA classifiers

	LS-SVM	QDA
add vs and	88.97 %	89.26 %
add vs sub	89.88 %	89.58 %
add vs exor	81.46 %	85.46 %
add vs or	93.10 %	89.21 %
add vs mov	89.28 %	87.13 %
sub vs mov	95.17 %	94.05 %
sub vs or	91.80 %	93.28 %
sub vs and	94.67 %	95.25 %
sub vs exor	94.13 %	91.57 %
exor vs or	91.5 %	88.12 %
exor vs mov	92.92 %	84.42 %
exor vs and	84.92 %	86.25 %
or vs mov	89.63 %	92.4 %
or vs and	91.85 %	90.37 %
mov vs and	88.76 %	86.92 %
add vs and vs exor	86.83 %	78.56 %
add vs and vs mov	82.73 %	79.93 %
add vs and vs or	83.64 %	82.91 %
add vs and vs sub	85.83 %	85.16 %
add vs exor vs mov	85.23 %	77. 31 %
add vs exor vs or	82.57 %	79. 82 %
add vs exor vs sub	86.01 %	81.6 %
add vs and vs exor vs mov vs or vs sub	82 %	70.1 %

CHAPTER 3. SECURITY METRICS

3.1 Introduction

In this chapter, we focus on SCA metrics to flag insecure combinational modules within a complete cryptographic system. We assume that the adversary is powerful enough to estimate power consumption accurately to account for the number of switching transitions including glitches in a complete cryptographic system. From the designer point of view, this assumption bases security on an all powerful adversary. Even though simulation based profiling can be performed at the logic level, it should be avoided due to efficiency. The number of input vectors of the simulation increases exponentially in the number of input bits, denoted by n. The power consumption can be estimated more efficiently by the Monte Carlo probabilistic methods. The Monte Carlo probabilistic power estimation model is based on the fact that power consumption depends on the transition probability and capacitance of the output node of logic gates [Najm (1994)]. But the probabilistic power estimation model does not consider glitches caused by the gate delay.

First, we propose a new stochastic power estimation method using renewal process and linear regression which includes the dynamic power caused by the glitching phenomenon. This method is used at the logic level design for efficient power profiling. Given any input transitions of the combinational circuit, the normal power distribution with the mean μ and variance σ^2 can be obtained.

Second, security metrics to capture SCA vulnerability with the power estimation are defined and computed. The CAD for design flow includes the SCA metric estimation and optimization just as area and delay estimation and optimization. The SCA security is quantified using **(1)** the normalized variance metric (or the coefficient of variance) [Basel Halak (2013)], **(2)**

Kullback-Leibler divergence and [S. Kullback and R. A. Leibler (1951)] **(3)** the information theoretic metric of the profiled power distribution. In our design flow, SCA vulnerability should be verified with these metrics at all implementation abstraction levels from logic (or gate) to layout level. We estimate Kullback-Leibler divergence from the power distribution gathered from the approximate and quick renewal process based logic level simulation. Once the SCA metric at the higher logic abstraction level is within safe bounds, the design flow can enter the next abstraction level refinement. This abstraction refinement (as in logic level to netlist level) introduces details that may develop new SCA vulnerabilities. Hence an acceptable SCA metric value at higher abstraction layers still necessitates SCA metric computation at lower levels. The mutual information metric is computed at the layout level with multiple SPICE level circuit simulations. The acceptable thresholds for SCA security metric are defined theoretically. If any combinational module has a value larger than the threshold, it is flagged as a vulnerable module. The vulnerable modules should be transformed into a secure module. One of the methods to accomplish this is to use a secure logic design style such as t-private circuits [Ishai et al. (2003)] or masked dual-rail dynamic logic [Mangard (2005)].

The chapter is organized as follows. The next section presents the basic definitions and lemmas for power estimation. We develop the power leakage model using renewal process and linear regression in Section 3.3. The SCA security metrics are presented in Section 3.4. The recognition rate using maximum likelihood estimation is defined in Section 3.5. The recognition rate is very related to KL divergence. Experimental results are presented in Section 3.6. Finally, Section 3.7 concludes the chapter.

3.2 Basic Definition and Lemma

In this section, the basic definitions and lemmas for stochastic power estimation of combinational circuits are presented.

Definition 1 (Boolean difference)**.** *[Mohyuddin et al. (2008)] The* ***partial Boolean difference*** *of $f(x_0, x_1, \ldots, x_{n-1})$ with respect to one variable or a subset of its variables is defined*

as:

$$\frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{x_i'}$$
$$\frac{\partial f}{\partial (x_{i_1} x_{i_2} \cdots x_{i_k})} = f_{x_{i_1} x_{i_2} \cdots x_{i_k}} \oplus f_{x'_{i_1} x'_{i_2} \cdots x'_{i_k}}.$$

where $f_{x_i} = f(x_0, x_1, \ldots, 1, \ldots, x_{n-1})$ *and* $f_{x_i'} = f(x_0, x_1, \ldots, 0, \ldots, x_{n-1})$. *The* ***total Boolean difference*** *of* $f(x_0, x_1, \ldots, x_i, \ldots, x_{n-1})$ *with respect to a k-variable subset of its inputs is defined as:*

$$\frac{df}{d(x_{i_1} x_{i_2} \cdots x_{i_k})} = \sum_{j=0}^{2^{k-1}-1} \left.\frac{\partial f}{\partial \vec{x}}\right|_{m_j} (m_j + m_{2^k - j - 1})$$

where m_j*'s are defined as follows:*

$$m_0 = x'_{i_1} x'_{i_2} \cdots x'_{i_{n-1}} x'_{i_k}$$
$$m_1 = x'_{i_1} x'_{i_2} \cdots x'_{i_{n-1}} x_{i_k}$$
$$\vdots$$
$$m_{2^k-1} = x_{i_1} x_{i_2} \cdots x_{i_{n-1}} x_{i_k},$$

and

$$\left.\frac{\partial f}{\partial \vec{x}}\right|_{m_j} = \frac{\partial f}{\partial (x^*_{i_1} x^*_{i_2} \cdots x^*_{i_k})}$$
$$\textit{where } m_j = x^*_{i_1} x^*_{i_2} \cdots x^*_{i_k} (x^*_i = x_i \text{ or } x'_i).$$

Definition 2 (Observability)**.** *The* ***observability*** *of* x_i *is the probability that* x_i *is observable at the output* $y = f(x_0, x_1, \cdots, x_i, \cdots, x_{n-1})$ *when the polarity of* x_i *is changed. Using the boolean difference with respect to* x_i*,*

$$\mathbf{Ob}_y(x_i) = \Pr\left[\frac{\partial f}{\partial x_i}\right] = \Pr[f_{x_i} \oplus f_{x_i'}].$$

In general the kth order observability of a subset of inputs $(x_{i_1} x_{i_2} \cdots x_{i_k})$ *at the output* $y = f(x_0, x_1, \cdots, x_{n-1})$ *is defined as:*

$$\mathbf{Ob}_y(x_{i_1}, x_{i_2}, \cdots, x_{i_k}) = \Pr\left[\frac{df}{d(x_{i_1} x_{i_2} \cdots x_{i_k})}\right].$$

The conditional observability given $x^*_{j_1}, ..., x^*_{j_m}$ *is defined as:*

$$\mathbf{Ob}_y(x_{i_1}, x_{i_2}, \cdots, x_{i_k} | x^*_{j_1}, ..., x^*_{j_m}) = \Pr\left[\frac{df_{x^*_{j_1}, ..., x^*_{j_m}}}{d(x_{i_1} x_{i_2} \cdots x_{i_k})}\right].$$

Definition 3 (Logic network graph)**.** *[Micheli (1994)] The logic network graph $G(V, E, W(E))$ is a directed acyclic weighted graph with the vertex set V which is in one-to-one correspondence with the primary inputs, local functions and primary outputs and the weight set $W(E) = \{w((v_i, v_j))|(v_i, v_j) \in E\}$. We denote a path $\mathscr{P}$ from the vertex v_1 to another vertex v_n by an alternating sequence of distinct vertices and edges such as the following equation :*

$$\mathscr{P} = \{v_1, (v_1, v_2), v_2, (v_2, v_3), \ldots, v_n\}.$$

Definition 4 (Reconvergent node)**.** *Two distinct directed paths are reconvergent if they start at a common vertex (v_a) and terminate at another common vertex (v_b). The vertex v_a is called a reconvergent fanout and the vertex v_b is called a reconvergent node.*

Definition 5 (Effective capacitance)**.** *We define the effective capacitance $C_y(x_i)$ as the average of total switched capacitances of all logic gates on the path from the input x_i to the output y when the input x_i is switched. We use the lumped-C model which describes the effective capacitance $C_y(x_i)$ as a lumped capacitance containing the intrinsic and the extrinsic capacitance of all logic gates.*

The effective capacitance of a CMOS logic gate depends on the diffusion capacitance C_d of the logic, the wiring capacitance C_w and the gate capacitance C_g of the following logic gates [Weste and Harris (2010)]. The effective capacitance of a logic gate u is given by the following equation:

$$C(u) = C_d(u) + C_w + \sum_{i=1}^{n} C_g(u_i) \tag{3.1}$$

where n is the number of logic gates u_i driven by the logic gate u and $C_g(u_i)$ is the gate capacitance of each of the following logic gates. These capacitances C_d, C_w and C_g depend on the physical properties of the process technology.

Assumption 1. For technology independent estimates at the logic level, we assume that the mobility of the nMOS transistors is two times the mobility of the pMOS transistor and that the transistor widths are chosen to achieve balanced rising and falling transition delays

[Weste and Harris (2010)]. We also assume that a unit transistor has the same gate capacitance C as the source/drain diffusion capacitance ($C = C_g = C_d$) and C_w is equal to zero.

Lemma 1. *The power consumption $\mathbf{P}_y(x_i)$ of logic gates on the path from the input x_i to the output y caused by switching the input x_i depends on the output observability of the input x_i and the effective capacitance $C_y(x_i)$. We define the dynamic power, $\mathbf{P}_y(x_i)$, caused by switching the input x_i as*

$$\mathbf{P}_y(x_i) = 0.5V_{DD}^2 f \cdot \mathbf{Ob}_y(x_i)C_y(x_i)$$

where f is the frequency and V_{DD} is the supply voltage.

Lemma 2. *Given a logic network graph $G(V, E, W(E))$, where the weight set $W = \{w((v_i, v_j))| w((v_i, v_j)) = \mathbf{Ob}_j(i), (v_i, v_j) \in E\}$, the path observability $\mathbf{Ob}^i_{a_n}(a_0)$ of the input a_0 at the output a_n on the path $\mathscr{P}_i = \{v_{a_0}, (v_{a_0}, v_{a_1}), v_{a_1}, \ldots, v_{a_n}\}$ is given by the following equation:*

$$\mathbf{Ob}^i_{a_n}(a_0) = \prod_{i=0}^{n-1} \mathbf{Ob}_{a_{i+1}}(a_i) = \prod_{\forall e \in \mathscr{P}_i} w(e). \tag{3.2}$$

Generally, there exist various paths since the path $\mathscr{P}_i$ may have the reconvergent fanout r_f and reconvergent node r_n. The observability of the input r_f at the output r_n is approximately equal to the sum of observability along all paths:

$$\mathbf{Ob}_{r_n}(r_f) = \sum_{i=0}^{m-1} \mathbf{Ob}^i_{r_n}(r_f) \tag{3.3}$$

where m is the number of paths.

Proof. By the Shannon expansion, the output y_0 is expressed by the following equations :

$$y_0 = f(a_0, \cdots) = a_0 f_{a_0} + a_0' f_{a_0'}. \tag{3.4}$$

Assuming f is decomposed into $a_1 = f^1(a_0, \ldots)$ and $y_0 = f^r(a_1, \ldots)$,

$$\begin{aligned} y_0 &= a_1 f^r_{a_1} + a_1' f^r_{a_1'} \\ &= \{a_0 f^1_{a_0} + a_0' f^1_{a_0'}\} f^r_{a_1} + \{a_0 f^1_{a_0} + a_0' f^1_{a_0'}\}' f^r_{a_1'}. \end{aligned} \tag{3.5}$$

At (3.4), using (3.5) f_{a_0} and $f_{a_0'}$ are the following equations :

$$\begin{aligned} f_{a_0} &= f(1,\ldots) \\ &= f^1_{a_0} f^r_{a_1} + \{f^1_{a_0}\}' f^r_{a_1'}. \\ f_{a_0'} &= f(0,\ldots) \\ &= f^1_{a_0'} f^r_{a_1} + \{f^1_{a_0'}\}' f^r_{a_1'}. \end{aligned}$$

The boolean difference of $f(a_0,\ldots)$ with respect to the variable a_0 is

$$\begin{aligned} \frac{\partial f}{\partial a_0} &= f_{a_0} \oplus f_{a_0'} \\ &= [f^1_{a_0} f^r_{a_1} + \{f^1_{a_0}\}' f^r_{a_1'}] \oplus [f^1_{a_0'} f^r_{a_1} + \{f^1_{a_0'}\}' f^r_{a_1'}] \\ &= [\{f^1_{a_0}\}' + \{f^r_{a_1}\}'][f^1_{a_0} + \{f^r_{a_1'}\}'][f^1_{a_0'} f^r_{a_1} + \{f^1_{a_0'}\}' f^r_{a_1'}] + \\ &\quad [f^1_{a_0} f^r_{a_1} + \{f^1_{a_0}\}' f^r_{a_1'}][\{f^1_{a_0'}\}' + \{f^r_{a_1}\}'][f^1_{a_0'} + \{f^r_{a_1'}\}'] \\ &= \{f^1_{a_0}\}' f^1_{a_0'} f^r_{a_1} \{f^r_{a_1'}\}' + f^1_{a_0} \{f^1_{a_0'}\}' \{f^r_{a_1}\}' f^r_{a_1'} + \\ &\quad f^1_{a_0} \{f^1_{a_0'}\}' f^r_{a_1} \{f^r_{a_1'}\}' + \{f^1_{a_0}\}' f^1_{a_0'} \{f^r_{a_1}\}' f^r_{a_1'} \\ &= (f^1_{a_0} \oplus f^1_{a_0'})(f^r_{a_1} \oplus f^r_{a_1'}) \\ &= \frac{\partial f^1}{\partial a_0} \frac{\partial f^r}{\partial a_1}. \end{aligned}$$

Similarly, the logic function, f^r is decomposed into $n-1$ logic functions, $a_i = f^i(a_{i-1},\ldots)$ for $i = 2,\ldots,n$. The boolean difference of $f^r(a_1,\ldots)$ with respect to the variable a_1 is

$$\frac{\partial f^r}{\partial a_1} = \frac{\partial f^2}{\partial a_1} \cdots \frac{\partial f^n}{\partial a_{n-1}}.$$

Thus,

$$\frac{\partial f}{\partial a_0} = \frac{\partial f^1}{\partial a_0} \frac{\partial f^2}{\partial a_1} \cdots \frac{\partial f^n}{\partial a_{n-1}}.$$

The path observability of the primary input a_0 at the primary output y_0 on the path P_i is

$$\begin{aligned} \mathbf{Ob}^i_{y_0}(a_0) &= \Pr\left[\frac{\partial f}{\partial a_0}\right] \\ &= \Pr\left[\frac{\partial f^1}{\partial a_0}\right] \Pr\left[\frac{\partial f^2}{\partial a_1}\right] \cdots \Pr\left[\frac{\partial f^n}{\partial a_{n-1}}\right] \\ &= \prod_{i=0}^{n-1} \mathbf{Ob}_{a_{i+1}}(a_i). \end{aligned}$$

□

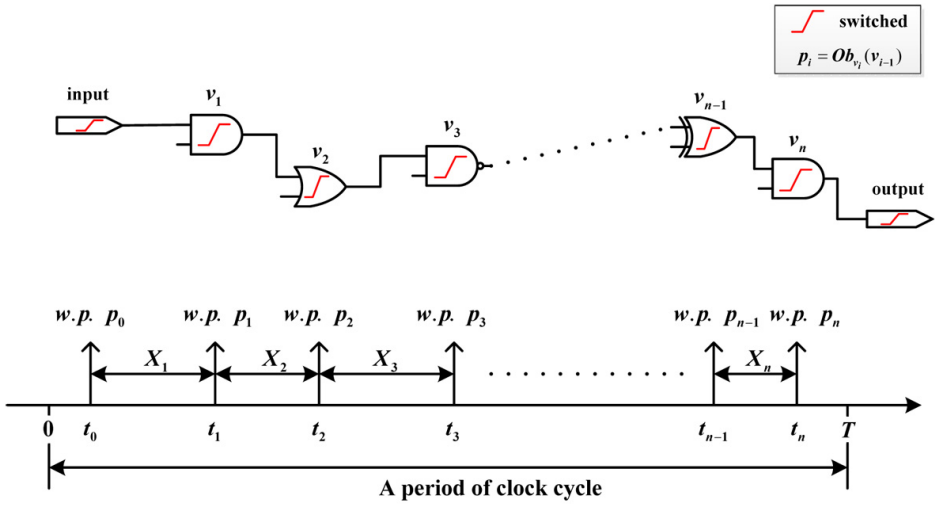


Figure 3.1: Renewal process of logic network

Lemma 3. *We let $C_{a_i}(a_{i-1})$ for $i = 1, \ldots, n$ be the effective capacitance of each local logic function, $a_i = f^i(a_{i-1}, \ldots)$ for $i = 1, \ldots, n$. The effective capacitance $C_{y_0}(a_0)$ of the complete logic function $y_0 = f(a_0, \ldots)$ due to the primary input a_0 is given by the following equation:*

$$C_{y_0}(a_0) = \frac{1}{\mathbf{Ob}_{y_0}(a_0)} \sum_{i=0}^{n-1} \mathbf{Ob}_{a_{i+1}}(a_0) C_{a_{i+1}}(a_i)$$
$$= \frac{1}{\mathbf{Ob}_{y_0}(a_0)} \sum_{i=0}^{n-1} \left[\prod_{j=0}^{i} \mathbf{Ob}_{a_{j+1}}(a_j) \right] C_{a_{i+1}}(a_i). \tag{3.6}$$

If y_0 and a_0 are a reconvergent node and fanout pair, respectively and there exists m paths between the two nodes,

$$C_{y_0}(a_0) = \frac{1}{\mathbf{Ob}_{y_0}(a_0)} \sum_{j=0}^{m-1} \sum_{i=0}^{n^j-1} \mathbf{Ob}_{a_{i+1}^j}(a_0) C_{a_{i+1}^j}(a_i^j) \tag{3.7}$$

where a_i^j is the node in the jth path.

3.3 Power Model Using Renewal Process and Linear Regression

3.3.1 Renewal process

We propose new power estimation model using the renewal process and linear regression in this section. We can model the switching behavior of logic circuits as a renewal process.

The transition or switching of each logic gate is regarded as a renewal. When switching events propagate through connected logic networks, the input transition events cause renewals at output nodes sequentially with renewal intervals between successive logic gates corresponding to the gate delays. The expected number of renewals includes normal transitions and unintended glitches due to variable delays and can be used for accurate power estimation. The accuracy and computational complexity of power estimation depends on the probability density function of the renewal intervals, X_i.

There exists a path $\mathscr{P}$ from the vertex v_1 to another vertex v_n in the logic network $G(V, E, W(E))$. Note that the $i-1$st logic gate should be triggered for switching the ith logic gate. Some logic gates are triggered in sequential order from switching the primary input x at time t_0 with the probability p_0. The logic gates $v_1, v_2, \cdots, v_n$ are triggered at time $t_1, t_2, \ldots, t_n$ with the probability $p_1, p_2, \ldots, p_n$, respectively. The renewal process [Nelson (1995)] can be used for modeling the behavior of the logic network. The transition points t_i are renewal points. Let X_n be the renewal interval between successive renewal points, $t_n - t_{n-1}$. Fig. 3.1 describes the renewal process of the logic network.

We define $S_0 = 0$ and

$$S_n \stackrel{def}{=} X_1 + X_2 + \cdots + X_n, \quad n \geq 1,$$

and let

$$N(t) \stackrel{def}{=} \max\{n : S_n \leq t\}.$$

Recall that S_n is the time of the nth renewal and $N(t)$ is the number of renewals that occur within the interval $(0, t]$. We let F_n be the distribution of the sum of n independent random variables distributed as X_i. F_n is defined as the nth-fold convolution of F_{X_i}, that is,

$$F_n(x) \stackrel{def}{=} F_{X_1} * F_{X_2} * \cdots * F_{X_n}$$

and let $f_n(x)$ be the corresponding density function. We are concerned with properties of $N(t)$.

Using $F_n(x)$, the density of $N(t)$ can be derived as

$$\begin{aligned}\Pr[N(t)=n] &= \Pr[N(t)\le n] - \Pr[N(t)\le n-1] \\ &= \Pr[S_{n+1}>t] - \Pr[S_n>t] \\ &= F_n(t) - F_{n+1}(t).\end{aligned}$$

The expected number of renewals during a given period of time, denoted by $R(t)$ can be obtained as follows:

$$R(t) = \sum_{i=1}^{n} i\Pr[N(t)=i] = \sum_{i=1}^{n} i(F_i(t) - F_{i+1}(t)) = \sum_{i=1}^{n} F_i(t) \tag{3.8}$$

Note that X_i can be modeled as the time for transition event from the switching event X_{i-1} or it can be modeled as time to the clock edge. The first scenario models X_i as a random variable with probability p_i as a normal distribution with the mean μ_i and the variance σ_i^2. The second scenario captures $T - t_{i-1}$ with probability $1 - p_i$, where T is the period of clock cycle. This means that if the logic node v_i transitions with probability p_i, X_i is a random value which includes the logic gate delay and wire delay. Otherwise, X_i is the remaining time to the period of the clock cycle.

We define the probability density of X_i as the followings:

$$f_{X_i}(t) = (1-p_i)\delta\left(t - (T - t_{i-1})\right) + p_i n(t;\mu_i,\sigma_i)$$

where $n(t;\mu_i,\sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu_i}{\sigma_i}\right)^2}$, $\delta(t)$ is the impulse function or Dirac delta function, $p_i = \mathbf{Ob}_{v_i}(v_{i-1})$ and T is the period of the clock cycle. The ith-fold convolution $F_i(t)$ at $t = T$ is the followings:

$$F_i(T) = \left(\prod_{j=1}^{i} p_j\right) \int_{t=0}^{T} n\left(t; \sum_{j=1}^{i}\mu_j, \sqrt{\sum_{j=1}^{i}\sigma_j^2}\right) \cong \left(\prod_{j=1}^{i} p_j\right)$$

The expected number of switched logic gates during a clock cycle $R(T)$ is the followings by Eq. (3.8):

$$R(T) = \sum_{i=1}^{n} F_i(T) = \sum_{i=1}^{n}\prod_{j=1}^{i} p_j = \sum_{i=1}^{n}\prod_{j=1}^{i} \mathbf{Ob}_{v_i}(v_{i-1}).$$

$R(T)$ means the expected number of switched signals on the path $\mathscr{P}$ by triggering a input during a clock cycle. If multiple inputs are triggered, there exist multiple paths from the inputs

to outputs. Let $\mathscr{P}_1$ and $\mathscr{P}_2$ be the path from the input a and b to the output y, respectively. If two paths share a common path from the node v_i to the output, the number of transition caused by triggering the input a and b varies according to the node v_i and the arrival time at the node v_i. If the node v_i is a XOR gate and the difference between two arrival time at the node v_i, denoted by δ is greater than 0, then the glitch at the output of the XOR gate occurs and propagates to the output through the shared path. Otherwise, there exist no transitions from the node v_i. The probability that δ is equal to 0, denoted by $p_{\delta=0}$ can be obtained as follows :

$$\begin{aligned} p_{\delta=0} = \Pr[S_{n_1} = S_{n_2} = t] &= n\left(t; \sum_{j=1}^{n_1} \mu_{1j}, \sqrt{\sum_{j=1}^{n_1} \sigma_{1j}^2}\right) \\ &= n\left(t; \sum_{j=1}^{n_2} \mu_{2j}, \sqrt{\sum_{j=1}^{n_2} \sigma_{2j}^2}\right), \end{aligned}$$

where S_{n1} and S_{n2} are the time of the n_1th and n_2th renewal of each path from each input to the node v_i. The expected number of transitions $R(T)$ is equal to

$$\begin{aligned} &R_1(t) + R_2(t) + 2(1 - p_{\delta=0})R_3(T) = \\ &\sum_{i=1}^{n_1}\prod_{j=1}^{i} \mathbf{Ob}_{v_{1,i}}(v_{1,i-1}) + \sum_{i=1}^{n_2}\prod_{j=1}^{i} \mathbf{Ob}_{v_{2,i}}(v_{2,i-1}) + 2(1 - p_{\delta=0})\sum_{i=1}^{n_3}\prod_{j=1}^{i} \mathbf{Ob}_{v_{3,i}}(v_{3,i-1}) \end{aligned}$$

where $R_1(t)$ and $R_2(t)$ are the expected number of transitions on each path from each input to the node v_i. $R_3(T)$ is the expected number of transitions when only one switching event is injected due to one of the incoming paths. Note that the term $(1 - p_{\delta=0})$ captures the probability that glitching occurs.

Similarly, if the node v_i is a gate other than XOR, such as NAND or NOR, the term $2p_{\delta=0}R_3(T)$ is changed into $\frac{1}{2}p_{\delta=0}R_3(T) + (1 - p_{\delta=0})R_3(T)$. Fig. 3.3 shows the reason why the term should be changed based on the truth table of different gates.

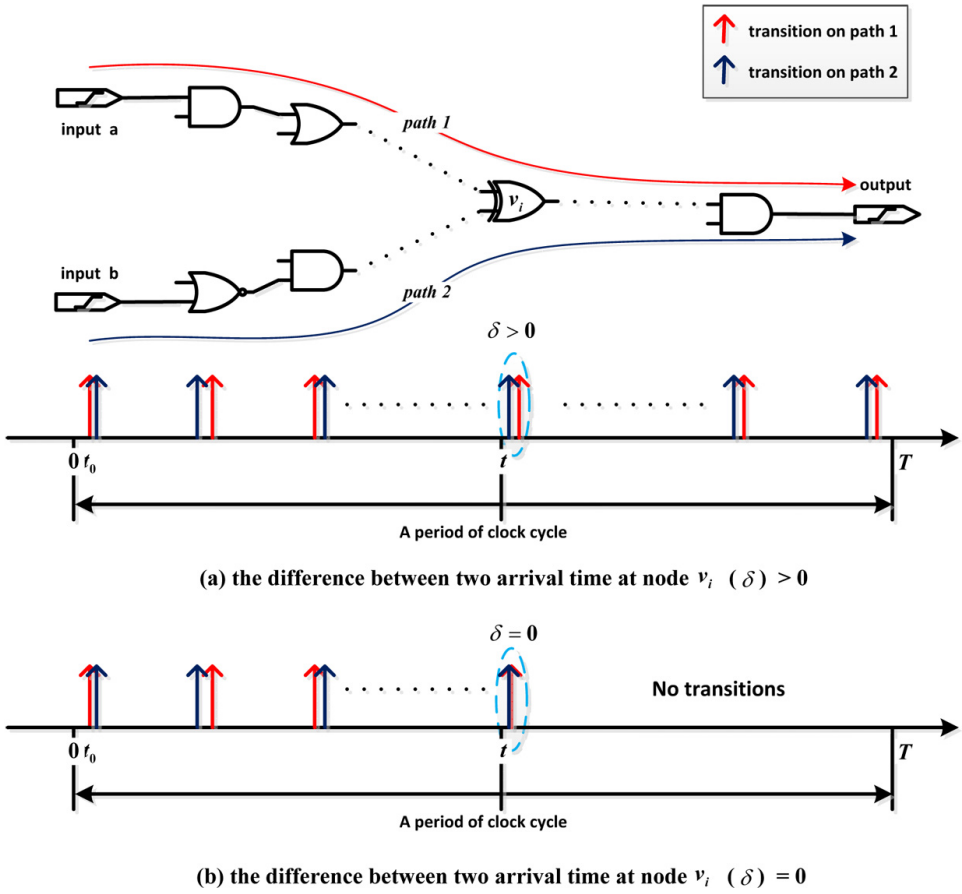


Figure 3.2: Renewal process caused by triggering two inputs

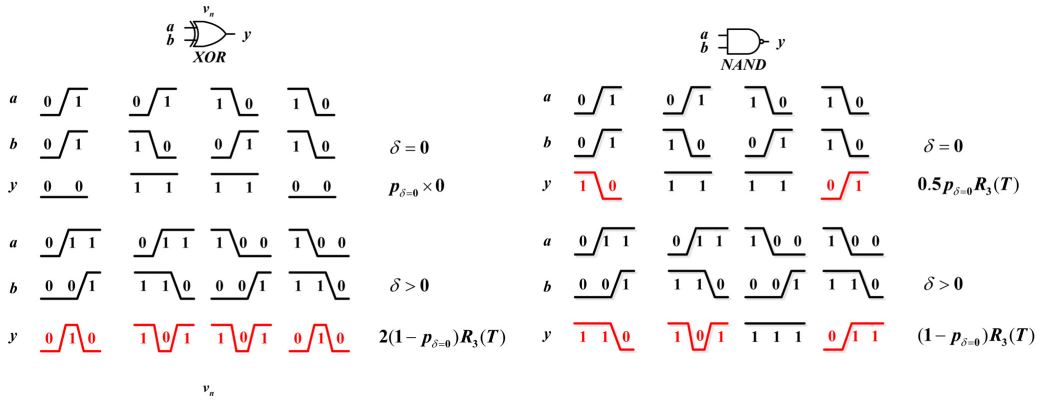


Figure 3.3: Different transition counts according to logic gate and δ

3.3.2 Graph based analysis

The logic network graph of the combinational logic circuit will be simplified through node collapsing using the properties in Eq. (3.2), (3.3), (3.6) and (3.7). This method is called *graph based analysis*. Let the corresponding logic network graph to be $G(V, E, W(E), W(V))$ with the edge weight set $W(E) = \{w((v_i, v_j))|w((v_i, v_j)) = \mathbf{Ob}_j(i), (v_i, v_j) \in E\}$ and the vertex weight vector set $W(V) = \{\vec{w}(v)|\vec{w}(v) = [w_i(v)], w_i(v) = C(v)\ for\ i = 0, \ldots, Indegree(v) - 1, v \in V\}$. For example, given a logic network graph of $y = g(a, b) = a \cdot b$, there exist four vertices v_a, v_b, v_g, v_y and three edges $(v_a, v_g), (v_b, v_g), (v_g, v_y)$. The components of W are the following:

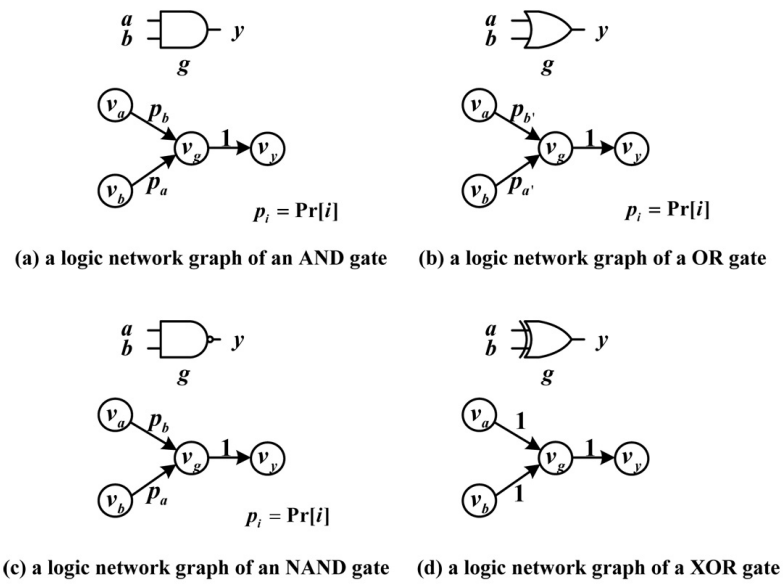
$$w((v_a, v_g)) = \mathbf{Ob}_g(a) = \Pr[f_a \oplus f_{a'}] = \Pr[b]$$
$$w((v_b, v_g)) = \mathbf{Ob}_g(b) = \Pr[f_b \oplus f_{b'}] = \Pr[a]$$
$$w((v_g, v_y)) = 1$$
$$\vec{w}(v_g) = [w_0(v_g)\ \ w_1(v_g)] = [C(g)\ \ C(g)].$$

The power consumption caused by switching an input is given by the following equations:

$$\mathbf{P}_y(a) = \alpha \cdot w((v_a, v_g)) \cdot w((v_g, v_y)) \cdot w_0(v_g) = \alpha \Pr[b] C(g)$$
$$\mathbf{P}_y(b) = \alpha \cdot w((v_b, v_g)) \cdot w((v_g, v_y)) \cdot w_1(v_g) = \alpha \Pr[a] C(g)$$

where α is $0.5 V_{DD}^2 f$. Other logic gates such as OR, NAND, NOR or XOR also correspond to a logic network graph. Fig. 3.4 shows these logic network graphs of basic logic gates. The logic network graph $G(V, E, W(E), W(V))$ can be simplified or reduced through node and edge reduction primitives by using Lemma 2 and Lemma 3.

Node Reduction: Two gate vertices v_{g1} and v_{g2} connected by an edge (v_{g1}, v_{g2}) can be united into a vertex $v_{g_1 g_2}$ with $Indegree(v_{g_1 g_2}) = Indegree(v_{g_1}) + Indegree(v_{g2}) - 1$ and $Outdegree(v_{g_1 g_2}) = Outdegree(v_{g_1}) + Outdegree(v_{g_2}) - 1$ after removing the edge (v_{g1}, v_{g2}). The weights of indegree edges of v_{g_1} are changed to $w((v_{g_1}, v_{g_2}))$ times their weights given by Eq. (3.2). The weights of incoming edges of v_{g_2} are not changed. The weight vectors of the united vertex $v_{g_1 g_2}$ are changed into $\vec{w}(v_{g_1 g_2}) = [w_i(v_{g_1 g_2})]$ where $w_i(v_{g_1 g_2}) = c(g_1)/w((v_{g_1}, v_{g_2})) + c(g_2)$ for $i = 0, \ldots, Indegree(v_{g_1}) - 1$ and $w_i(v_{g_1 g_2}) = c(g_2)$ for $i = Indegree(v_{g_1}), \ldots, Indegree(v_{g_1}) + Indegree(v_{g_2}) - 2$ by Eq. (3.6). This vertex reduction can be repeated until only the pairs of the



(a) a logic network graph of an AND gate (b) a logic network graph of a OR gate

(c) a logic network graph of an NAND gate (d) a logic network graph of a XOR gate

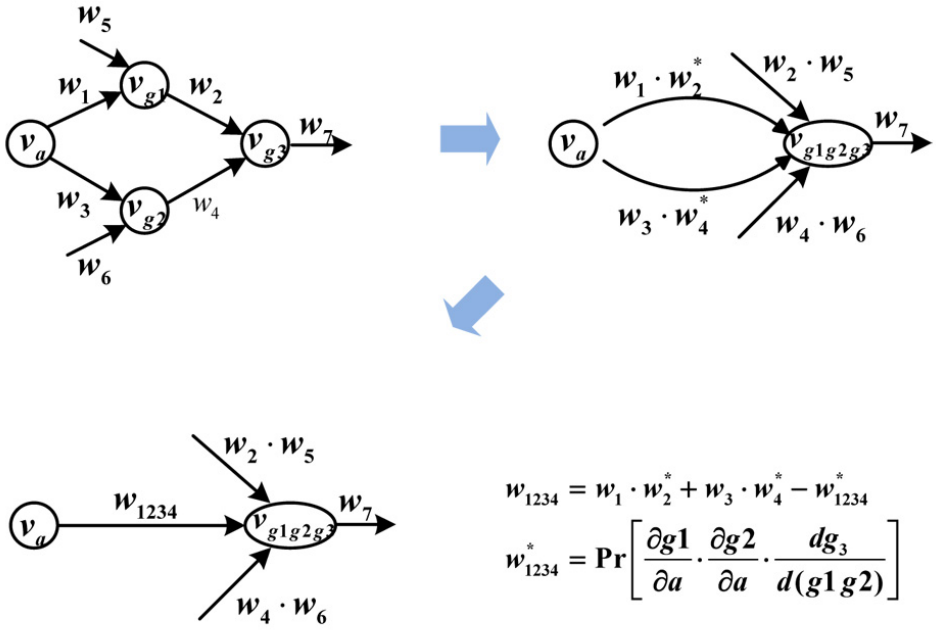
Figure 3.4: Logic network graphs of basic logic gates

reconvergent fanout v_{rf} and node v_{rn} with two or more edges between them remain, along with the primary input and output nodes. Two or more edges of the pairs of the reconvergent fanout and node can be reduced by the following edge reduction. Also, the pairs of the reconvergent fanout and node with a reduced edge can be reduced by this node reduction except that the weight of the vertex $v_{rf,rn}$ is derived by Eq. (3.7). Note that each node reduction step reduces the node count by at least 1.

Edge Reduction: The edges between the reconvergent fanout v_{rf} and node v_{rn} are reduced into a single edge with the weight $\mathbf{Ob}_{rn}(rf)$ given by Eq. (3.3). Finally, the simplified network graph $G'(V, E, W(E), W(V))$ has only a single logic (function) node, the primary input nodes, and the primary output node. Fig. 3.5 shows the vertex reduction in the logic network graph. Algorithm 2 presents the reduction method to reduce a logic graph into a singleton graph in order to compute the power consumption of the combinational circuit trivially.

If all effective capacitances in the logic network are set to 1, the expected number of switched signals on each path is equal to the weight of the corresponding edge in the simplified network.

(a) Reduction of two nodes



(b) Reduction of reconvegent paths

Figure 3.5: Reduction of Logic network graph

Algorithm 1 Reduction of $G(V, E, W(E), W(V))$

Input : A logic network graph, $G(V, E, W(E), W(V))$
$V = \{V_{pi}, V_{po}, V_g | \text{pi: privary inputs, po : privary outputs, g : local functions}\}$
$W(E) = \{w_{(v_i,v_j)} | w_{(v_i,v_j)} = \mathbf{Ob}_j(i), (v_i, v_j) \in E\}$
$W(V) = \{\vec{w}(v) | \vec{w}(v) = [w_i(v)], w_i(v) = C(v)$
$for\ \forall\ i = 0, \ldots, Indegree(v) - 1, v \in V\}$
Output : A simplified network graph,
$G'(V, E, W(E), W(V))$
for $k = 1 \rightarrow |V_g| - 1$ **do**
Select two connected vertices v_{gi}, v_{gj} $for\ \forall\ v_{gi}, v_{gj} \in V_g$
if v_{gi} = Reconvergent fanout and v_{gj} = Reconvergent node with two more edges **then**
Reduction edge (v_{gi}, v_{gj}), Reduction vertex (v_{gi}, v_{gj})
else
Reduction vertex (v_{gi}, v_{gj})
end if
end for

3.3.3 Linear regression

The number of transitions of signals in the hardware implementation is highly correlated to the dynamic power consumption [Mangard et al. (2005)]. In order to estimate power consumption using the number of transitions, linear regression is used. Let X and Y be the random variables of the number of transitions and power consumption, respectively. The estimator of Y, denoted by $\hat{Y}$ is the followings:

$$\hat{Y} = \hat{\alpha} + \hat{\beta}X, \quad \hat{\beta} = \frac{S_{xy}}{S_{xx}}, \quad \hat{\alpha} = \overline{Y} - \hat{\beta}\overline{X} \tag{3.9}$$

where $S_{xy} = \sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})$, $S_{xx} = \sum_{i=1}^{n}(x_i - \overline{X})^2$ and $\overline{X}$ and $\overline{Y}$ are the sample means of X and Y, respectively. Thus, the probability density function of the power can be refered to $n(x; \mu_{\hat{Y}}, \sigma_{\hat{Y}})$, where $\mu_{\hat{Y}} = \hat{Y}$ and $\sigma_{\hat{Y}} = \sqrt{\hat{\beta}\sigma_X^2}$ by a few number of samples. This leakage distribution will be used to induce power based SCA security metrics in the following section.

3.4 SCA Security Metrics

Power based SCA security metrics were defined [Basel Halak (2013)] in order to measure the effectiveness (inverse of robustness or resistance) of side-channel attacks on the target boolean

function : $\vec{v} = f(\vec{k}, \vec{x})$, where $\vec{k}$ is a part of the secret data and $\vec{x}$ is related to the plaintext or ciphertext. The more distinguishable and identifiable power consumption is to different inputs, the more vulnerable is the SCA security of the target boolean function. In order to quantify SCA effectiveness, the normalized standard deviation was used in [Basel Halak (2013)]. The normalized standard deviation is defined by the following equation :

$$\frac{\hat{\sigma}}{\hat{\mu}} = \frac{n\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}{\sqrt{n-1}\sum_{i=1}^{n} y_i}$$

where y_i is a random sample of power consumption given any input pattern from the sample space with the mean μ and the variance σ^2, $\overline{y}$ is the sample mean of y_i. As n goes to infinity, the normalized standard deviation is equal to σ/μ.

Note that if we allow constant current components in the circuit, this metric is flawed. Given a circuit C_0 with mean μ_0 and standard deviation σ_0, the metric is altered from (σ_0/μ_0) to $(\sigma_0/(\mu_0+\mu_1))$ when another isolated, disconnected circuit C_1 with constant current ($\sigma_1 = 0$ and mean μ_1) is added. This indicates a quantitative reduction in the SCA effectiveness for no good reason. Also, this metric has large value for countermeasure circuits with randomly independent power consumption even if the circuits have robustness against SCA attacks. Additionally, there is no obviously justifiable mechanism to determine a safety threshold for this metric to flag a circuit as vulnerable when the metric exceeds the threshold. For these reasons, we propose a new SCA security metric using Kullback-Leibler divergence in the following subsection.

3.4.1 Kullback-Leibler divergence

Let's consider the failure probability that the adversary makes an incorrect inference using the standard power based SCA attacks. A circuit with high failure probability should be more secure than the circuit with low failure probability. First, we assume that the adversary wants to know only an output bit Y by SCA attacks. We also assume that $\Pr[Y = 0] = \Pr[Y = 1]$ as a starting point to define the new SCA security metric. Let $\Pr[l|y_0]$ be the probability density function of the power leakage given that the output Y is 0 and $\Pr[l|y_1]$ be the probability density function of the power leakage given that the output Y is 1. Suppose that the conditional probability density functions are normal distributions with the means μ_0, μ_1 (assuming that

$\mu_1 > \mu_0$) and the same variation σ_0^2. Assuming that the adversary knows the conditional probability density functions, s/he can estimate the output Y by the following hypothesis test :

$$\begin{aligned}
\Pr[y_0|l] &\underset{H_1}{\overset{H_0}{\gtrless}} \Pr[y_1|l], \\
\Pr[l|y_0]\Pr[y_0] &\underset{H_1}{\overset{H_0}{\gtrless}} \Pr[l|y_1]\Pr[y_1] \\
\frac{\Pr[l|y_0]}{\Pr[l|y_1]} &\underset{H_1}{\overset{H_0}{\gtrless}} \frac{\Pr[y_1]}{\Pr[y_0]} = 1.
\end{aligned}$$

The adversary should choose 0 output if the a posteriori probability $\Pr[y_0|l]$ is greater than the a posteriori probability $\Pr[y_1|l]$. Otherwise, s/he should choose 1.

The failing probability of the adversary is defined as the sum of the probability that given the output 0, the hypothesis test H_1 is selected and the probability that given the output 1, the hypothesis test H_0 is selected. That is

$$\begin{aligned}
\Pr_F &= \Pr[H_0|y_1] + \Pr[H_1|y_0] \\
&= 2\int_{\frac{\mu_0+\mu_1}{2}}^{\infty} \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_0}{\sigma_0}\right)^2\right] dx \\
&= 2\int_{\frac{\mu_1-\mu_0}{2\sigma_0}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \\
&= 2Q\left(\frac{\mu_1-\mu_0}{2\sigma_0}\right)
\end{aligned} \tag{3.10}$$

where $Q(x)$ is called the complementary error function. SCA security metric using the normalized standard deviation $\frac{\sigma}{\mu}$ is the following:

$$\begin{aligned}
\mu &= \frac{\mu_0+\mu_1}{2} \\
\sigma &= \sqrt{\frac{\left(\mu_0 - \frac{\mu_0+\mu_1}{2}\right)^2 + \left(\mu_1 - \frac{\mu_0+\mu_1}{2}\right)^2}{2}} = \frac{\mu_1-\mu_0}{2} \\
\frac{\sigma}{\mu} &= \frac{\mu_1-\mu_0}{\mu_0+\mu_1}
\end{aligned} \tag{3.11}$$

Comparing Eq.(3.10) with Eq.(3.11), earlier SCA security metric for normalized variance does not match with the failing probability (μ is not required). In this case, σ/σ_0 is a better choice as the SCA security metric. For example, if the circuit designer wants the SCA failing probability

to beat least 0.9, the SCA security metric should be less than 0.12 by the following equations:

$$\Pr_F = 2Q\left(\frac{\mu_1 - \mu_0}{2\sigma_0}\right) = 0.9$$
$$\frac{\mu_1 - \mu_0}{2\sigma_0} = 0.12 = \frac{\sigma}{\sigma_0}.$$

3.4.1.1 Two normal distributions with different means and variances

Suppose that two conditional probability density functions $\Pr[l|y_0]$ and $\Pr[l|y_1]$ have different means and variances ($\mu_0 \neq \mu_1, \sigma_0 \neq \sigma_1$). In this case, the failing probability $\Pr_F$ is equal to the shaded area in Fig. 3.6 called overlapping coefficient.

$$\begin{aligned}\Pr_F &= \int_{-\infty}^{\infty} min(\Pr[x|y_0], \Pr[x|y_1])dx \\ &= \int_{\alpha}^{\infty} n(x;\mu_0,\sigma_0)dx + \int_{-\infty}^{\alpha} n(x;\mu_1,\sigma_1)dx \end{aligned} \tag{3.12}$$

where $\alpha = \frac{(\sigma_1^2\mu_0 - \sigma_0^2\mu_1) + \sigma_0\sigma_1\sqrt{(\mu_1-\mu_0)^2 + 2\ln\frac{\sigma_0}{\sigma_1}(\sigma_0^2-\sigma_1^2)}}{\sigma_1^2-\sigma_0^2}$. The above equation cannot be simplified such as Eq.(3.10) and also it is difficult to obtain the exact value. In order to get simple and approximate value of $\Pr_F$ in general cases, we assume that each conditional probability density faction has the same variance σ_0 at the following subsection.

3.4.1.2 N normal distributions with the same variance σ_0

The power consumptions of any circuit can be classified as N normal distribution with $\mu_0, \mu_1, \ldots, \mu_{N-1}$ and the same σ_0^2 based on the number of outputs. The failing probability $\Pr_F$ of the adversary is equal to the overlapping coefficient of N normal distributions. The $\Pr_F$ is larger than the smallest overlapping coefficient between two normal distributions of N normal distributions. The two normal distributions have the smallest mean and largest mean, respectively. The smallest overlapping coefficient is selected as the threshold of the failing probability denoted as $\Pr_{Fth}$. If the designer set $\Pr_{Fth}$ to any value, the failing probability should be larger than the value. In this case, the threshold of the failing probability and SCA

security metric are the following equations:

$$\Pr_{Fth} = 2Q\left(\frac{\sup_{i\neq j}|\mu_i - \mu_j|}{2\sigma_0}\right)$$

$$SCA\ security\ metric = \frac{\sup_{i\neq j}|\mu_i - \mu_j|}{2\sigma_0}$$

Generally, N normal distributions have different means and variances. In the general case, it is difficult to compute the threshold of the failure probability and define SCA security metric. Kullback-Leibler divergence is used to define new SCA security metric.

3.4.1.3 Kullback-Leibler divergence

Let $f_X(z)$ and $f_Y(x)$ be the probability density functions of random variable X and Y, respectively. Kullback-Leibler divergence is defined as the following equation [S. Kullback and R. A. Leibler (1951)]:

$$D_{KL}(X||Y) = \int f_X(z)\log\frac{f_X(z)}{f_Y(z)}dz$$

If X and Y are the normal distribution with μ_0, σ_0^2 and μ_1, σ_1^2, respectively, then

$$\begin{aligned} D_{KL}(X||Y) &= \int n(x;\mu_0,\sigma_0)(\log n(x;\mu_0,\sigma_0) - \log n(x;\mu_1,\sigma_1))dx \\ &= \left\{(\mu_0-\mu_1)^2 + \sigma_0^2 - \sigma_1^2\right\}/(2\sigma_1^2) + \ln(\sigma_1/\sigma_0) \end{aligned} \tag{3.13}$$

Kullback-Leibler divergence of two random variables with the normal distribution can be computed easily. The maximum of Kullback-Leibler divergence for allowable failure probability can be obtained. For example, if we want the failure probability of more than 0.9, the Kullback-Leibler divergence should be less than 0.03.

Also, Kullback-Leibler divergence is related to the number of traces N that is necessary to assert with a confidence of $(1-\alpha)$ that the two normal distributions X and Y are different. The number of traces N is a significant contributor in quantifying a lower bound on the attack complexity. The smallest number of traces to satisfy that $\Pr\left[|\overline{X} - \overline{Y} - (\mu_X - \mu_Y)| < \epsilon\right] = (1-\alpha)$ is

$$N \geq \frac{(\sigma_0+\sigma_1)^2}{\epsilon^2(\mu_0-\mu_1)^2}\cdot z_{1-\alpha/2}^2$$

where the quantile $z_{1-\alpha/2}$ of the standard normal distribution has the property that $\Pr\left[Z \leq z_{1-\alpha/2}\right] = 1 - \alpha/2$. Comparing to Eq. (3.13), as Kullback-Leibler divergence of two random variables increases, the number of traces N decreases. Ideally, we would like to be able to show that N has a non-trivial, super polynomial lower bound in n - the number of bits in the secret.

3.4.1.4 SCA security metric using Kullback-Leibler divergence

Generally, suppose that the adversary knows N normal probability density functions with different means and variances. We define SCA security metric as Maximum Kullback-Leibler divergence of two random variables among N random variables:

$$SCA\ security\ metric = \underset{\substack{X_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \\ X_j \sim \mathcal{N}(\mu_j, \sigma_j^2) \\ 0 \leq i,j \leq N}}{\text{MAX}} \{D_{KL}(X_i || X_j)\} \tag{3.14}$$

3.4.2 Mutual information

The second security metric to quantify DPA effectiveness is to use the mutual information [Standaert et al. (2009)].

$$I(\vec{K}; \vec{L}) = H[\vec{K}] - H[\vec{K}|\vec{L}] \tag{3.15}$$

where $\vec{K}$ is a variable containing a part of the secret data and $\vec{L}$ is a leakage observation such as power consumption through the side channel. The entropy of $\vec{K}$, denoted by $H[\vec{K}]$ is $log_2|\vec{K}|$ assuming that $\vec{K}$ is uniformly distributed. The conditional entropy $H[\vec{K}|\vec{L}]$ is the following equation :

$$H[\vec{K}|\vec{L}] = -\sum_k \Pr[k] \int \Pr[l|k] \cdot log_2 \Pr[k|l] dl$$
$$where\ \ \Pr[k|l] = \frac{\Pr[l|k]\Pr[k]}{\sum_{k^* \in \vec{K}} \Pr[l|k^*]\Pr[k^*]}.$$

In order to compute the mutual information, the conditional probability $\Pr[l|k]$ should be estimated. Using simulation tools such as SPICE, the power consumptions can be measured

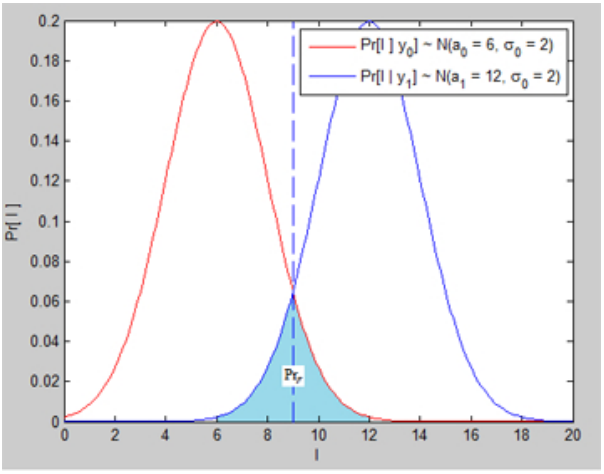


Figure 3.6: The failure probability $\Pr_F$: Overlapping coefficient of two normal distributions

resulting in a sampled estimate of the probability distribution. Since the simulation-based power estimation requires significant time due to detailed circuit level SPICE simulation, it is important to determine the required minimum number of sample measurements to obtain statistically significant probability distribution. Assuming the probability distribution is the normal distribution with the mean μ and the variance σ^2, the smallest number of measurements to satisfy that $\Pr\left[\frac{\overline{L}-\mu}{\sigma\sqrt{N}} < \epsilon\right] = 1-\alpha$ is $N = \frac{\sigma^2}{\epsilon^2} \cdot z^2_{1-\alpha/2}$. The mutual information based SCA analysis will be exploited for more realistic and accurate verification at the physical transistor or layout level.

3.5 Recognition Rate Using Maximum Likelihood Estimation

The maximum likelihood estimator of c is defined as the following:

$$\hat{c} = \arg\max_{c_i \in C} T_{c_i} = \arg\max_{c_i \in C} \frac{1}{n} \sum_{m=1}^{n} \ln f_{L|c_i}(\vec{l_m})$$

where T_{c_i} is the test statistic for the class c_i and $f_{L|c_i}$ is the probability density function of the side-channel leakage L given a class c_i. It requires the log-likelihood of the correct class c^* be larger than all other classes for the MLE to successfully recognize the side-channel leakage l into the correct class c^*. The successful recognition rate is defined as the probability that the

test statistic for the correct class c^*, T_{c^*} is larger than all $\{T_{\{c\}-c^*}\}$ [Fei et al. (2014)] :

$$SR = \Pr[T_{c^*} > \{T_{\{c\}-c^*}\}] \tag{3.16}$$

We first consider the recoginition rate when there exist two classes such as c_1, c_2. Assuming that c_1 is the correct class c^*, the successful recongnition rate SR is equal to the following:

$$SR = \Pr[T_{c_1} > T_{c_2}] = \Pr[T_{c_1} - T_{c_2} > 0] = \Pr[\Delta_{c_1,c_2} > 0]$$

where

$$\Delta_{c_1,c_2} = T_{c_1} - T_{c_2} = \frac{1}{n}\sum_{m=1}^{n}[\ln f_{L|c_1}(\vec{l_m}) - \ln f_{L|c_2}(\vec{l_m})].$$

In general, the disassembler exploits only one leakage observation $\vec{l_1}$. For a leakage observation, the mean and variance of Δ_{c_1,c_2} are given by the followings:

$$\mu_{\Delta_{c_1,c_2}} = \mathbf{E}_{L|c_1}[\ln f_{L|c_1}(\vec{l_1}) - \ln f_{L|c_2}(\vec{l_1})] \tag{3.17}$$

$$\sigma^2_{\Delta_{c_1,c_2}} = \mathbf{Var}_{L|c_1}[\ln f_{L|c_1}(\vec{l_1}) - \ln f_{L|c_2}(\vec{l_1})]. \tag{3.18}$$

Definition 6 (Noncentral chi-square distribution)**.** *The random variable Y is said to have a noncetral chi-square distribution [Mathai and Provost (1992)] with k degrees of freedom and noncentrality parameter δ if Y has the density*

$$f_Y(y;k,\delta) = e^{-\delta}\sum_{r=0}^{\infty}\frac{\delta^r}{r!}\frac{y^{\frac{k}{2}+r-1}e^{-y/2}}{2^{\frac{k}{2}+r}\Gamma\left(\frac{k}{2}+r\right)}, \tag{3.19}$$

where $0 < x < \infty, k = 1, 2, \ldots, \delta \geq 0$, which is denoted as $Y \sim \chi^2_{k,\delta}$.

Theorem 4. *If we assume that $f_{L|c_i}$ is the normal density function with the mean μ_{c_i} and the variance σ_{c_i}, then Δ_{c_1,c_2} has the linear transformed noncentral chi-square distribution with one degree of freedom, $\chi^2_{1,\delta}$, where $\delta = \left(\frac{(\mu_1-\mu_2)\sigma_1}{\sigma_1^2-\sigma_2^2}\right)^2$. The successful recognition rate is equal to the following:*

$$SR = \mid \frac{1}{a} \mid \left[1 - F_Y\left(\frac{b^2-4ac}{4a^2}\right)\right] \tag{3.20}$$

where $F_Y(y)$ is the cumulative density function of $\chi^2_{1,\delta}, a = \frac{\sigma_1^2-\sigma_2^2}{2\sigma_2^2}, b = \frac{(\mu_1-\mu_2)\sigma_1}{\sigma_2^2}$ and $c = \frac{(\mu_1-\mu_2)^2}{2\sigma_2^2} + \ln\frac{\sigma_2}{\sigma_1}$.

Proof.

$$\begin{aligned}\Delta_{c_1,c_2} &= \ln \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{-\frac{(l-\mu_1)^2}{2\sigma_1^2}\right\} - \ln \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left\{-\frac{(l-\mu_2)^2}{2\sigma_2^2}\right\} \\ &= -\frac{(l-\mu_1)^2}{2\sigma_1^2} + \frac{(l-\mu_2)^2}{2\sigma_2^2} + \ln \frac{\sigma_2}{\sigma_1}\end{aligned}$$

Let $x = \frac{l-\mu_1}{\sigma_1}$,

$$\begin{aligned}\Delta_{c_1,c_2} &= -\frac{1}{2}x^2 + \frac{\{\sigma_1 x + (\mu_1-\mu_2)\}^2}{2\sigma_2^2} + \ln \frac{\sigma_2}{\sigma_1} \\ &= \left(\frac{\sigma_1^2-\sigma_2^2}{2\sigma_2^2}\right)x^2 + \frac{(\mu_1-\mu_2)\sigma_1}{\sigma_2^2}x + \frac{(\mu_1-\mu_2)^2}{2\sigma_2^2} + \ln \frac{\sigma_2}{\sigma_1} \\ &= ax^2 + bx + c \\ &= a\left(x + \frac{b}{2a}\right)^2 + c - \frac{b^2}{4a}\end{aligned}$$

$$\left(a = \frac{\sigma_1^2-\sigma_2^2}{2\sigma_2^2}, \quad b = \frac{(\mu_1-\mu_2)\sigma_1}{\sigma_2^2}, \quad c = \frac{(\mu_1-\mu_2)^2}{2\sigma_2^2} + \ln \frac{\sigma_2}{\sigma_1}\right)$$

where l is the realization of a random variable L which has the normal distribution with the mean μ_1 and the variance σ_1^2, x is the realization of a random variable $X = \frac{L-\mu_1}{\sigma_1}$. Let $Z = aY + c - \frac{b^2}{4a}$, where $Y = \left(X + \frac{b}{2a}\right)^2$. The probability density function of Y is given by $f_Y(y;k,\delta)$ which is the noncentral chi-square density function with the $k=1$ degree of freedom and the noncentrality parameter $\delta = \frac{b^2}{4a^2}$. By the transformation technique, the probability density function of Z is induced by $f_Z(z) = f_Y(w(z)) \mid w'(z) \mid, w(z) = \frac{z}{a} - \frac{c}{a} + \frac{b^2}{4a^2}$.

$$\begin{aligned}f_Z(z) &= f_Y\left(\frac{z}{a} - \frac{c}{a} + \frac{b^2}{4a^2}; k=1, \delta = \frac{b^2}{4a^2}\right) \mid \frac{1}{a} \mid \\ &= f_Y\left(\frac{1}{a}\left(z - \frac{4ac-b^2}{4a}\right); k=1, \delta = \frac{b^2}{4a^2}\right) \mid \frac{1}{a} \mid\end{aligned}$$

$$\lambda = \left(\frac{(\mu_1-\mu_2)\sigma_1}{\sigma_1^2-\sigma_2^2}\right)^2$$

$$\begin{aligned}SR = \Pr[z>0] &= \int_0^\infty f_Z(z)dz \\ &= \int_0^\infty \mid \frac{1}{a} \mid f_Y\left(\frac{1}{a}\left(z - \frac{4ac-b^2}{4a}\right); k=1, \delta = \frac{b^2}{4a^2}\right) dz \\ &= \mid \frac{1}{a} \mid \left[1 - F_Y\left(\frac{b^2-4ac}{4a^2}\right)\right].\end{aligned}$$

□

Theorem 5. *If we assume that $f_{L|c_i}$ is the D-dimensional normal density function with the mean $\vec{\mu_{c_i}}$ and the variance Σ_{c_i}, then Δ_{c_1,c_2} has the linear transformed noncentral chi-square distribution with the D degree of freedom, $\chi^2_{D,\delta}$.*

Proof.

$$
\begin{aligned}
\Delta_{c_1,c_2} &= \ln \frac{1}{(2\pi)^{D/2}\det(\Sigma_1)^{1/2}} \exp\left\{-\frac{1}{2}(\vec{l}-\vec{\mu_1})^T\Sigma_1^{-1}(\vec{l}-\vec{\mu_1})\right\} \\
&\quad - \ln \frac{1}{(2\pi)^{D/2}\det(\Sigma_2)^{1/2}} \exp\left\{-\frac{1}{2}(\vec{l}-\vec{\mu_2})^T\Sigma_1^{-1}(\vec{l}-\vec{\mu_2})\right\} \\
&= -\frac{1}{2}(\vec{l}-\vec{\mu_1})^T\Sigma_1^{-1}(\vec{l}-\vec{\mu_1}) + \frac{1}{2}(\vec{l}-\vec{\mu_2})^T\Sigma_2^{-1}(\vec{l}-\vec{\mu_2}) + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= -\frac{1}{2}\vec{Z}^T\vec{Z} + \frac{1}{2}(\vec{Z}+\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2}))^T\Sigma_1^{1/2}\Sigma_2^{-1}\Sigma_1^{1/2}(\vec{Z}+\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2})) \\
&\quad + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= -\frac{1}{2}\vec{Z}^TPP^T\vec{Z} + \frac{1}{2}(\vec{Z}+\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2}))^TP\Lambda P^T(\vec{Z}+\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2})) \\
&\quad + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= -\frac{1}{2}(P^T\vec{Z})^T(P^T\vec{Z}) + \frac{1}{2}(P^T\vec{Z}+P^T\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2}))^T\Lambda(P^T\vec{Z}+P^T\Sigma_1^{-1}(\vec{\mu_1}-\vec{\mu_2})) \\
&\quad + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= -\frac{1}{2}\sum_{i=1}^{D}u_i^2 + \frac{1}{2}\sum_{i=1}^{D}\lambda_i(u_i+b_i)^2 + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= \frac{1}{2}\sum_{i=1}^{D}\left((\lambda_i-1)u_i^2 + 2\lambda_ib_iu_i + \lambda_ib_i^2\right) + \ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}} \\
&= \sum_{i=1}^{D}\alpha_i\left(u_i+\frac{\beta_i}{2\alpha_i}\right)^2 + \gamma_i - \frac{\beta_i^2}{4\alpha_i^2} \\
&\quad \left(\alpha_i = \frac{\lambda_i-1}{2}, \beta_i = \lambda_ib_i, \gamma_i = \frac{\lambda_ib_i}{2} + \frac{1}{D}\ln\frac{\det(\Sigma_2)^{1/2}}{\det(\Sigma_1)^{1/2}}\right)
\end{aligned}
$$

where $f_x(;k,\delta)$ is the noncentral chi-square density function with the k degree of freedom and the noncentrality parameter δ, $\Lambda = diag(\lambda_1,\lambda_2,\ldots,\lambda_n)$, λ_i is the eigenvalue of $\Sigma_1^{1/2}\Sigma_2^{-1}\Sigma_1^{1/2}$,

$P = [\vec{p_1}, \vec{p_2}, \ldots, \vec{p_n}], PP^T = I, \vec{p_i}$ is the eigenvector corresponding to λ_i, $\vec{U} = P^T\vec{Z}$, $\mathbf{E}[\vec{Z}] = \vec{0}, \mathbf{Cov}[\vec{Z}] = I, \mathbf{E}[\vec{U}] = \vec{0}, \mathbf{Cov}[\vec{U}] = I$, and $\vec{b} = P^T\Sigma_1^{-1}(\vec{\mu_1} - \vec{\mu_2})$. □

Theorem 6. *If there exist three classes c_1, c_2, c_3 and the correct class of a sample is c_1, the range of successful recognition rate is the following :*

$$\min\{\Pr[\Delta_{c_1,c_2} > 0], \Pr[\Delta_{c_1,c_3} > 0]\} \leq SR \leq \max\{\Pr[\Delta_{c_1,c_2} > 0], \Pr[\Delta_{c_1,c_3} > 0]\}$$

Proof. Let T_{c_i} be the test statistic for the class c_i : $T_{c_i} = \ln f_{L|c_i}(\vec{l_m})$. Since we assume that the sample belongs to the class c_1, the successful recognition rate SR is equal to $\Pr[\max\{T_{c_1}, T_{c_2}, T_{c_3}\} = T_{c_1}]$ or $\Pr[T_{c_1} > T_{c_2}, T_{c_1} > T_{c_3}]$.

Given that $T_{c_2} > T_{c_3}$, $\Pr[\max\{T_{c_1}, T_{c_2}, T_{c_3}\} = T_{c_1}] = \Pr[T_{c_1} > T_{c_2}]$. Otherwise, that is, given that $T_{c_3} \geq T_{c_2}$, $\Pr[\max\{T_{c_1}, T_{c_2}, T_{c_3}\} = T_{c_1}] = \Pr[T_{c_1} > T_{c_3}]$. Thus, the successful recognition rate is equal to the following:

$$\begin{aligned} SR &= \Pr[T_{c_2} > T_{c_3}]\Pr[T_{c_1} > T_{c_2}] + \Pr[T_{c_3} \geq T_{c_2}]\Pr[T_{c_1} > T_{c_3}] \\ &= \alpha\Pr[T_{c_1} > T_{c_2}] + (1-\alpha)\Pr[T_{c_1} > T_{c_3}] \\ &= \{\Pr[T_{c_1} > T_{c_2}] - \Pr[T_{c_1} > T_{c_3}]\}\,\alpha + \Pr[T_{c_1} > T_{c_3}] \end{aligned}$$

where α is $\Pr[T_{c_2} > T_{c_3}]$.

If $\Pr[T_{c_1} > T_{c_2}] > \Pr[T_{c_1} > T_{c_3}]$, SR has the minimum of $\Pr[T_{c_1} > T_{c_3}]$ at $\alpha = 0$ and the maximum of $\Pr[T_{c_1} > T_{c_2}]$ at $\alpha = 1$. If $\Pr[T_{c_1} > T_{c_2}] < \Pr[T_{c_1} > T_{c_3}]$, SR has the minimum of $\Pr[T_{c_1} > T_{c_2}]$ at $\alpha = 1$ and the maximum of $\Pr[T_{c_1} > T_{c_3}]$ at $\alpha = 0$. Thus, the range of SR is between $\Pr[T_{c_1} > T_{c_2}]$ and $\Pr[T_{c_1} > T_{c_3}]$. Fig. 3.7 shows the successful recognition rate according to α in the both cases. □

3.6 Experiment

We implemented AES SBOX based on composite finite field proposed by Satoh *et al.* [Satoh et al. (2001)] to compute our SCA security metrics. We used Cadence *RTL Compiler* and

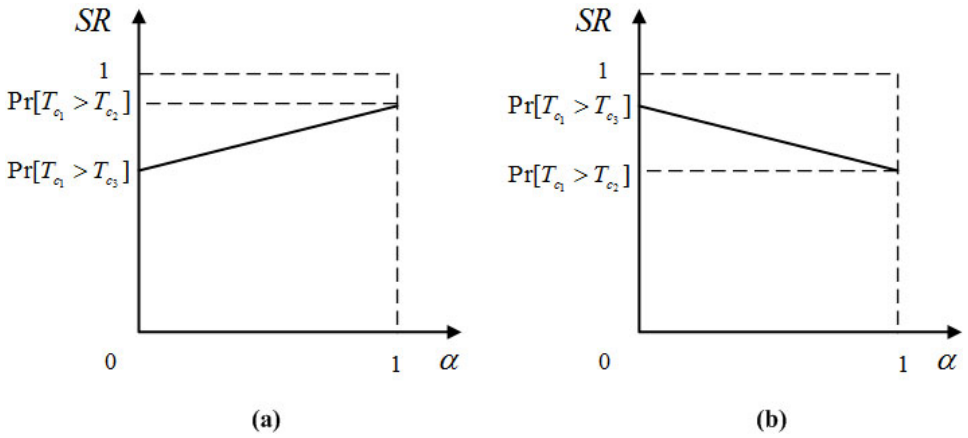


Figure 3.7: Successful recognition rate according to α (a) when $\Pr[T_{c_1} > T_{c_2}] > \Pr[T_{c_1} > T_{c_2}]$ *(b) when* $\Pr[T_{c_1} > T_{c_3}] > \Pr[T_{c_1} > T_{c_2}]$

OSU standard cell library based on AMI C5N 0.6 μ process as the logic synthesizer and the technology library, respectively. The calculator of the expected number of transitions depending on triggered input bits is programmed with Perl/Tk. It generates the logic network graph with the synthesized netlist and searches all paths from triggered inputs to outputs. Shared paths of all paths are split and the number of transitions on those paths is calculated differently depending on the staring node of the shared path and the different between arrival times at the node. The sum of the number of transitions on each path, denoted by $R(T)$ is the total number of transitions in the SBOX during computation.

In order to compute coefficients of Eq.(3.9), 1000 random pairs of (x_i, y_i) (which represent the number of transitions and average power during computation, respectively) are sampled using Cadence *Spectre* analog simulator for sampling y_i and transition counter for sampling x_i. Fig. 3.8 shows scattered plots of 1000 sample pairs and the linear regression line. In this case, $\hat{\beta}$ and $\hat{\alpha}$ are computed as 0.085 and 1.05, respectively. That is, the mean of estimated power of any input vector, $\mu_{\hat{Y}}$ is equal to $\hat{\alpha} + \hat{\beta}R(T)$ and the variance $\sigma^2_{\hat{Y}}$ is $\hat{\beta}\sigma^2_R$. The probability density function of the estimated power has the normal distribution with the mean $\mu_{\hat{Y}}$ and the variance $\sigma^2_{\hat{Y}}$. 256 normal distributions which results from all possible input vector (2^8) can be obtained by 1000 times simulations and renewal process based estimation. SCA security metric using KL divergence is 3.72 which corresponds to about 17% failure probability.

Using simulated samples, correlation power analysis attack of this SBOX was executed. We

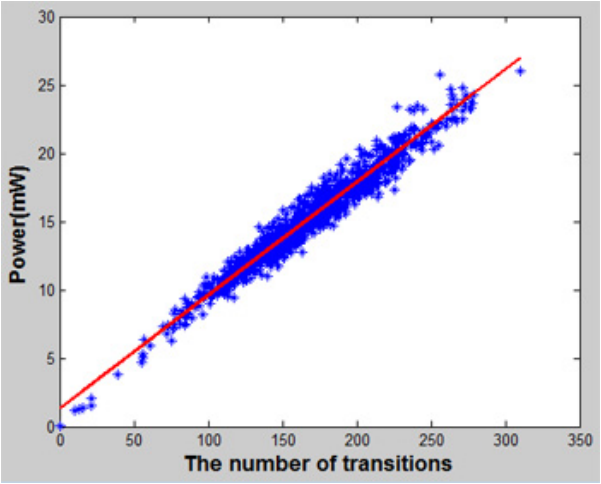


Figure 3.8: Scattered plots and linear regression ($\hat{\beta} = 0.085, \hat{\alpha} = 1.05$) of 1000 random samples

assume that the correct key value is 19. The correlation coefficient ρ of 19 guess key has the highest value (0.53) and the guess key is correct. Also, the success probability was measured depending on the number of samples (N). The success probability is over 95% when N is more than 220 samples. Fig. 3.12 shows the result of CPA attack. Thus, this SBOX should be protected against power based SCA attacks.

3.7 Conclusion

We have developed (1) a quantitative metric to capture the SCA resistance of a combinational circuit, (2) developed and implemented a stochastic power estimation method using renewal process and linear regression which is more efficient than simulation based method. As an example, we applied our metric and estimation method to AES SBOX implementation at the logic level. We will apply these techniques to many unprotected and protected cryptographic implementations and develop secure implementations with $D_{KL} < 0.03$ (which means the threshold of the failing probablity is greater than 90%).

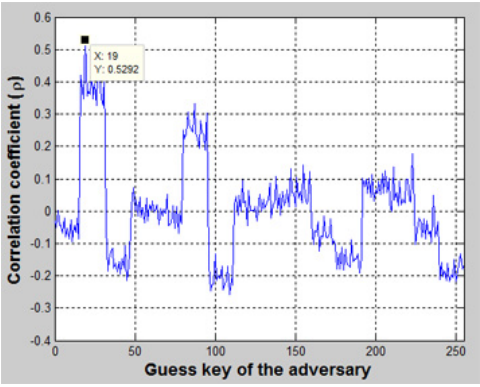


Figure 3.10 Correlation Power Analysis attack of AES SBOX (N = 1000)

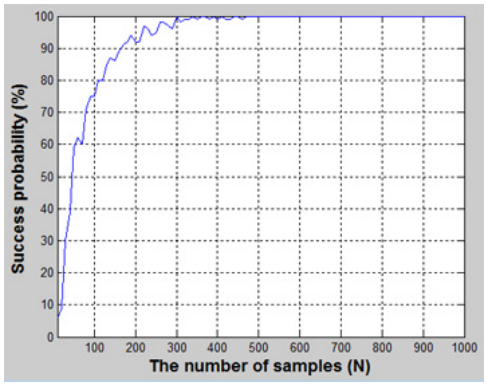


Figure 3.11 Success probability according to the number of samples (N)

Figure 3.12: CPA attack of AES SBOX

CHAPTER 4. SECURE LOGIC STYLE

4.1 Introduction

In order to remove dependency between power consumption and intermediate values of the executed cryptographic algorithm, the cryptographic hardware can be implemented with secure primitive logic cells such as Sense Amplified Based Logic (SABL) [Tiri et al. (2002)], Wave Dynamic Differential Logic (WDDL) [Tiri and Verbauwhede (2005)] and t-private logic circuit [Ishai et al. (2003)]. These secure logic style have the different method to make independent power consumption of the performed operation and the processed data values. SABL and WDDL consume equal amounts of power consumption in each clock cycle, but on the other hand, t-private logic circuit randomizes amounts of power consumption in each clock cycle. In other words, SABL and WDDL implement the hiding countermeasure and t-private logic circuit implements the masking countermeasure.

Also, all these secure cells have robustness against side-channel attacks but only t-private logic circuit prevents from the probing attack by which an adversary can observe only t-limited number of internal nodes per each clock cycle. In a view of the design implementation, t-private logic circuit and WDDL are implemented with the general CMOS digital cell library but each SABL cell should be full-customized. The area of t-private logic circuit has the largest among these secure logic style but the power consumption of t-private logic circuit is the smallest. Since SABL and WDDL have two phase (the precharge phase and the evaluation phase) during each clock cycle in which phase signals are switched, the power consumption of SABL and WDDL has larger value than the power consumption of t-private logic circuit. Table 4.1 shows the summary of these secure logic style.

Oklahoma State University (OSU) digital cell library based on the FreePDK45 techonalogy

library is exploited to implement the secure logic style. For the logic synthesis and physical layout, commercial EDA tools such as Cadence's tools and Synopsys's tools are used. Our logic cell library consists of implemented secure logic cells, OSU digital cells and FreePDK45 analog cells. This cell library defines the cell function, area, delay and power dissipation as the liberty file format.

This chapter is organized as follows. Section 4.2 presents sense amplifier based logic (SABL). Section 4.3 describes wave differential dynamic logic (WDDL). t-private logic circuits are presented in Section 4.4. Section 4.5 presents implementation of secure logic style. Finally, Section 4.6 summaries this chapter.

4.2 Sense Amplifed Based Logic (SABL)

Sense Amplifier Based Logic has been introduced by Tiri *et al.* [Tiri et al. (2002), Mangard et al. (2007)]. SABLs are specially designed to have a constant internal power consumption independent of the proposed logic values. SABLs are implemented as dual-rail precharge logic styles which means that each input is encoded as the pair of wires consisting of the original signal and inverted signal and all logic signals alternate between precharge values and evaluated values. In the precharge phase, the values of the complementary wires are set to the precharge value. During the evaluation phase, the values on the complementary wires are set to $(0, 1)$ or $(1, 0)$ according to the the processed data. Assuming that the precharge value is 0 and that the half of the clock cycle corresponds to the evaluation phase, one complementary output should

Table 4.1: Secure logic style

	SABL	WDDL	t-private logic
SCA resistance	✓	✓	✓
Probing resistance	×	×	✓
Method	Hiding	Hiding	Random masking
Design	Full custom	Semi custom	Semi custom
Area	Medium	Low	High
Power	Medium	High	Low

perform the transitions $0 \rightarrow 1 \rightarrow 0$ during a clock clock and another complementary output has no transition. This means that SABL always performs the same transitions at its outputs during each clock cycle independent of its inputs.

Fig. 4.1 shows the transistor schematic of a generic n-type SABL cell. The n-type SABL cell consists of the differential pull-down network (DPDN) which is made of NMOS transistors and the cross-coupled inverters I_1 and I_2 of which the output is connected to the input of another inverter and vice versa.

An n-type SABL cell is in the precharge phage when the clock signal is 0. During the precharge phase, the PMOS transistors M_3 and M_4 are turned on and then all internal nodes of an n-type SABL cell are set to 1. As a result, the inverters I_3 and I_4 produce a precharge value of $(0, 0)$ at the complementary outputs.

When the clock signal is 1, the n-type SABL cell is in the evaluation phase. During the evaluation phase, the input signals $in_1, \overline{in_1}, \ldots, in_n, \overline{in_n}$ are set to complementary values. The NMOS transistor M_2 is turned on and the PMOS transistors M_3 and M_4 are turned off. Thus, the nodes n_3 and n_4 of the DPDN are set to 0. One of the nodes n_1 and n_2 is connected to one of the nodes n_3 and n_4, which is determined by the structure of the DPDN. If n_1 is connected to 0 via the DPDN, the inverter I_1 is operational. Since the input signal n_6 of the inverter I_1 is still 1, the output signal n_5 of the inverter I_1 is switched to 0. The node n_5 also works as the input of the inverter I_2 and thus the output signal n_6 of the inverter I_2 stays at 1. The complementary outputs *out* and $\overline{out}$ are set to $(1, 0)$. If n_2 is connected to 0 via the DPDN, the inveter I_2 is working. By the signal 1 of n_5, the node n_6 is switched to 0. The complementary outputs *out* and $\overline{out}$ are set to $(0, 1)$.

In order for n-type SABL cells to consume constant power, the DPDN in the cell must be satisfied with some requirements, and the internal structure of the cells must be balanced.

DPDN requirements :

1) Every internal node of the DPDN should be connected to one of the four output nodes n_1, n_2, n_3 or n_4. Together with the NMOS pass-transistor M_1, this structure ensures that all internal nodes of the DPDN are discharged to 0 during the evaluation phase and charged to 1 during the precharge phase.

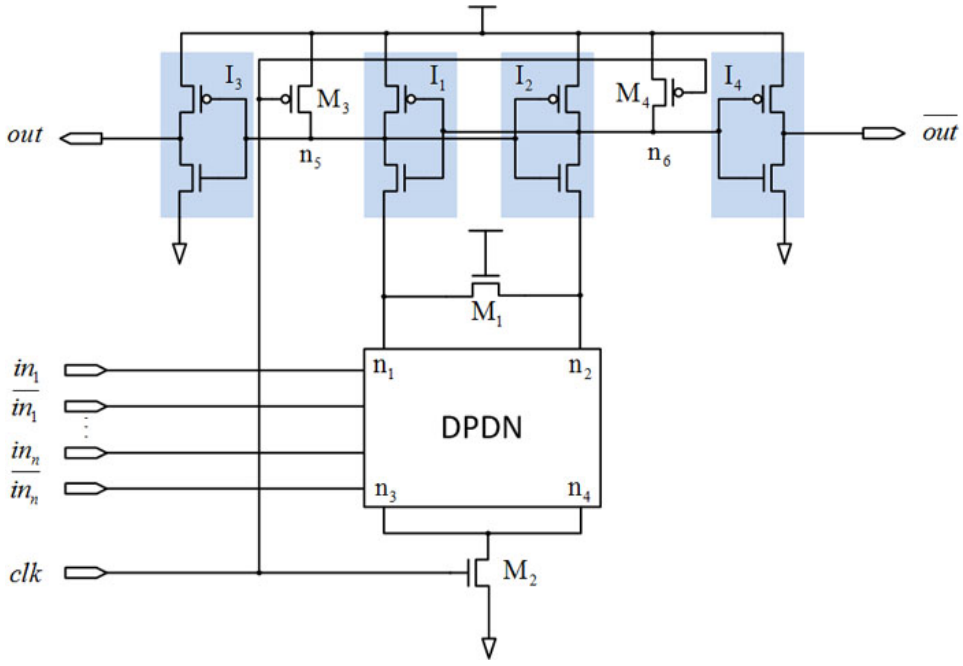


Figure 4.1: Schematic of a n-type SABL cell

2) Every possible conducting path in the DPDN should have the same resistance.

3) Both wire of every complementary input wire pair must be connected to the same number of gate terminals of transistors with identical parameters. This ensures that the capacitance of complementary inputs of SABL cells are pairwise balanced.

4.3 Wave Dynamic Differential Logic (WDDL)

Wave Dynamic Differential Logic has been also introduced by Tiri *et al.* [Tiri and Verbauwhede (2004), Mangard et al. (2007)]. WDDL cells can be built based on general logic cells in the standard cell library. The structure of WDDL cells is much simpler than that of SABL cells. This leads in general to less complex and significantly smaller circuits. Another advantage of WDDL cells is that they can also be realized on FPGAs.

Fig. 4.2 shows the schematic of a combinational WDDL cell. A combinational WDDL cell

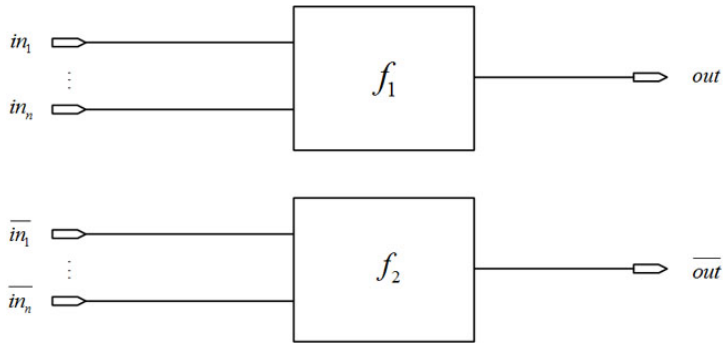


Figure 4.2: Schematic of a combinational WDDL cell

basically consists of two circuits that realize the Boolean function f_1 and f_2 such that

$$f_1(in_1, \ldots, in_n) = out$$
$$f_2(\overline{in_1}, \ldots, \overline{in_n}) = \overline{out}$$
$$f_1(in_1, \ldots, in_n) = \overline{f_2(\overline{in_1}, \ldots, \overline{in_n})},$$

where $(in_1, \overline{in_1}, \ldots, in_n, \overline{in_n})$ are complementary input signals and $(out, \overline{out})$ are complementary output signals. These Boolean functions must be positive monotonic in order to achieve the same transitions at output signals during each clock cycle for all possible transitions of input signals. The positive monotonic Boolean functions mean that if any input signals change in a direction $0 \rightarrow 1$ or $1 \rightarrow 0$, either *out* or $\overline{out}$ must be switched in the same direction.

Assuming that the precharge value is set to 0, in the precharge phase, all complementary input signals are set to 0. Since any $1 \rightarrow 0$ transitions of input signals result in only one $1 \rightarrow 0$ transitions of an output signal, complementary output signals must be set to 0. In the evaluation phase, all input signals are set to complementary values such as $(0, 1)$ or $(1, 0)$. A $0 \rightarrow 1$ transition at either *out* or $\overline{out}$ node must occur because of $0 \rightarrow 1$ transitions of input signals. As a result, either *out* or $\overline{out}$ always changes like $0 \rightarrow 1 \rightarrow 0$ and another output signal stays at 0 during a clock cycle.

4.4 t-private Private Circuit

We assume that an adversary can observe only limited number of internal nodes per clock cycle. In other words, this adversary has bandwidth limitations. This is the t-observation limited, interactive adversary of Ishai *et al.* [Ishai et al. (2003)]. We adopt a variant of Agrawal and Aggarwal [Agrawal and Aggarwal (2001)] who provide an entropy based definition of privacy.

Definition 7. *Privacy of a single variable X is defined as the entropy of X:*

$$h(X) = -\int_{\Omega_X} f_X(x) \log f_X(x) dx$$

Note that Ω_X is the domain of X and x is a value in Ω_X. This is the classical information theoretic definition of entropy for a variable X viewed as a random variable.

If this variable X's privacy were to be enhanced by applying a perturbing variable R, we can capture conditional entropy of X as follows.

Definition 8. *Conditional privacy of a single variable X perturbed by a variable R is defined as the conditional entropy of X:*

$$h(X|R) = -\int_{\Omega_{X,R}} f_{X,R}(x,r) \log f_{X|R}(x|r) dX dR.$$

The loss of privacy for X resulting from the exposure of R is the key definition of privacy developed in Agrawal and Aggarwal [Agrawal and Aggarwal (2001)].

Definition 9. *The privacy loss for variable X resulting from the exposure of a perturbing variable R is defined as :*

$$1 - \frac{2^{h(X|R)}}{2^{h(X)}}.$$

Note that if R is a random variable chosen independently from X (as is the case in [Messerges (2000)] and [Ishai et al. (2003)]), the privacy loss is 0 since $h(X|R) = h(X)$. Now we can define the notion of *privacy* as used in Ishai *et al.* [Ishai et al. (2003)] .

Definition 10 (t-private circuits:)**.** *A variable x is designed to be t-private if when perturbed by $k \leq t$ variables $r_{x_1}, r_{x_2}, \ldots, r_{x_k}$, the privacy loss for x resulting from the exposure of any subset of up to t perturbing variables is 0.*

Note that this definition of privacy insists on maintaining 0 correlation between the protected variable x and any subset of its perturbing variables. In these schemes, x is represented by at least $t+1$ physical variables, also known as its shares $x_{s_0}, x_{s_1}, \ldots, x_{s_t}$. In other words, almost all the shares of x carry 0 information about x in these schemes. We call such privacy schemes *information isolating* schemes or *information isolating* shares.

Messerges [Messerges (2000)] splits each variable x into two shares r_x (a random bit) and $r_x \oplus x$. He calls this scheme a masking scheme. He also introduces a similar arithmetic masking variant. Ishai *et al.* generalize this scheme to split x into $t+1$ shares $x_{s_0} = r_{x_1}, x_{s_1} = r_{x_2}, \ldots, x_{s_{t-1}} = r_{x_t}$, $x_{s_t} = r_{x_1} \oplus r_{x_2} \oplus \cdots \oplus r_{x_t} \oplus x$. They then provide a transformation for the Boolean basis of a NOT gate and an AND gate where each operand is a $t+1$ bit value.

4.4.1 Ishai's t-private circuit

Definition 11 (Input Encoder *I*)**.** *Each input x_i is split into $t+1$ shares: First, t random binary values, $r_{x_1}, r_{x_2}, \ldots, r_{x_t}$ are chosen for $x_{s_0}, x_{s_1}, \ldots, x_{s_{t-1}}$ using t random-bit gates. And then x_{s_t} is encoded into $x \oplus r_{x_1} \oplus r_{x_2} \oplus \cdots \oplus r_{x_t}$. The circuit I computes the encoding of each input bit independently in this way.*

Definition 12 (Output Decoder *O*)**.** *Each output of a circuit has $t+1$ bits, $y_{s_0}, y_{s_1}, \ldots, y_{s_t}$, which are decoded into $y_{s_0} \oplus y_{s_1} \oplus \cdots \oplus y_{s_t}$ in order to obtain real output.*

Definition 13 (t-private NOT circuit)**.** *Only a wire of split inputs, $x_{s_0}, x_{s_1}, \ldots, x_{s_t}$ is connected to a NOT gate.*

Definition 14 (t-private AND circuit)**.** *Consider an AND gate with inputs a, b and ouput c. Let input shares of a and b be a_i, b_i for $0 \leq i \leq t$, respectively and output shares of c be c_i for $0 \leq i \leq t$. In the transformation of an AND gate, we first compute intermediate values $z_{i,j}$ for $i \neq j$. For each $0 \leq i < j \leq t$, $z_{i,j}$ is a random bit and $z_{j,i}$ is equal to $(z_{i,j} \oplus a_i b_j) \oplus a_j b_i$. Now, we compute the output bits $c_0, c_1, \cdots, c_t$ as*

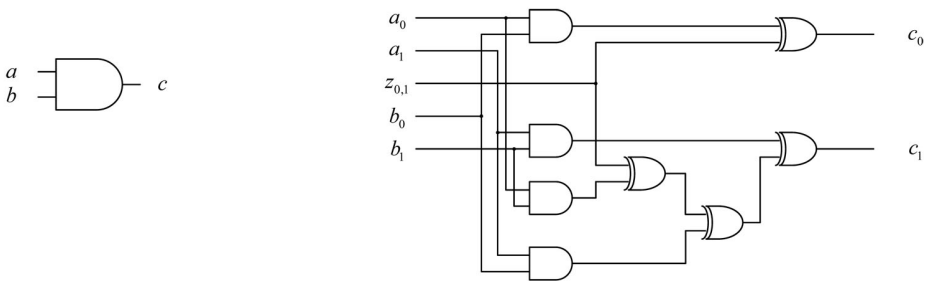
$$c_i = a_i b_i \oplus \bigoplus_{j \neq i} z_{i,j}. \tag{4.1}$$

Definition 15 (t-private OR circuit)**.** *Consider an OR gate with inputs a, b and ouput c. Let input shares of a and b be a_i, b_i for $0 \leq i \leq t$, respectively and output shares of c be c_i for $0 \leq i \leq t$. For one a_i and one b_j, these bits should be inverted. In the transformation of an OR gate, we first compute intermediate values $z_{i,j}$ for $i \neq j$. For each $0 \leq i < j \leq t$, $z_{i,j}$ is a random bit and $z_{j,i}$ is equal to $(z_{i,j} \oplus a_i b_j) \oplus a_j b_i$. Now, we compute the output bits $c_0, c_1, \cdots, c_t$ as*

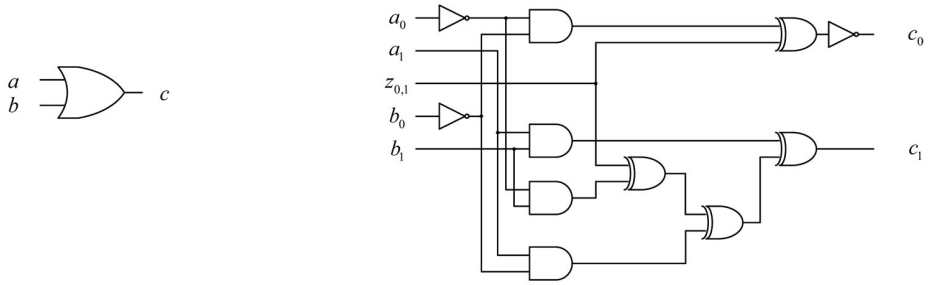
$$c_i = a_i b_i \oplus \bigoplus_{j \neq i} z_{i,j}. \tag{4.2}$$

where one c_i is connected to a NOT gate.

Fig. 4.3 describes the Ishai's t-private AND and OR circuit when t is 1. The area and energy overhead is of the order of t^2 [Tyagi (2005)]. We develop the following schema with smaller overhead.



(a) a AND gate and t-private AND circuit ($t = 1$)



(b) a OR gate and t-private OR circuit ($t = 1$)

Figure 4.3: The Ishai's t-private circuits ($t = 1$).

4.4.2 The modified t-private circuit

Theorem 7 (AND-XOR network with a random bit)**.** *Fig. 4.4 AND-XOR network with a random bit has the perfect secrecy for two inputs, x_1, x_2 and an intermediate value, x_{i1}, if r is a random variable.*

Proof.

$$\begin{aligned}\Pr_{z|x}(i|j) &= \frac{\Pr_{z,x}(i,j)}{\Pr_x(j)} = \frac{\Pr_z(i)\Pr_x(j)}{\Pr_x(j)} \\ &= \Pr_z(i) = 0.5 \\ &for \;\; i,j \in \{0,1\},\; x \in \{x_1, x_2, x_{i1}\}.\end{aligned}$$

□

Theorem 8 (Expanded AND-XOR network)**.** *We can expand an AND-XOR network with a random bit by XORing it with another AND gate. This expanded network can be expanded*

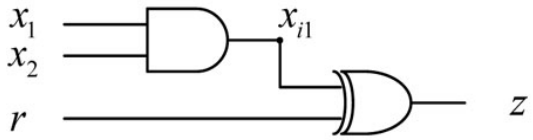


Figure 4.4: An AND-XOR network with a random bit.

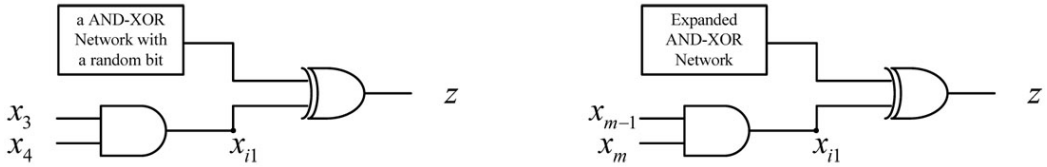


Figure 4.5: An expanded AND-XOR network.

continuously using the same structure. These expanded AND-XOR networks also have perfect secrecy for all inputs and any intermediate value. In other words, $\Pr_{z|x}(i|j)$ *is equal to* $\Pr_z(i)$ *for* $i, j \in \{0, 1\}$.

We modified Ishai's t-private circuit [Ishai et al. (2003)] into a simpler t-private circuit using the expanded AND-XOR network. This also requires fewer random bits.

Definition 16 (Modified t-private AND circuit)**.** *i) When* t *is an odd number,*

$$c_i = (a_0 b_i \oplus z_{i \bmod \frac{t+1}{2}}) \bigoplus_{j=1}^{t} a_j b_{(j+i) \bmod t+1}$$

$$for\ i = 0, 1, \ldots, t$$

ii) When t *is an even number,*

$$c_i = (a_0 b_i \oplus z_{i \bmod \frac{t+2}{2}}) \bigoplus_{j=1}^{t} a_j b_{(j+i) \bmod t+1}$$

$$for\ i = 0, 1, \ldots, t-1$$

$$c_t = (a_0 b_t \oplus z_{t \bmod \frac{t+2}{2}} \oplus z_{t+1 \bmod \frac{t+2}{2}}) \bigoplus_{j=1}^{t} a_j b_{(j+i) \bmod t+1}$$

Table 4.2: Comparison between t-private AND circuits

	Modified t-private AND circuit	Ishai's t-private AND circuit
outputs ($t = 2$)	$c_0 = (a_0b_0 \oplus z_0) \oplus a_1b_1 \oplus a_2b_2$ $c_1 = (a_0b_1 \oplus z_1) \oplus a_1b_2 \oplus a_2b_0$ $c_2 = (a_0b_2 \oplus z_0 \oplus z_1) \oplus a_1b_0 \oplus a_2b_1$	$c_0 = a_0b_0 \oplus z_{0,1} \oplus z_{0,2}$ $c_1 = a_1b_1 \oplus z_{1,2} \oplus \{(a_0b_1 \oplus z_{0,1}) \oplus a_1b_0\}$ $c_2 = a_2b_2 \oplus \{(a_0b_2 \oplus z_{0,2}) \oplus a_2b_0\} \oplus \{(a_1b_2 \oplus z_{1,2}) \oplus a_2b_1\}$
# of random bits	$\lceil \frac{t+1}{2} \rceil = O(t)$	$\frac{t(t+1)}{2} = O(t^2)$
# of XOR gates	Ishai's model has additional $t(t+1) - 2\lceil \frac{t+1}{2} \rceil$ XOR gates compared to modified t-private model.	

where z_j is a random bit.

Table 4.2 shows comparison between the modified t-private AND circuit and the Ishai's t-private AND circuit. The modified t-private circuit has smaller number of random bits and XOR gates and almost the same delay.

4.5 Design of Secure logic style

4.5.1 Design of SABL-NAND

Fig. 4.6 shows the transistor schematic of a n-type SABL NAND gate using *Virtuoso* schematic editor with NCSU FreePDK45 technology library [NCSU (2011)] . It consists of the differential pull-down network (DPDN) and the cross-coupled inverters. The DPDN is made of NMOS transistors. The DPDN satisfies all requirments. First, all internal nodes for complementary inputs signals are connected to one of the four output nodes of the DPDN. Second, every conducting path goes through two NMOS transistors and thus has the same resistance since all NMOS transistors in the DPDN are equally sized. Third, all complementary input wires are connected to the same number of transistors.

4.5.1.1 Simulation of SABL-NAND

We simulate the SABL NAND gate using Cadence Spectre. Fig. 4.12 shows the waveforms of inputs, outputs and currents. One output of output signals is switched such as $0 \rightarrow 1 \rightarrow 0$.

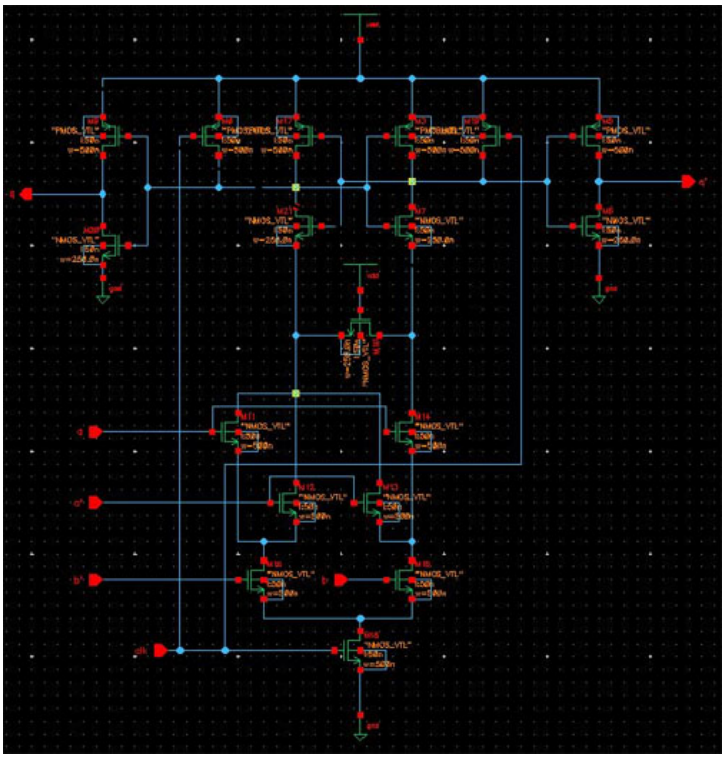


Figure 4.6: Schematic of SABL-NAND gate

The waveforms of currents for all possible inputs have the same form and power consumptions are almost constant. Table 4.3 shows power consumption and peak current for all possible inputs.

Table 4.3: Power consumption of **SABL NAND** (45 nm process)

Input a	Input b	Power consumption (nW)	Peak Current (mA)
0	0	5757.87	0.2544
0	1	5752.88	0.2536
1	0	5760.58	0.2544
1	1	5753.98	0.2526
-	Average(μ)	5756.33	0.2538
-	Standard deviation(σ)	3.5524	0.0008
-	$\frac{\sigma}{\mu}$	0.0006	0.0034

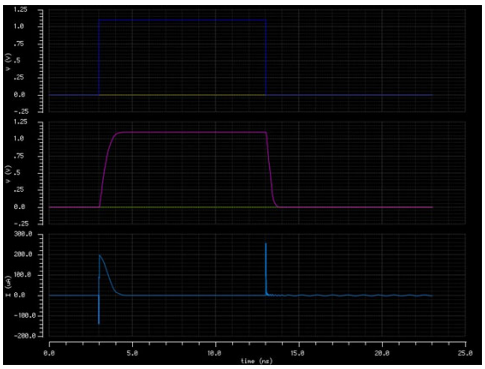


Figure 4.8 Input $a = 0, b = 0$

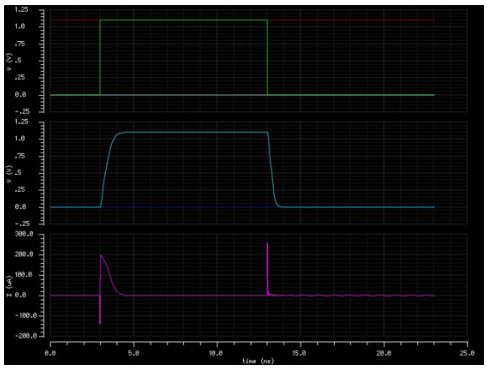


Figure 4.9 Input $a = 0, b = 1$

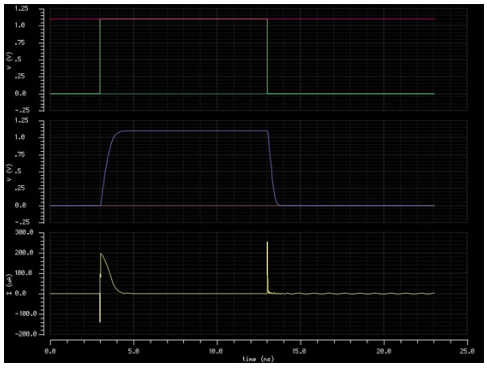


Figure 4.10 Input $a = 1, b = 0$

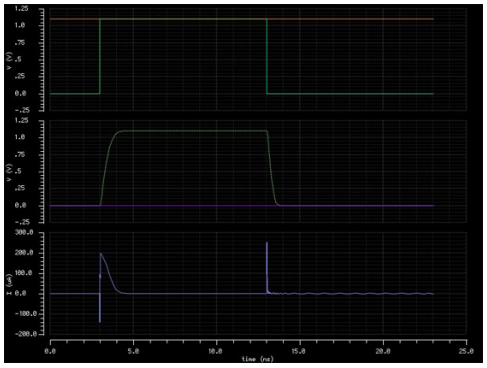


Figure 4.11 Input $a = 1, b = 1$

Figure 4.12: Waveform of SABL NAND gate

4.5.2 Design of WDDL

Oklahoma State University digital cell library based on the FreePDK45 technology library [OSU (2008)] is used to design WDDL cells. Fig. 4.13 shows the schematic of a WDDL-NAND gate using *Virtuoso* schematic editor.

4.5.2.1 Simulation of WDDL-NAND

We simulate the WDDL-NAND gate using Cadence Spectre. Fig. 4.19 shows the waveforms of inputs, outputs and currents. One output of output signals is switched such as $1 \rightarrow 0 \rightarrow 1$. The waveforms of currents for all possible inputs have the same form and power consumptions are almost constant. Table 4.4 shows power consumption and peak current for all possible

inputs.

Table 4.4: Power consumption of **WDDL NAND** (45 nm process)

Input a	Input b	Power consumption (nW)	Peak Current (mA)
0	0	5939.15	0.4086
0	1	5927.37	0.4065
1	0	5885.04	0.4286
1	1	5872.73	0.4137
-	Average(μ)	5906.07	0.4144
-	Standard deviation(σ)	32.15	0.0099
-	$\frac{\sigma}{\mu}$	0.0054	0.024

4.5.3 Design of t-private logic cells

Fig. 4.20 and 4.21 show the schematic of $t = 1$-private NAND circuit and $t = 1$-private AND circuit using *Virtuoso* schematic editor with the OSU FreePDK45 cell library [OSU (2008)].

4.5.3.1 Simulation of t-private logic circuit

We simulate $t = 1$-private NAND circuit and $t = 1$-private AND circuit using Cadence *Spectre*. Table 4.5 and 4.6 show power consumption and peak currents for all possible output transitions of $t = 1$-private NAND and AND circuit, respectively.

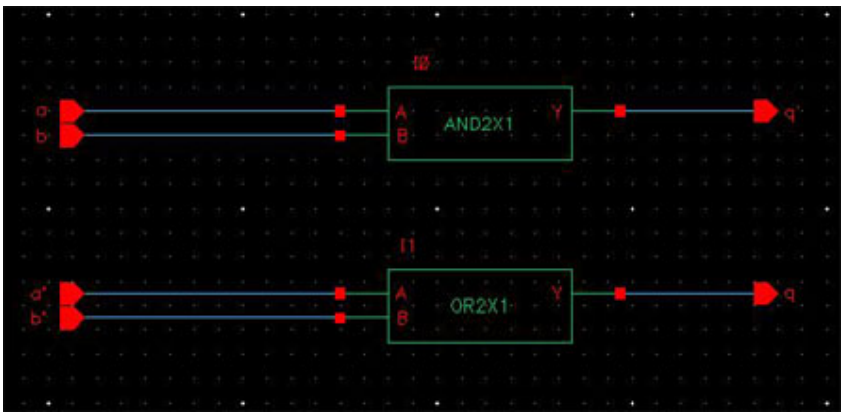


Figure 4.13: Schematic of WDDL-NAND gate

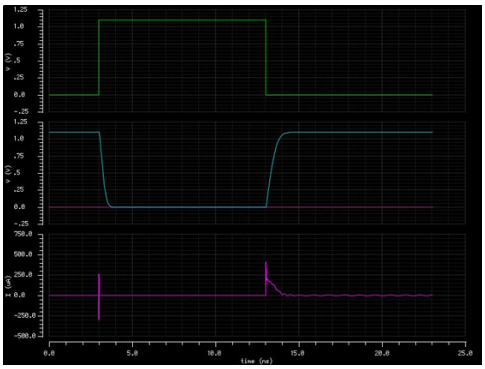


Figure 4.15 Input $a = 0, b = 0$

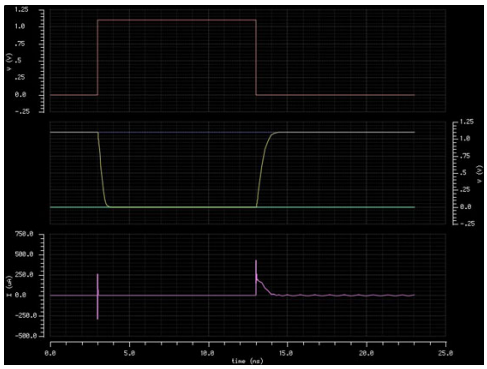


Figure 4.16 Input $a = 0, b = 1$

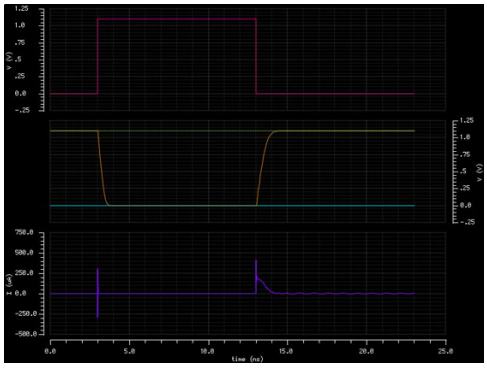


Figure 4.17 Input $a = 1, b = 0$

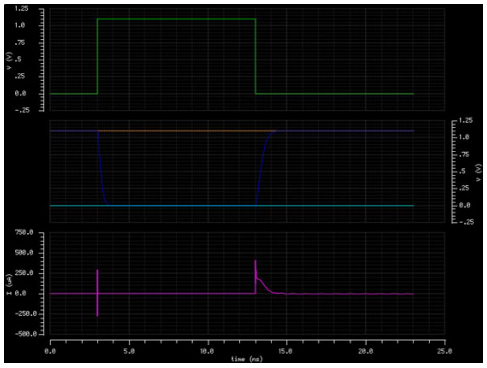


Figure 4.18 Input $a = 1, b = 1$

Figure 4.19: Waveform of WDDL NAND gate

4.5.4 Comparison of t-private NAND, SABL-NAND and WDDL-NAND

Table 4.7 shows the mean (μ) and the standard deviation (σ) of power consumption for all possible transitions of output signals, the average peak current, the number of PMOS transistors and the number of NMOS transistors. The power consumption of $t(= 1)$-private NAND circuit has the smallest values even though it consumes the largest area. Since SABL-NAND and WDDL-NAND require the precharge phase and the evaluation phase during a clock cycle in which phase transitions of input and output signals occur, the power consumption of SABL-NAND and WDDL-NAND has larger values than that of t-private NAND circuit. But the peak current of t-private NAND is the highest. Based on the normalized variance metric, SCA vulnerability of SABL-NAND is the lowest. Note that only t-private logic circuit has

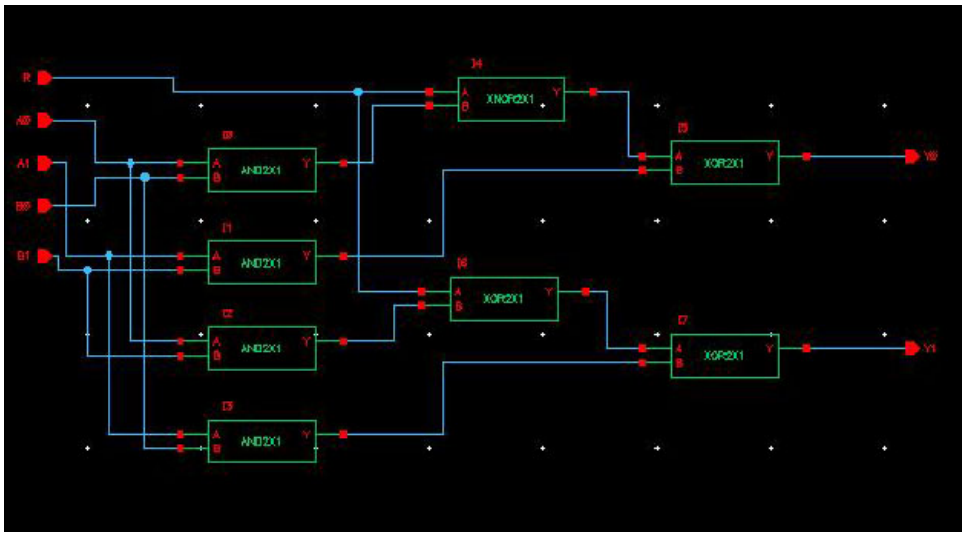


Figure 4.20: Schematic of NAND2X1t1

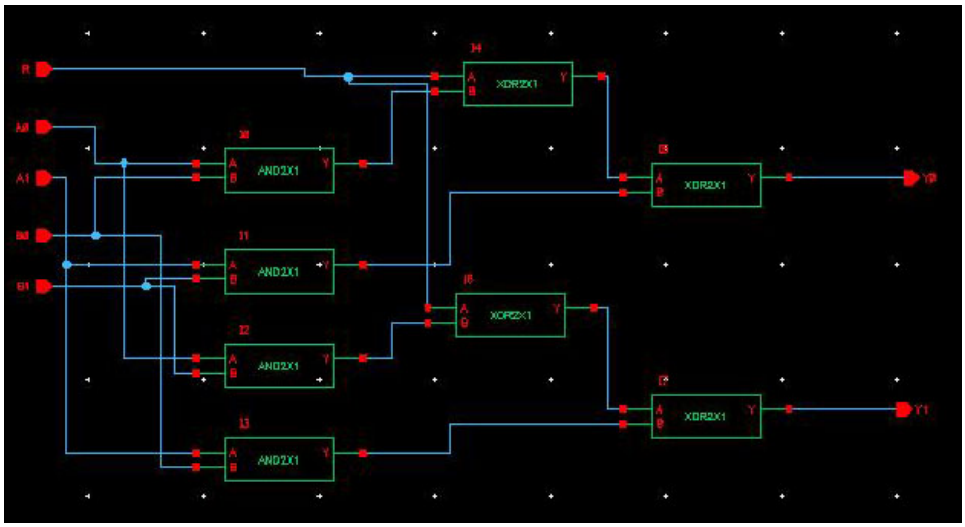


Figure 4.21: Schematic of AND2X1t1

Table 4.5: Power consumption of **NAND2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4194.55	0.719	64
$0 \rightarrow 1$	4173.27	0.745	192
$1 \rightarrow 0$	4194.40	0.668	192
$1 \rightarrow 1$	4178.08	0.701	576
Average(μ)	4185.08	0.709	-
Standard deviation(σ)	121.73	0.001	-
$\frac{\sigma}{\mu}$	0.029	0.0014	-

Table 4.6: Power consumption of **AND2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4209.30	0.717	576
$0 \rightarrow 1$	4230.54	0.657	192
$1 \rightarrow 0$	4207.52	0.725	192
$1 \rightarrow 1$	4225.71	0.701	64
Average(μ)	4215.76	0.699	-
Standard deviation(σ)	77.15	0.0009	-
$\frac{\sigma}{\mu}$	0.018	0.0013	-

robustness against both SCA attack and probing attack even though it has the largest area and peak current. In order to prevent from both SCA attacks and probing attack, t-private logic circuits should be utilized.

Table 4.7: Comparison of t-private NAND, SABL-NAND and WDDL-NAND

	t-private NAND	SABL-NAND	WDDL-NAND
Average (μ) of Power (nW)	4185.08	5736.33	5906.07
Standard deviation (σ) of Power (nW)	121.73	3.55	32.15
Average of Peak current (mA)	0.709	0.2538	0.4144
$\frac{\sigma}{\mu}$	0.029	0.0006	0.0054
Number of PMOS	36	6	6
Number of NMOS	36	12	6

4.5.5 SCA attacks of t-private logic circuit

In order to verify SCA vulnerability of t-private logic circuits, profiling SCA attacks are performed. Simulation results of Cadence *Specte* are used for profiling (or training). LS-SVM and QDA classifier recognize power traces as one of 4 classes which corresponds to $0 \rightarrow$

Table 4.8: Successful recognition rate of t-private circuits using LS-SVM and QDA classifiers

	LS-SVM	QDA
NAND2X1t1	19.26 %	31.44 %
AND2X1t1	24.85 %	25.09 %
OR2X1t1	14.81 %	15.52 %
NOR2X1t1	15.50 %	16.21 %

$0,0 \rightarrow 1,1 \rightarrow 0$ and $1 \rightarrow 1$ transition of the output signal. QDA classifier has the largest successful recognition rate (31.44%) of t-private NAND circuit. This value is only 6 % larger than randomly selected recognition rate which is equal to 25%. In other cases, successful recognition rates are less than 25%. As a result, these t-private logic circuits are mostly secure against SCA attacks.

4.6 Conclusion

In this chapter, SABL cells, WDDL cells and t-private logic cells have been implemented. These secure logic styles are necessities for SCA robust hardware implementation. They are included in the technology library. We verify SCA vulnerability of t-private logic circuits using machine learning technique such as LS-SVM and QDA.

CHAPTER 5. FPGA IMPLEMENTATION AND ASIC IMPLEMENTATION

5.1 Introduction

In this chapter, we propose the methodology of secure hardware implementation on two different hardware, FPGA and ASIC. FPGA chip is made up of a finite number of configurable logic blocks (CLBs) with programmable interconnects to implement a reconfigurable digital circuit. The CLBs are the basic logic unit of an FPGA and made up of two basic components : flip-flops and lookup tables (LUTs). On the other hand, ASIC is implemented with standard cells which consist of digital logic gates such as AND, OR, NAND, INVERTER, XOR, flip-flops, buffers and so on. All kinds of secure logic styles can be utilized on ASIC implementation but secure logic styles to synthesize on FPGA are WDDL cells are t-private logic circuits.

We focus on t-private logic circuits for both FPGA and ASIC design. For more suitable and efficient design on FPGA, t-private logic circuits are modified. The modified version is called tail-recursive t-private circuits. In Section 5.2, the tail recursive t-private circuit is defined and we deal with how to map the secure logic style on FPGA. Typical ASIC design flow requires the standard cell library for logic synthesis, place & route, physical layout and timing verification. t-private logic circuit as well as general digital logic cells should be included in the standard cell library. In Section 5.3, we propose the method to build the secure logic cell library and to implement secure ASIC design. Finally, Section 5.5 concludes this chapter.

5.2 FPGA Implementation

5.2.1 The tail recursive t-private circuit

Ishai [Ishai et al. (2003)] describes a transformation that is best applied in topological order with input bits transformed first. An alternate way would be to apply the recursion at the output node. This is what we call *tail-recursive* private circuits.

Consider a function $f(x_1, x_2, \ldots, x_n)$ of n bits. If we wanted t-privacy, we will first determine t random shares just as in Ishai's schema. However, these random shares are at the granularity of function truth tables. Hence we will generate $f_i^r(x_1^0, \ldots, x_1^t, x_2^0, \ldots, x_2^t, \ldots, x_n^0, \ldots, x_n^t)$ as a random truth table $[t_0^{f_i}, t_1^{f_i}, \ldots, t_{2^{nt}-1}^{f_i}]$ for $i = 0, 1, \ldots, t-1$. The $(t+1)$st share would be derived from the other random t shares so that $f_t(x_1^0, \ldots, x_1^t, x_2^0, \ldots, x_2^t, \ldots, x_n^0, \ldots, x_n^t)$ has the truth table $[t_0^{f_t}, t_1^{f_t}, \ldots, t_{2^{nt}-1}^{f_t}]$ such that $t_j^{f_t} = t_j^f \oplus t_j^{f_1} \oplus t_j^{f_2} \oplus \cdots \oplus t_j^{f_{t-1}}$ for $0 \leq j \leq 2^{nt} - 1$. For the perfect secrecy, each function, f_i^r, f_t should meet the following condition:

$$\Pr_{f_i|x_j}(p|q) = \Pr_{f_i}(q) \; for \; p, q \in \{0, 1\}.$$

Definition 17 (The tail recursive t-private circuit)**.** *Let a original function with n inputs be $f(x_1, x_2, \ldots, x_n)$. Each input, x_m has t-random shares, $x_m^0, x_m^1, \ldots, x_m^{t-1}$, and an encoded bit, $x_m^t = x_m \oplus x_m^0 \oplus x_m^1 \oplus \cdots \oplus x_m^{t-1}$. The tail recursive t-private circuit is defined as follows:*

$$f_i^r = \bigoplus_{m \in M} x_m^i \tag{5.1}$$

$$f_t = f \bigoplus_{i \in I} f_i^r \tag{5.2}$$

$$where \; I = \{0, 1, \ldots, t-1\},$$

$$M \subseteq \{1, 2, \ldots, n\},$$

$$1 \leq |M| \leq n.$$

Note that M is a random subset of $\{1, \ldots, n\}$ and f_i^r in (5.1) is a random function.

Proof. Since x_m^i for $i \in I$ is a random variable,

$$\Pr_{f_i^r|x_m^i}(p|q) = \Pr_{f_i^r}(p) = 0.5, \ \textit{where } p, q \in \{0, 1\}.$$

Thus, f_i^r has perfect secrecy for all inputs.

Let us verify whether f_t has perfect secrecy. An n-variable Boolean function f can be expressed in the following Canonical Reed-Muller expansion [Reed (1954)] of 2^n terms:

$$f(x_1, x_2, \ldots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_{2^n-1} x_1 x_2 \cdots x_n,$$

where $a_i \in \{0, 1\}$.

If we substitute $a_i(x_i^0 \oplus x_i^1 \oplus \cdots \oplus x_i^t)$ for $a_i x_i$, then

$$\begin{aligned}
f(x_1, \ldots, x_n) = {} & a_0 \oplus a_1(x_1^0 \oplus x_1^1 \oplus \cdots \oplus x_1^t) \\
& \oplus a_2(x_2^0 \oplus x_2^1 \oplus \cdots \oplus x_2^t) \oplus \cdots \\
& \oplus a_n(x_n^0 \oplus x_n^1 \oplus \cdots \oplus x_n^t) \\
& \oplus a_{n+1}\Big(\bigoplus_{i \neq j} x_i x_j\Big) \oplus \cdots \oplus a_{2^n-1} x_1 x_2 \cdots x_n \\
= {} & \Big(a_1 x_1^0 \oplus a_2 x_2^0 \oplus \cdots \oplus a_n x_n^0\Big) \\
& \oplus \Big(a_1 x_1^1 \oplus a_2 x_2^1 \oplus \cdots \oplus a_n x_n^1\Big) \oplus \cdots \\
& \oplus \Big(a_1 x_1^{t-1} \oplus a_2 x_2^{t-1} \oplus \cdots \oplus a_n x_n^{t-1}\Big) \\
& \oplus \Big(a_0 \oplus a_1 x_1^t \oplus a_2 x_2^t \oplus \cdots \oplus a_n x_n^t \oplus \\
& a_{n+1}\Big(\bigoplus_{i \neq j} x_i x_j\Big) \oplus \cdots \oplus a_{2^n-1} x_1 x_2 \cdots x_n\Big).
\end{aligned}$$

$$
\begin{aligned}
f_t &= f_0^r \oplus f_1^r \oplus \cdots \oplus f_{t-1}^r \oplus f \\
&= \Big(a_1 x_1^0 \oplus a_2 x_2^0 \oplus \cdots \oplus a_n x_n^0 \oplus f_0^r\Big) \\
&\oplus \Big(a_1 x_1^1 \oplus a_2 x_2^1 \oplus \cdots \oplus a_n x_n^1 \oplus f_1^r\Big) \oplus \cdots \\
&\oplus \Big(a_1 x_1^{t-1} \oplus a_2 x_2^{t-1} \oplus \cdots \oplus a_n x_n^{t-1} \oplus f_{t-1}^r\Big) \\
&\oplus \Big(a_0 \oplus a_1 x_1^t \oplus a_2 x_2^t \oplus \cdots \oplus a_n x_n^t \\
&\oplus a_{n+1}\big(\bigoplus_{i \neq j} x_i x_j\big) \oplus \cdots \oplus a_{2^n-1} x_1 x_2 \cdots x_n\Big) \\
&= \Big(\big(\bigoplus_{m_0 \in M} x_{m_0}^0\big) \oplus \cdots \oplus \big(\bigoplus_{m_{t-1} \in M} x_{m_{t-1}}^{t-1}\big)\Big) \oplus f \\
&= f^r(x_1^0, \ldots, x_1^{t-1}, \ldots, x_n^0, \ldots, x_n^{t-1}) \oplus f(x_1, \ldots, x_n) \qquad (5.3) \\
&= f_t(x_1^0, \ldots, x_1^t, \ldots, x_n^0, \ldots, x_n^t). \qquad (5.4)
\end{aligned}
$$

Since $\Pr_{f^r|x_m^i}(p|q) = \Pr_{f^r}(p) = 0.5$ and $\Pr_{x_m|x_m^i}(p|q) = \Pr_{x_m}(p) = 0.5$ in (5.3),

$$\Pr_{f_t|x_m^i}(p|q) = \Pr_{f_t}(p) = 0.5,\ p, q \in \{0, 1\} \text{ in (5.4)}.$$

Thus, f_t also has perfect secrecy for all inputs, x_m^i. □

5.2.2 Mapping into k-LUTs with unlimited number of inputs

FPGAs have k-LUT granularity truth tables built in their architecture. From the power probing point of view each LUT is a black-box. This is because SRAMs precharge both *bit* and $\overline{bit}$ lines for all bits. Exactly one bit-line discharges. Hence, the proposed tail-recursive private circuits are ideally suited for FPGA architectures.

Each of the randomized truth tables can be mapped to its own LUT. Hence, all the t function level shares are isolated. The key assumption is that the t probes an adversary can

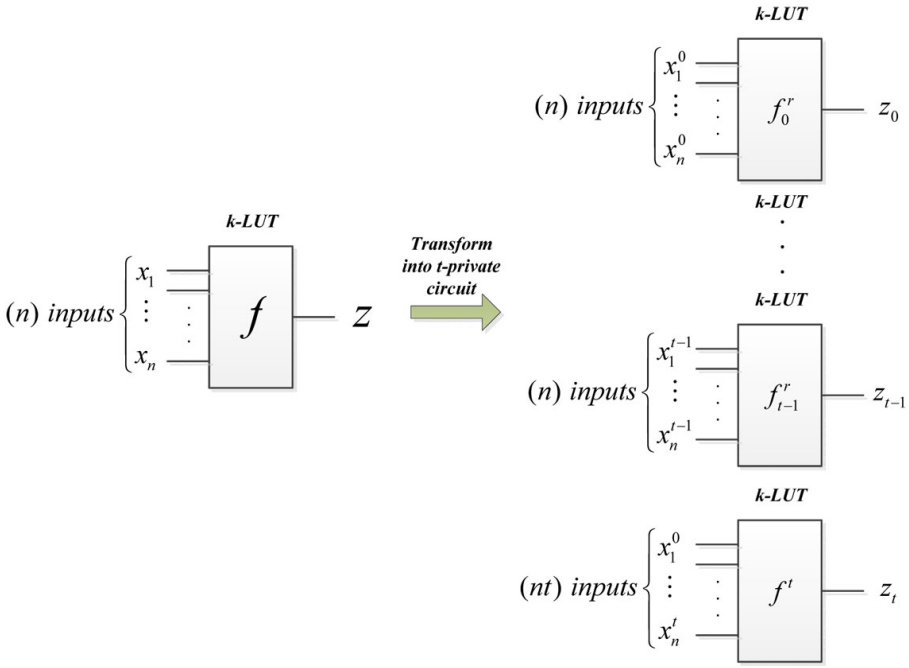


Figure 5.1: Transformation into LUT-based t-private circuit

use do not go inside a truth table. If we also assume that k-LUTs has sufficiently many inputs so that $|x_m^i| = nt \leq k \quad for \quad m \in \{1, .., n\},\ i \in \{0, ..., t\}$, the number of k-LUTs increases to t times the number needed for the original functions. With the LUT blackbox assumptions, we get t-privacy at a somewhat lower area and delay cost.

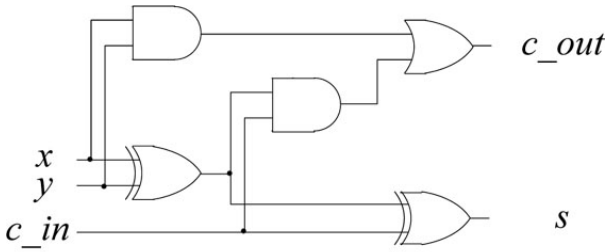
Lemma 9. *We assume that an adversary cannot probe internal nodes of LUTs and $k \geq nt$.*

The number of LUTs used increases linearly with t to achieve t-privacy. In order words, the complexity of the LUT-based t-private circuits is $O(t)$ and the depth of this circuit is $O(1)$.

Fig. 5.1 shows a function f mapped into a k-LUT and then transformed into $t+1$ k-LUTs in order to make it secure.

5.2.3 Mapping into k-LUTs with limited number of inputs

Most commercial FPGAs have from 4-LUT to 6-LUT granularity. With this choice for k, most LUTs utilize all their inputs after technology mapping. Given this practical constraint



$$s = x \oplus y \oplus c_{in}$$
$$c_{out} = xy + c_{in}(x \oplus y)$$

Figure 5.2: Full adder cell schemetic

on k-LUT granularity, our assumption should be changed to $k < nt$.

Lemma 10. *We assume that an adversary cannot probe internal nodes of LUTs and $k < nt$.*

The complexity of LUT-based t-privacy is $O(t + \log_k t)$ and the depth of this circuit is $O(\log_k t)$.

5.2.4 Implementation of t-private full adder

We synthesized adders in the Ishai's framework and in the LUT based tail-recursive model. We used Xilinx ISE tools for the synthesis. The target device is Xilinx Virtex-5 FPGA (XC5VFX70T-3FF1136). Fig. 5.2 shows a reference full adder. Fig. 5.3 shows a schematic for the modified Ishai's (t=1)-private full adder. Fig. 5.4 shows a chart comparing various adder implementations with respect to the number of LUTs (n-bit ripple carry adder based on Ishai's model with $t = 1, 2$ and 3; and the tail-recursive LUT based model with $t = 1, 2$). Fig. 5.5 shows the critical path delay for the same set of adders. The key point to note here is that the tail-recursive design takes approximately 50% area of Ishai scheme for similar privacy. The delay advantage of tail-recursive scheme is about 33%.

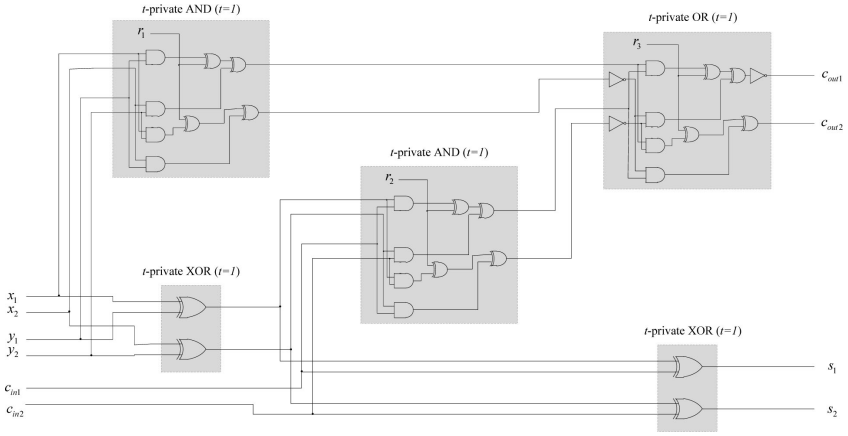


Figure 5.3: $(t = 1)$-private full adder cell schematic

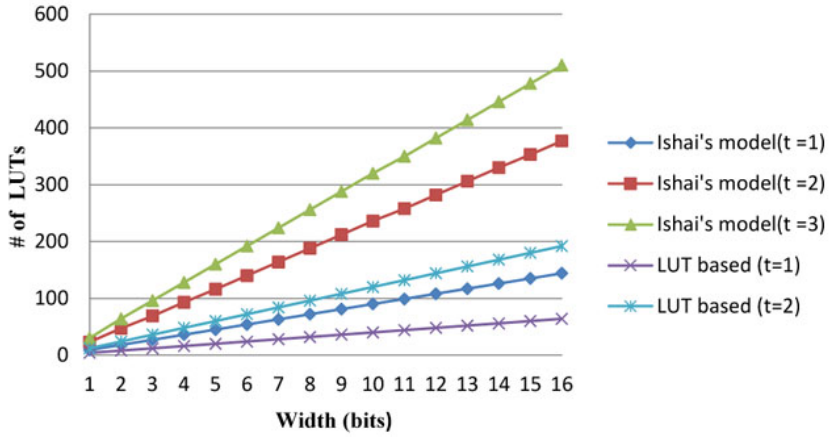


Figure 5.4: LUT costs of various t-private adders

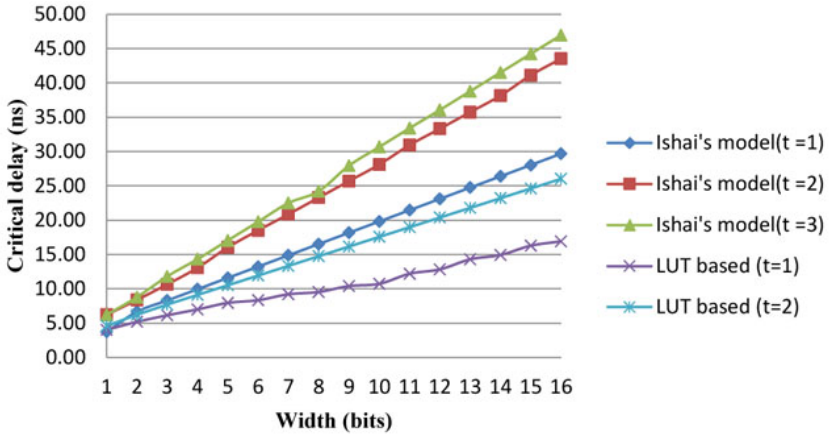


Figure 5.5: Delay costs of various t-private adders

5.3 ASIC Implementation

We introduce ASIC design implementation of t-private system using commercial EDA tools such as Cadence's tools and Synopsys's tools.

5.3.1 t-private Logic synthesis

After general logic synthesis, modules with low SCA resistance are flagged by the graph based SCA analysis. The flagged modules should be resynthesized at the logic level so that KL divergence metric is zero or almost zero. We call this re-synthesis t-private logic synthesis since t-private logic [Ishai et al. (2003)] will be employed. These t-private logic circuits have SCA resistance, which means that the normalized standard deviation of t-private logic circuits is almost zero. We will verify that these primitives of t-private logic synthesis have SCA robustness at the physical layout level in the following subsection. If each gate is replaced with the corresponding t-private logic circuit in such a way that AND gate is replaced with t-private AND circuit, the area of the module increases significantly [Park and Tyagi (2012)]. In order to reduce area, t-private XOR or NXOR are a better choice since these circuits have smaller area than t-private AND or OR circuits. An n-variable function $y = f(x_0, x_1, \ldots, x_{n-1})$ can be represented by

$$f = \sum \oplus x_0^* x_1^* \cdots x_{n-1}^*, \tag{5.5}$$

where x_i^* can be $1, x_i$ or x_i' and $\sum \oplus$ represents the EXOR sum-of-products (ESOP) [Sasao and Fujita (1996)]. The minterms and products of Eq. (5.5) can be replaced with t-private AND and XOR circuits, respectively.

Lemma 11. *If the boolean function $y = f(x_0, x_1, \ldots, x_{n-1}) = \sum \oplus x_0^* x_1^* \cdots x_{n-1}^*$ can be synthesized with t-private AND and XOR circuits, SCA effectiveness of the resulting combinational circuit is zero.*

Proof. First, consider the observability of a 2-input XOR gate $c = a \oplus b$.

$$\mathbf{Ob}_c(a) = \mathbf{Ob}_c(b) = \Pr[f_a \oplus f_{a'}] = 1$$

$$\mathbf{Ob}_c(a, b) = \Pr[(a \oplus b) \oplus (a' \oplus b')] = 0.$$

Thus, $\mathbf{P}_c(a)$ is equal to $\mathbf{P}_c(b)$ and the SCA effectiveness is zero. The observability of x_i at the primary output y is 0.5 because the observability is the multiplication of the observability of the input at a t-private AND circuit and all the observability of the input at an XOR gate : $0.5 \times 1 \times \cdots \times 1$. The effective capacitances $C_y(x_i)$ are almost equal. Consequently, $\mathbf{Var}[P_y(x_i)]$ is zero. This means that the combinational circuit is robust against SCA attacks. □

5.3.2 Design Flow

The design flow of the ASIC design of t-private system is shown in Fig. 5.6. All design procedures except for t-private logic synthesis is the same as the general ASIC design process. First, cryptographic system is designed at the behavioral level using HDL language such as Verilog or VHDL.

Second, the behavioral design is transformed into technology dependent gate level by logic synthesizer such as Candence's *RTL Compiler* or Synopsys's *Design Compiler* with the technology library. We call this process logic synthesis. We use *RTL Compiler* and OSU standard cell library based on NCSU FreePDK 45nm process as the logic synthesizer and the technology library, respectively. The technology library has liberty file format and the file extension of **.lib** which is the semiconductor industry's most widely supported library standard. These exist no difference with the general design flow until the second step.

The third process is to transform the vulnerable design based on the security metrics into t-private logic design which has robustness against the tth order side-channel attacks. We call this procedure t-private logic synthesis. This process is divided into two sub-steps. The first sub-step is to change each general gate into matched t-private gate in such a way that **AND** gate is changed into t-private **AND** circuit. It is performed automatically by *Perl* script. The following step is to optimize t-private logics depending on the time constraint using *RTL Compiler*. For this logic synthesis, we use our technology library including t-private logic cells such as **AND2X1t1, NAND2X1t2, XOR2X1t1** and so on. The name of the t-private logic cells represents operation function, the number of inputs, drive strength and t parameter in sequential order. For example, **AND2X1t1** means that this cell is X1 2-input AND with $t = 1$.

After the t-private logic synthesis, the structural Verilog file consisting of t-private logic cells is generated.

The back-end design starts from the fourth process for the final physical layout. We use the *SOC Encouter* tool from Cadence for the floorplan, place and route. The required files are our technical library file (**.lib**), cell abstract information file (**.lef**), the structural Verilog (**.v**) and delay constraint information file (**.sdc**), which the last two files are outputs of the previous process. The generated layout should pass DRC and LVS and is saved as the gds file format (**.gds**).

Finally, we should verify whether our implementation has security against the $t-$th order side-channel attacks or not based on power simulation using *Spectre* analog simulator from Cadence.

5.3.3 Technology Library

In order to perform t-private logic synthesis and physical layout, our technology library should be required. The technology library defines the cell function, area, delay and power dissipation of each t-private logic cell. The cell definition of **AND2X1t1** as liberty file format is show in Listing 5.1. To generate our technology library, several steps should be required as the following:

1) Draw the schematic of each t-private logic cell using *Virtuoso* schematic editor like Fig. 5.8.

2) Make a structural Verilog file based on the schematic like Fig. 5.15.

3) Synthesize the t-private logic cell using *RTL Compiler* like Fig. 5.10.

4) Generate a layout of the t-private logic cell using *SOC encounter* like Fig. 5.11.

5) Check DRC and LVS.

6) Extract timing and power characteristics of the t-private logic cell using *Spectre Analog Environment*.

Since t-private logic cells are made of gates of OSU standard digital cell library, OSU standard cell library is used for logic synthesis and layout. After Step 3, generated Verilog may be different from the structural Verilog at Step 2. Power and area can be estimated after logic synthesis. Table 5.1 shows area, power and delay time estimation of 5 ($t = 1$)-private

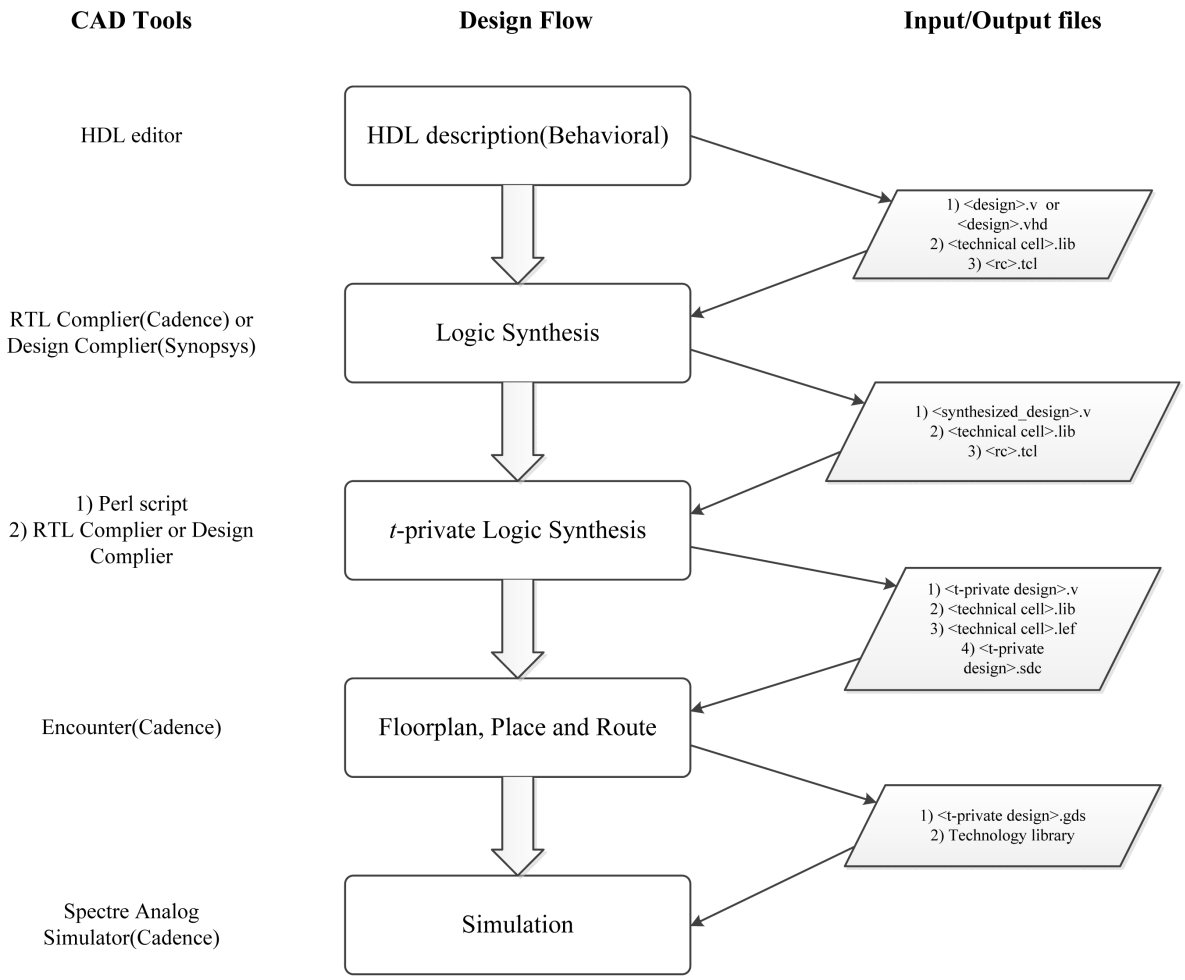


Figure 5.6: The design flow of the ASIC implementation

logic cells. We generate layouts and liberty descriptions of basic 8 ($t = 1$)-private logic cells (**AND2X1t1, NAND2X1t1, OR2X1t1, NOR2X1t1, XOR2X1t1, XNOR2X1t1, BUFX2t1, INVX2t1**) through the above method.

Table 5.1: Area, power and delay estimation of each t-private logic cell after logic synthesis

cell	Area	Leakage Power (nW)	Dynamic Power (nW)	Delay (ps)
NAND2X1t1	31	1.19	4185.08	55
AND2X1t1	31	1.19	4112.97	55
NOR2X1t1	32	1.052	4418.91	66
OR2X1t1	32	1.052	4407.66	66
XNOR2X1t1	10	0.36	4433.56	14
XOR2X1t1	10	0.361	4439.20	13

5.3.4 Verification of robustness

After finishing layout of basic t-private logics, we also verify the robustness against power analysis attacks. For the verification, we measured the power and current of logic cells using *Spectre Analog Environment* with the analog extracted view of the cell which includes all parasitic capacitances. The power consumption of logic gates in general standard cell libraries depends on transitions of the output. For example, the power consumption of **NAND2X1** of OSU standard cells varies according to how the output is changed. When transition of the output occurs, power of the supply is dissipated significantly compared to the power consumption in case of no transition. It also has difference between the transition from 0 to 1 and the transition from 1 to 0. This **NAND2X1** does not have robustness against power analysis attacks since the power consumption depends on processed data.

Basic t-private logic cells are simulated for all possible input pattern and the corresponding power and peak current were measured in each case. Two input t-private logics except for **XOR** and **XNOR** has $4^{2(t+1)+r}$ possible input patterns where r is equal to $\frac{\lceil t+1 \rceil}{2}$ and the number of required random bits for perfect secrecy of internal nodes. Since t-private **XOR** and **XNOR** does not require additional random bits for the perfect secrecy, the number of all possible input pattern is $4^{2(t+1)}$. The measured powers and peak currents were classified according to the

Listing 5.1: A sample example liberty description of AND2X1t1

```
cell (AND2X1t1) {
area : 3168;
  cell_leakage_power : 1.19;
  pin(A0)  {
    direction : input;
    capacitance : 0.021674;
    rise_capacitance : 0.021579;
    fall_capacitance : 0.021674;
  }
  pin(A1)  {
    ...
  }
  ...
  pin(Y0)  {
    direction : output;
    capacitance : 0;
    rise_capacitance : 0;
    fall_capacitance : 0;
    max_capacitance : 0.924889;
    function : "(A0*B0 ^ R ^ A1*B1)";
    timing() {
      related_pin : "A0";
      timing_sense : positive_unate;
      cell_rise(delay_template_5x5) {
        index_1 ("0.05, 0.1, 0.2, 0.6, 1.2");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          ...
      }
      rise_transition(delay_template_5x5) {
        ...
      }
      cell_fall(delay_template_5x5) {
        ...
      }
      fall_transition(delay_template_5x5) {
        ...
      }
    }
    timing() {
      related_pin : "A1";
      ...
    }
    internal_power() {
      related_pin : "A0";
      rise_power(energy_template_5x5) {
        ...
      }
      fall_power(energy_template_5x5) {
        ...
      }
    }
    ...
}
```

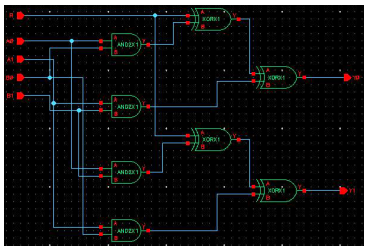


Figure 5.8 Schematic of AND2X1t1

```
module AND2t1 ( Y0, Y1, A0, A1, B0, B1, R );

  input B1;
  output Y0;
  input A1;
  output Y1;
  input A0;
  input B0;
  input R;

  wire net21, net22, net23, net24, net25, net26;

  AND2X1 I0(.A(A0), .B(B0), .Y(net22));
  AND2X1 I1(.A(A1), .B(B1), .Y(net24));
  AND2X1 I2(.A(A0), .B(B1), .Y(net21));
  AND2X1 I3(.A(A1), .B(B0), .Y(net23));
  XOR2X1 I4(.A(R), .B(net22), .Y(net26));
  XOR2X1 I5(.A(net26), .B(net24), .Y(Y0));
  XOR2X1 I6(.A(R), .B(net21), .Y(net25));
  XOR2X1 I7(.A(net25), .B(net23), .Y(Y1));

endmodule
```

Figure 5.9 Verilog description of AND2X1t1

```
// Generated by Cadence Encounter(R) RTL Compiler v10.10-s209_1

// Verification Directory fv/AND2t1

module AND2t1(Y0, Y1, A0, A1, B0, B1, R);
  input A0, A1, B0, B1, R;
  output Y0, Y1;
  wire A0, A1, B0, B1, R;
  wire Y0, Y1;
  wire n_0, n_1, n_2, n_3, n_4, n_5;
  XOR2X1 g139(.A (n_4), .B (n_1), .Y (Y0));
  XOR2X1 g140(.A (n_5), .B (n_0), .Y (Y1));
  XOR2X1 g141(.A (n_2), .B (R), .Y (n_5));
  XOR2X1 g142(.A (n_3), .B (R), .Y (n_4));
  AND2X2 g144(.A (B0), .B (A0), .Y (n_3));
  AND2X2 g145(.A (A0), .B (B1), .Y (n_2));
  AND2X2 g146(.A (B1), .B (A1), .Y (n_1));
  AND2X2 g143(.A (B0), .B (A1), .Y (n_0));
endmodule
```

Figure 5.10 Synthesized logic design

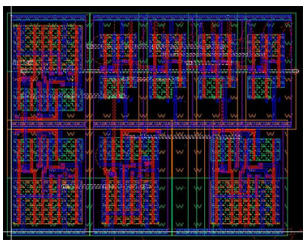


Figure 5.11 Layout of AND2X1t1

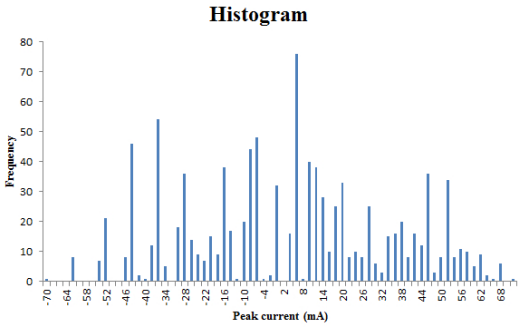


Figure 5.14 Peak currents of **NAND2X1t1**

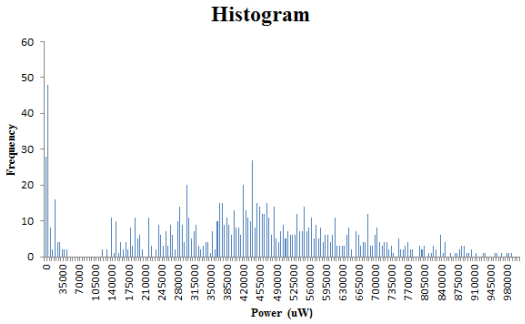


Figure 5.15 Powers of **NAND2X1t1**

Figure 5.16: Distribution of powers and peak currents of **NAND2X1t1**

output transition ($0 \to 0, 0 \to 1, 1 \to 0$ and $1 \to 1$) and the powers and peak currents in each group were averaged. If there is no dependency of power consumption on the input pattern, the logic gate has resistance against power analysis attacks. In other words, if it is difficult to distinguish averaged powers and peak powers of each group, the logic gate is robust. Table 5.2 shows the averaged power consumption, peak current and the number of cases of **NAND2X1t1** in each group according to output transition. The powers and peak currents are almost equal so that it is difficult to distinguish. We utilize the ratio of standard deviation(σ) to average(μ) called the coefficient of variation in order to quantify the dependency or robustness. The larger the value the larger dependency on output transition(or input pattern) or the smaller robustness against power analysis attacks. The coefficient of variation of **NAND2X1t1** is too smaller than the coefficient of variation of **NAND2X1**. Fig. 5.16 shows the distribution of power consumptions and peak currents of **NAND2X1t1**. Table 5.3 5.4 5.5 5.6 5.7 show power consumption and peak current of **AND2X1t1**, **NOR2X1t1**, **OR2X1t1**, **XOR2X1t1** and **XNOR2X1t1**, respectively.

Table 5.2: Power consumption of **NAND2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4194.55	0.719	64
$0 \rightarrow 1$	4173.27	0.745	192
$1 \rightarrow 0$	4194.40	0.668	192
$1 \rightarrow 1$	4178.08	0.701	576
Average(μ)	4185.08	0.709	-
Standard deviation(σ)	121.73	0.001	-
$\frac{\sigma}{\mu}$	0.029	0.0014	-

Table 5.3: Power consumption of **AND2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4209.30	0.717	576
$0 \rightarrow 1$	4230.54	0.657	192
$1 \rightarrow 0$	4207.52	0.725	192
$1 \rightarrow 1$	4225.71	0.701	64
Average(μ)	4215.76	0.699	-
Standard deviation(σ)	77.15	0.0009	-
$\frac{\sigma}{\mu}$	0.018	0.0013	-

5.4 Example : SBOX design

We implemented the AES S-Box through our proposed SCA-secure design methodology for a preliminary validation. The AES S-Box operation of the AES encryption or decryption in the first round or last round is especially vulnerable to DPA attacks [Mangard et al. (2005), Prouff and Rivain (2007)]. The vulnerable AES S-Box should be synthesized with t-private primitives into a secure layout with our design flow. As a baseline, insecure AES S-Box based

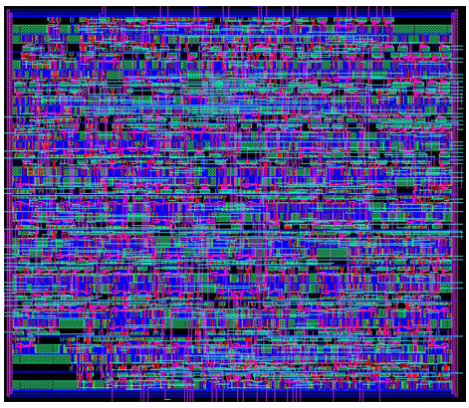


Figure 5.17: Layout of the secure AES S-Box

Table 5.4: Power consumptions of **NOR2X1t1** (45 *nm* process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4807.01	0.711	576
$0 \rightarrow 1$	4836.72	0.709	192
$1 \rightarrow 0$	4786.24	0.699	192
$1 \rightarrow 1$	4864.25	0.712	64
Average(μ)	4823.55	0.708	-
Standard deviation(σ)	34.13	0.0059	-
$\frac{\sigma}{\mu}$	0.007	0.008	-

Table 5.5: Power consumption of **OR2X1t1** (45 *nm* process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	4894.25	0.703	64
$0 \rightarrow 1$	4786.24	0.711	192
$1 \rightarrow 0$	4836.72	0.698	192
$1 \rightarrow 1$	4807.01	0.722	576
Average(μ)	4831.06	0.709	-
Standard deviation(σ)	46.95	0.104	-
$\frac{\sigma}{\mu}$	0.009	0.014	-

on composite finite field proposed by Satoh *el al.* [Satoh et al. (2001)] is implemented. It is re-synthesized with t-private re-synthesis using *RTL Compiler*. After t-private synthesis, the cell area and critical path delay are compared to the reference baseline design. The cell area increases by a factor 5.77 and and delay goes up by a factor 1.69 as compared to the reference design. The result of the layout shows that the die size of the secure S-Box is 4.37 times larger. But the DPA security metric (σ/μ) is reduced by 59% and it has robustness against the first order probing attack. Table 5.8 shows the comparison of the secure and insecure S-box designs. Fig. 5.17 shows the layout of the secure AES S-Box.

5.5 Conclusion

In this chapter, SCA resistant hardware implementation for FPGA and ASIC design has been proposed using t-private logic circuits. The standard cell library including t-private logic circuits can be used for logic synthesis, place & route and physical layout. Vulnerable modules to be flagged by SCA security metrics should be re-synthesized with t-private logic cells. After

Table 5.6: Power consumption of **XOR2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	1078.51	0.358	64
$0 \rightarrow 1$	1077.23	0.339	64
$1 \rightarrow 0$	1076.82	0.327	64
$1 \rightarrow 1$	1077.46	0.379	64
Average(μ)	1077.51	0.351	-
Standard deviation(σ)	0.72	0.023	-
$\frac{\sigma}{\mu}$	0.0007	0.065	-

Table 5.7: Power consumption of **XNOR2X1t1** (45 nm process)

Transition of output	Power consumption (nW)	Peak Current (mA)	Number of Transitions
$0 \rightarrow 0$	997.12	0.388	64
$0 \rightarrow 1$	997.14	0.375	64
$1 \rightarrow 0$	997.62	0.328	64
$1 \rightarrow 1$	998.33	0.370	64
Average(μ)	997.55	0.365	-
Standard deviation(σ)	0.567	0.026	-
$\frac{\sigma}{\mu}$	0.0005	0.071	-

the physical layout, SCA vulnerability of the hardware implementation can be verified by security metrics and simulating attacks.

Table 5.8: Comparison of insecure and secure S-Box

	cell area(μm^2)	delay(ns)	σ/μ
insecure	332.23	0.427	0.48
secure	1919.96	0.723	0.07

CHAPTER 6. t-PRIVATE SYSTEMS: UNIFIED PRIVATE MEMORIES AND COMPUTATION

6.1 Introduction

The goal of countermeasures against side channel attacks is to significantly reduce or remove the correlation between side channel leakage and the data or state processed by the computational system. A representative approach to counteract side channel attacks is to mask intermediate values with randomized bits at the gate level. Ishai *et al.* [Ishai et al. (2003)] proposed t-private circuit using such a masking method. They assume that an adversary can probe or observe up to t nodes in the circuit. Their assumption is that the adversary is perfect, and hence able to probe the circuit state of the logic with 100% certainty. The Ishai's t-private circuits need at least t random bits to ensure zero correlation between t probed nodes each clock cycle. This makes information loss to the adversary equal to 0.

t-private logic only targets the privacy of computation. However, cryptographic systems also include some memory, particularly, memories that hold private keys which are typically Read Only Memory (ROM). Many secret keys associated with a cryptographic system are stored in ROMs. For instance, hundreds of 1024-bit RSA private keys are not uncommon for a Trusted Platform Module (TPM) [Group (2013)]. ROMs are especially vulnerable to t-probing adversary of Ishai since their state does not change over time unlike computation. Moreover, these keys in memory can be targeted directly by physical attacks [Samyde et al. (2002)]. The adversary with physical access to the secret key part of the chip can succeed even if power has been turned off. The physical access based attacks slice the silicon until individual transistors are exposed by a Focused Ion Beam (FIB). An electron microscope is used to examine the silicon. Halderman *et al.* [Halderman et al. (2008)] proposed "cold-boot attack" which is a

method to extract a significant fraction of data stored in a powered-off memory (e. g. DRAM) by cooling the chip to around $-50°C$. Valamehr *et al.* [Valamehr et al. (2012)] developed several masking methods to prevent such memory attacks. The simplest of them is Ishai's [Ishai et al. (2003)] t-private coding applied to memory resident data. The key idea is that the secret key (x_i) does not need to be stored in the memory in its original form. Instead, a $t+1$-tuple $[r_1, r_2, \ldots, r_t, x_i \oplus r_1 \oplus \cdots \oplus r_t]$ is stored. We call this memory masking with t random bits a *t-private memory*. An adversary must learn all the t random bits and the encoded bit in order to reveal even a single bit of the secret key. The adversary attack model for ROM is based on the persistent physical access attack - not the transient probing attack for computational logic. The memory attack has statistical observation limitations. Therefore, Valamehr *et al.* [Valamehr et al. (2012)] assume that it succeeds only with probability p for each bit. Unlike Ishai's perfect secrecy analysis model, they define the success probability P_{succ} of this memory attack as a new figure of merit. It captures the event that at least one bit of the secret key has been learned. Even though a successful outcome of P_{succ} event does not break a cryptographic system, the possible key space can be reduced considerably when other side channel attacks are combined.

Practical computing systems consist of both memory and computational logic components. In order to build a t-private system, we need both a t-private memory and t-private logic that integrate seamlessly. Ishai's t-private scheme is not the most efficient one when applied to memory protection. Most of Valamehr's memory protection schemes [Valamehr et al. (2012)] are not *computable* in the sense that a computational logic schema does not exist within the coded domain (unlike Ishai scheme). These stored coded keys have to be decoded first before being used for computation, hence exposing them to probing attacks. This is a big weakness. In this paper, we develop a unified computable coding scheme applicable to both memory and computation logic. This scheme is more efficient than Valamehr's schemes in their memory analysis framework. It also shows zero information loss in the Ishai's analysis framework. We believe that our proposed coding scheme is an ideal candidate to build t-private systems unifying the memory and computing logic. In summary, this chapter makes the following contributions:

1) We analyze the storage overhead and the success probability (P_{succ}) of various t-private memory schemas within a unified framework that is easier to understand than Valamehr's. However, it may overestimate P_{succ}. We also quantify and describe a trade-off between these two attributes – storage overhead and P_{succ}.

2) We introduce a new notion of *computable* encoding method for t-private memories to capture the schemes which can compute with the encoded keys using a complementary t-private logic. We also propose a new, computable, t-private, inspection resistant memory with a corresponding computable encoding method. This new approach requires new t-private logic combinational gates which are more efficient than Ishai's [Ishai et al. (2003)] t-private circuits in their use of random bits without any loss of privacy.

3) We propose new combinational logic circuits suitable for our new memory scheme.

We define our adversary model and the notation (variables/parameters used) in Section 6.2. Our new more general analysis of t-private memories is presented in Section 6.3. Section 6.4 develops our proposed t-private memory scheme. Logic schema for our proposed memory is presented in Section 6.5. Hardware implementation results are presented in Section 6.6. Finally, Section 6.7 concludes the paper.

6.2 Assumptions and Notation

We assume that the memory leaks information in contrast to Micali's paper [Micali and Reyzin (2003)] in which they assume that only computation leaks information. An adversary conducts experiments to reveal the bits stored in the memory with a measurement apparatus. Let $\mathcal{L}$ be the leakage function selected by an adversary. The value of leakage of any bit x_i in the memory $\mathcal{M}$ is converted to the finite field $GF(2)$ based on the ability of an adversary: $f : \mathcal{L}(x_i) \rightarrow \{0, 1\} \quad \text{for} \quad x_i \in \mathcal{M}$.

We assume that an adversary has limited capability to learn any memory resident bit exactly due to noisy measurement apparatus. Hence, we define the limited leakage probability of a bit as $\Pr[f(\mathcal{L}(x_i)) = x_i] = p \quad \forall x_i \in \mathcal{M}$.

Table 6.1: Variables used in this chapter

k	key length
p	leakage probability for 1 bit
P_{succ}	probability of successful attack
r_i	random bit
x_i	one-bit secret key
t	the number of random bits
t_p	the number of probing nodes per clock cycle
n	the number of keys
c	the number of bits to be stored per key
$\mathbf{T}$	random bit matrix
T_{ij}	the ith row and jth column element of $\mathbf{T}$
$\vec{a} = [a_1, \ldots, a_t]$	a binary vector
$\bar{x}$	complement of x
$\wedge$	bit-wise AND operation

This p is the characteristic of the memory (encoding) schema. If adversary's target is computational circuit $\mathcal{C}$, our assumption is the same as Ishai's adversary model [Ishai et al. (2003)]. In other words, an adversary can probe t_p nodes every cycle: $\Pr[f(\mathcal{L}(y_i)) = y_i] = 1 \quad \forall y_i \in Y,\ Y \subset \mathcal{C},\ |Y| = t_p$.

A *memory attack* is a set of such experiments that are possibly adaptively controlled. We assume that the goal of a memory attack is to reveal at least one bit in the memory with probability 1. Success probability of a memory attack captures this goal.

Definition 18 (success probability)**.** *We define the success probability P_{succ} of a memory attack as the probability that at least one bit of the original secret key has been revealed.*

Memory may store multiple keys with the same key length k. The parameters/variables of the memory schema, adversary experiments, and memory attacks are defined in Table 6.1. If not otherwise stated, these variables hold for the rest of the chapter.

6.3 t-Private Memory: Schemas, Architecture, and Analysis

The k raw bits of a key $[x_k, x_{k-1}, \ldots, x_1]$ can be stored in memory in many ways. The t-privacy schemes could conceivably be transistor level schemes. However, encoding schemes applied at the write-port of a memory are more obvious and effective. A memory schema is a pair of encoding & decoding functions for memory. The base case is to do nothing - just store and retrieve the raw bits - with a schema of the identity function. All the following memory schemas except for t-private system are from Valamehr *et al.* [Valamehr et al. (2012)]. The unified analysis is ours.

A bit x_i of the secret key can be hidden by creating $t+1$ random shares using t random bits $[r_1, r_2, \ldots, r_t, x_i \oplus r_1 \oplus r_2 \oplus \cdots \oplus r_t]$ where r_i's are random bits. The t random bits constitute t shares. The $(t+1)st$ share is derived by an XOR of the t random bits and the original bit x_i.

The easiest memory architecture for the secrecy is to store all the $t+1$ share bits of a raw bit of the secret key. Therefore the total number of stored bits for a secret key of length k is $k(t+1)$. In this schema, each key bit uses a different set of t random bits. The set of random bits can be re-used or shared between various key bits. Depending on this reuse and sharing of random bits, the storage overhead and the success probability of the memory attack can vary. There are four memory schemes in [Valamehr et al. (2012)] which will be analyzed in this section (all except the dynamic matrix scheme using hash function). Fig. 6.6 shows these architectural memory schemes.

6.3.1 Original memory scheme without secrecy

Original memory refers to raw memory without any protection against memory attacks. The total number of bits stored for the n secret keys with key length k is nk. This value is the storage reference/baseline. We define the storage overhead as the ratio of the number of bits used for the secret keys storage to the storage reference. The success probability P_{succ} of memory attacks is $1-(1-p)^k$, where $(1-p)^k$ is the probability of the adversary experiments failing on all of the k key bits.

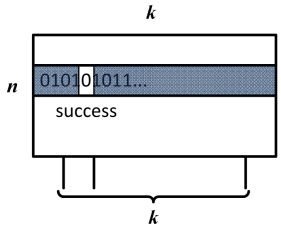


Figure 6.2 The original memory scheme

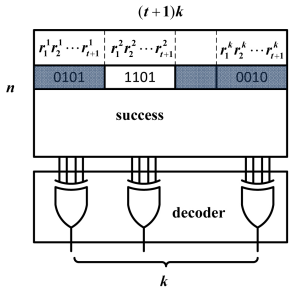


Figure 6.3 The t-private memory scheme

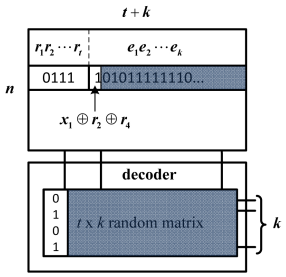


Figure 6.4 The t-private memory scheme with a random matrix

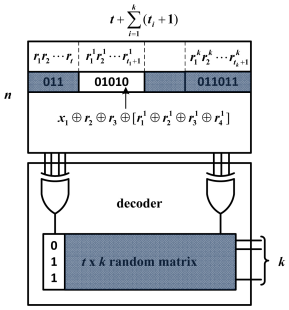


Figure 6.5 The hybrid memory scheme

Figure 6.6: 4 architectural memory schemes

6.3.2 t-private memory scheme

Each bit x_i of the secret key is represented by t random bits and the encoded bit $e_i = x_i \oplus r_1 \oplus \ldots \oplus r_t$ which are stored in the memory. Each key bit uses its own set of t random bits. Total number of bits stored for n secret keys is $cn = (t+1)k \cdot n$ and therefore the storage overhead is $t+1$. The success probability is

$$P_{succ} = 1 - (1 - p')^k \tag{6.1}$$

where $p' = p^{t+1}$, which is the probability that an adversary learns t random bits and the encoded bit to reveal x_i. p' is less than p since $0 \leq p \leq 1$. As noted earlier, this scheme mirrors the t-private circuits introduced in Ishai *et al.* [Ishai et al. (2003)].

6.3.3 t-private memory scheme using a random matrix T

The straightforward t-private memory requires t random bits per key bit. This may be an unreasonably large random bit overhead. This scheme attempts to reduce the number of

random bits needed for the entire schema. Randomly selected t_i random bits $R_i = \{r_j | r_j \in R, |R_i| = t_i\}$ from a set of t random bits $R = \{r_1, r_2, \ldots, r_t\}$ per key bit are used to encode each bit x_i of the secret key. The encoded bit e_i of x_i is $x_i \oplus \left[\bigoplus_{r_j \in R_i} r_j\right]$. The position/index j of randomly selected t_i random bits are stored in a fixed $t \times k$ random matrix $\mathbf{T}$. For example, if r_1, r_2, r_5 are randomly selected for encoding x_1, the first column T_1 of the random matrix $\mathbf{T}$ is $[1, 1, 0, 0, 1, 0, ...]^T$. The random matrix $\mathbf{T}$ is used for decoding $x_i = e_i \oplus \left[\bigoplus_{j=1}^{t} r_j \cdot T_{ji}\right]$. In this case, c is $t + k$ and total number of bits stored for n secret keys including a $t \times k$ random matrix table is equal to $(t + k)n + tk$. The storage overhead is

$$\frac{(t+k)n + tk}{nk} = 1 + t\left(\frac{1}{n} + \frac{1}{k}\right).$$

In order to reveal a single secret key-bit x_i, all of the t random bits and the ith column T_i of the random matrix $\mathbf{T}$ should be required:

$$x_i = e_i \oplus \left[\bigoplus_{j=1}^{t} r_j \cdot T_{ji}\right], \quad \text{where } r_j \in R, T_{ji} \in T_i.$$

The failing cases of our memory attack scenario are divided into two cases. The first case is that an adversary does not know all the random bits. The second case corresponds to the case that an adversary does not know the ith column of the random matrix $\mathbf{T}$ even though all the random bits are known. Note that we assume that the leakage probability of the matrix $\mathbf{T}$'s random bit is also p, which is independently distributed. Thus, the failure probability P_{fail} of this attack is equal to the sum of the probabilities of two cases . The success probability P_{succ} is given by the following equations:

$$\begin{aligned} P_{succ} = 1 - P_{fail} &= 1 - \{ \underbrace{1 - p^t}_{\text{the first case's probability}} + \underbrace{p^t(1 - p^{t+1})^k}_{\text{the second case's probability}} \} \\ &= p^t\{1 - (1 - p^{t+1})^k\}. \end{aligned} \tag{6.2}$$

Compared with Eq (6.1), the success probability of the t-private memory scheme using a random matrix is p^t factor less than the success probability of the t-private scheme for the same t.

6.3.4 Hybrid memory scheme

The hybrid scheme is a combination of t-private memory scheme and t-private memory scheme using a fixed random matrix. This scheme is devised in [Valamehr et al. (2012)] in order to minimize p_{succ} per random bit. Intuitively, it uses a few of the t bits to reduce p with the classical t-private scheme. The rest of the t private bits are used in a random matrix schema. The details of the hybrid schema and analysis in [Valamehr et al. (2012)] are ambiguous. In the following, we have chosen a version of many possible designs for the hybrid schema.

The number of random bits t_i to encode each secret key bit x_i with the t-private scheme is a parameter individualized to each x_i. We let the set of the random bits be $R'_i = \{r_1^i, r_2^i, \ldots, r_{t_i}^i\}$. Another set of random bits per secret key $R = \{r_1, r_2, \ldots, r_t\}$ is required for the encoding method with a $t \times k$ random matrix $\mathbf{T}$. Each secret key bit x_i can be encoded by the following equation:

$$e_i = x_i \oplus \{r_1^i \oplus \cdots \oplus r_{t_i}^i\} \oplus \left[\bigoplus_{r_j \in R_i} r_j \right] \quad \text{for } 1 \le i \le k$$

where R_i is a randomly selected subset of $R = \{r'_1, \ldots, r'_t\}$.

The storage overhead is

$$\frac{n\left[t + \sum_{i=1}^{k}(t_i+1)\right] + tk}{nk} = 1 + t\left(\frac{1}{n} + \frac{1}{k}\right) + \frac{1}{k}\sum_{i=1}^{k} t_i.$$

The failing cases for an adversary are also divided into two cases as in the t-private scheme using a random matrix. The first case is that an adversary does not know all of the t random bits $\{r_1, r_2, \ldots, r_t\}$ to encode with the random matrix. The second case is that an adversary does not know the ith column of the random matrix $\mathbf{T}$ and all t_i random bits for the t-private encoding even though (conditioned on) all the random bits $\{r_1, r_2, \ldots, r_t\}$ are known. The success probability P_{succ} is

$$\begin{aligned} P_{succ} = 1 - P_{fail} &= 1 - \{ \underbrace{1 - p^t}_{\text{the first case's probability}} + \underbrace{p^t \prod_{i=1}^{k}(1 - p^{t_i+t+1})}_{\text{the second case's probability}} \} \\ &= p^t \left[1 - \prod_{i=1}^{k}(1 - p^{t_i+t+1}) \right]. \end{aligned} \tag{6.3}$$

The t-private memory scheme with a random matrix is the special case of this hybrid memory scheme when all t_i for $1 \leq i \leq k$ is zero. Compared to the t-private memory scheme with a random matrix when both t is equal and all t_i's are the same, the success probability of the hybrid scheme decreases slightly since p^{t+1} in Eq. (6.2) is larger than p^{t_i+t+1} Eq. (6.3). But the storage overhead increases by t_i.

6.3.5 Comparison

Table 6.2 shows the storage overhead and the success probability of the 4 architectural schemes. We assume that the key length k is 128 bits and the number of secret keys n is 10 and the leakage probability of each bit p is 0.9. Fig. 6.10 shows the storage overhead and the success probability of the t-private scheme, the t-private scheme with a random matrix and the hybrid memory scheme with $t_i = 10$ parametrized by the number of random bits t. Compared to the t-private memory scheme with a random matrix, the hybrid memory scheme does not have any advantage since the storage overhead is larger without a significant reduction in the success probability. In the following sections, our proposed memory scheme will be compared to the t-private memory scheme with a random matrix.

6.4 New Approach

Note that all the encoding schemes in Section 6.3 except for the classical t-private memory scheme require the stored keys to be decoded before they can be used in a cryptographic computation (such as AES encryption). A more secure and private system can be designed if the computation with the key is also implemented as private logic (along the lines of Ishai

Table 6.2: The storage overhead and the success probability of the 4 architectural schemes

	Original	t-private	t-private with $\mathbf{T}$	Hybrid
Storage overhead	1	$1+t$	$1+t\left(\frac{1}{n}+\frac{1}{k}\right)$	$1+t\left(\frac{1}{n}+\frac{1}{k}\right)+\frac{1}{k}\sum_{i=1}^{k} t_i$
P_{succ}	$1-(1-p)^k$	$1-(1-p^{t+1})^k$	$p^t\{1-(1-p^{t+1})^k\}$	$p^t\left[1-\prod_{i=1}^{k}(1-p^{t_i+t+1})\right]$

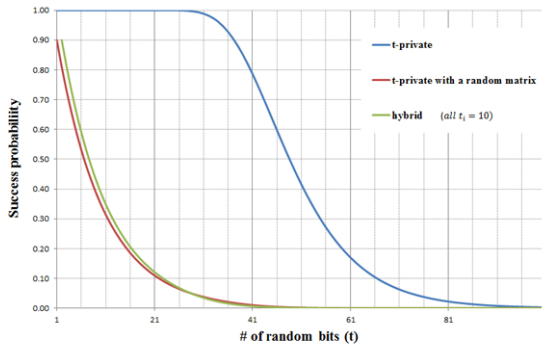


Figure 6.8 The success probability

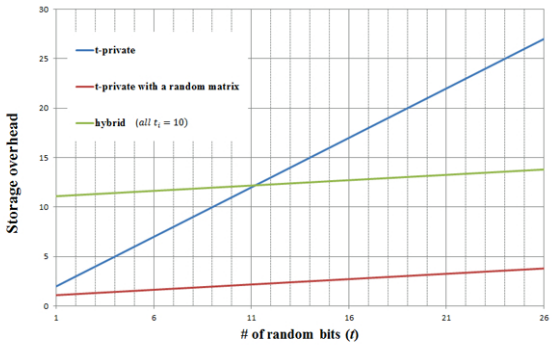


Figure 6.9 The storage overhead

Figure 6.10: Comparison between t-private scheme, t-private scheme with a random matrix and the hybrid scheme when $p = 0.9, k = 128, n = 10, t_i = 10$

$$
\begin{array}{l}
\quad X\ r_t r_{t-1} \ldots r_1 \\
+\ c_{t+1} c_t c_{t-1} \ldots c_1 \\
\hline
\quad e_{t+1} e_t e_{t-1} \ldots e_1
\end{array}
\qquad\qquad
\begin{array}{l}
\quad e_{t+1} e_t e_{t-1} \ldots e_1 \\
-\ c_{t+1} c_t c_{t-1} \ldots c_1 \\
\hline
\quad d_{t+1} d_t d_{t-1} \ldots d_1 = \\
\quad X\ r_t r_{t-1} \ldots r_1
\end{array}
$$

Figure 6.11: t-Private: (Left) Encoding; (Right) Decoding

scheme [Ishai et al. (2003)]). A memory encoding scheme that does not require the key to be decoded so that the key can participate in a computation implemented with private logic is called a *computable* encoding or schema. In such cases, a private logic family consistent with the memory encoding must exist. In a memory schema that is not computable, the decoded key can be attacked dynamically in flight. The only attacks that a non-computable memory schema prevents against are static memory attacks such as chip slicing based observation of transistor fatigue.

t-private encoding is obviously a computable schema. The t-private storage can be used directly in the t-private encryption/decryption implementation without additional decoding. Hence, the t-private memory scheme should be selected in order to prevent the adversary from attacking the raw key at the decoding step even though it does not have the best success probability and storage overhead tradeoff.

Basic Encoding Scheme: t-private implementations require many random bits - they do not share/reuse random bits (unlike the random matrix schema). They pose a t^2 factor area overhead and a factor t delay overhead. Our goal was to come up with a computable version of random matrix method. Alternately, we need a scheme that reuses random bits in a t-private logic implementation. We propose the computable and t-private encoding with these properties. We could use addition like invertible function with the t-private masking method to reduce the success probability. Note that such a function is not commutative in the bits of its operand. In other words, unlike the t random bits in Ishai's t-privacy schema, the order of these bits within the coding operand matters. Each ordering of t random bits gives a different seed and hence a different encoding. This allows any permutation of t random bits to give a different random seed from the encoding perspective. This results in a possibility of $t!/(a!b!) \approx t!/((t/2)! * (t/2)!)$ reuses of t random bits, where a is the number of 1's and b is the number of 0's of the t random bits.

Fig. 6.11 shows the basic idea. We add two $t+1$-bit words for encoding. One operand is derived by concatenating the bit to be encoded x with t random bits $r_t, r_{t-1}, \ldots r_1$. This word is added to another random constant c (either one c per chip or one c per x). Note that different

permutations of the t random bits $r_{i_t}, r_{i_{t-1}}, \ldots r_{i_1}$ lead to different encoded result when added to c. Decoding consists of simply subtracting c from the encoded word $e_{t+1}e_t \ldots e_1$. The most significant bit of the decoded word is x.

Refined Encoding Schema: The basic encoding schema has some flaws that expose the bit x when forming complex entangling gates such as AND and OR as discussed in Section 6.5. In order to fix that, instead of x at the MSB of arithmetic word with random bits, we use the Ishai code $x \oplus r_t \oplus \cdots \oplus r_1$.

We define the computable and t-private encoding for x_i (bit to be coded) as follows:

$$\vec{e_i} = Encode(x_i) = [x_i \oplus r_t^i \oplus r_{t-1}^i \oplus \cdots \oplus r_1^i, \vec{r_i}] + \vec{c_i}$$

where $\vec{r_i}$ and $\vec{c_i}$ are vectors/words of t random bits $[r_t^i, r_{t-1}^i, \ldots, r_1^i]$ and constant bits $[c_{t+1}^i, c_t^i, \ldots, c_1^i]$ respectively. Note that this schema uses a constant word per x_i. We form an arithmetic word comprising of t random bits and x_i. By placing x_i at the most significant end we allow all the t random bits to effect its encoding. A simpler encoding would have added $[x_i, r_t, \ldots, r_1]$ to a constant vector per chip or per computation session. Note that since the constant vector $\vec{c}$ is constant over longer periods - entire computation session, entire boot-up phase, to be conservative, it may not contribute to the entropy of encoding. We must assume that the adversary knows such a persistent $\vec{c}$.

The decoding can then be done as follows:

$$\vec{d_i} = Decode(\vec{e_i}) = \vec{e_i} - \vec{c_i} = [x_i \oplus r_t^i \oplus \cdots \oplus r_1^i, r_t^i, \ldots, r_1^i].$$

Most significant bit of $\vec{d_i}$ is $x_i \oplus r_t^i \oplus \cdots \oplus r_1^i$. The decoded vector $\vec{d_i}$ can be directly connected to t-private encryption/decryption logic. This computable and t-private encoding method does not reveal the original key bit after this decoding process. Algorithm 2 represents our computable t-private encoding/decoding method. Note that this algorithm creates all m reuses of each bit within the encoding of the same key. Such a localized reuse may not be optimal in practice. It is presented in the algorithm for its simplicity. In practice, for CAD, we will likely incorporate global randomized reuse. Also note that we have used a random instance of a permutation of t

bits π_r picked uniformly from $t!$ space. $\pi_r(i) = j$ maps the ith bit position to jth bit position. Fig. 6.12 shows our proposed computable and t-private memory scheme.

Since $t+1$ encoded bits per key bit are stored in the memory in this scheme, the storage overhead is

$$\frac{nk(t+1)}{nk} = t+1.$$

Algorithm 2 `Computable` t-`private memory encoding/decoding scheme`

Encoding
Input : A k-bit secret key $\vec{x} = [x_k, x_{k-1}, \ldots, x_i, \ldots, x_1]$; $g = \lceil k/m \rceil$ distinct t-bit random vectors $\vec{r^0} = [r_t^0, r_{t-1}^0, \ldots, r_1^0]$, $\vec{r^1} = [r_t^1, r_{t-1}^1, \ldots, r_1^1]$, $\ldots$, $\vec{r^{g-1}} = [r_t^{g-1}, r_{t-1}^{g-1}, \ldots, r_1^{g-1}]$; constant vector (per chip or per computation session) $\vec{c} = [c_{t+1}, c_t, \ldots, c_1]$
Output : Encoded secret key bit vectors, $\vec{e_i}$ for $i = 1, 2, \ldots, k$ such that $e(\vec{x}) = \vec{e_k}\vec{e_{k-1}} \ldots \vec{e_1}$
for $i = 1 \rightarrow k$ **do**
$j \leftarrow k \,\%\, g$
Key bit x_i is XORed with the t random bits in jth random vector : $y_i = x_i \oplus r_t^j \oplus r_{t-1}^j \oplus \cdots \oplus r_1^j$
Concatenate XORed bit y_i with a randomly picked permutation of t bits π_r : $y_i || \pi_r(\vec{r^j}) = \left[y_i, r^j_{\pi_r^{-1}(t)}, r^j_{\pi_r^{-1}(t-1)}, \ldots, r^j_{\pi_r^{-1}(1)}\right]$
Add constant vector $\vec{c}$: $\vec{e_i} = \left[y_i, r^j_{\pi_r^{-1}(t)}, r^j_{\pi_r^{-1}(t-1)}, \ldots, r^j_{\pi_r^{-1}(1)}\right] + \vec{c}$
end for
Decoding
Input : Encoded secret key vectors, $\vec{e_i}$ for $i = 1, 2, \ldots, k$; constant vector $\vec{c}$
Output : Decoded secret key vectors, $\vec{d_i} = [y_i, r_t, \ldots, r_1]$ for $i = 1, 2, \ldots, k$
for $i = 1 \rightarrow k$ **do**
Subtract constant vector $\vec{c}$: $\vec{d_i} = [e_{t+1}^i, e_t^i, \ldots, e_1^i] - \vec{c} = [x_i \oplus r_t^j \oplus r_{t-1}^j \oplus \cdots \oplus r_1^j, r_t^j, r_{t-1}^j, \ldots, r_1^j]$ for $j = k \,\%\, g$
end for

Constant vector $\vec{c}$ storage/routing: The constant vector $\vec{c_i}$ need not to be stored in memory. Its lifetime is only from the producer gate to the consumer gate. It can be hardwired in the routing of wires from the producer gate to the consumer gate. For a per chip or per session constant $\vec{c}$, similar hardwiring will work with a bootup or session-startup initialization step. For a random choice of $\vec{c_i}$ per x_i, we assume that the adversary learns each bit with probability 0.5 randomly. This requires the adversary to conduct all possible 2^{t+1} $\vec{c_i}$ experiments to reveal

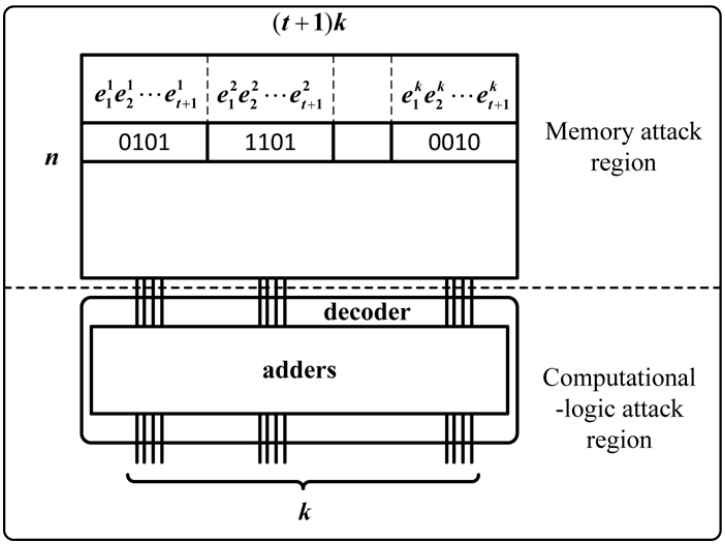


Figure 6.12: The proposed memory scheme

a key bit. The success probability P_{succ} then is

$$\frac{1}{2^{t+1}}(1-(1-p^{t+1})^k). \tag{6.4}$$

However, since the goal of this paper is to save on random bits, henceforth in this paper, we assume that $\vec{c}$ is a constant per chip or per computation session. Furthermore, the adversary knows $\vec{c}$. Hence we cannot use the entropy of $\vec{c}$ in our security analysis.

$$(1-(1-p^{t+1})^k). \tag{6.5}$$

If we assume instead the memory attack model with probability p to reveal each bit of $\vec{c_i}$ then the success probability is $P_{succ} = p^{t+1} \times (1-(1-p^{t+1})^k)$. Similarly, if we assume that the constant vector is fixed for the chip design or for each boot-up session, we give the benefit of doubt to the adversary leading to $P_{succ} = (1-(1-p^{t+1})^k)$. Effectively, this gives us two types of t-private systems: **(1)** ones with constant $\vec{c}$ with higher success probability but with lower number of random bits requirement (which is the one analyzed in the following), **(2)** constant $\vec{c_i}$ per x_i with lower success probability at the cost of higher number of random bits.

When a permutation of a vector of t random bits is reused upto m times for encoding other information/key bits, we need to consider two cases for revealing the coded bits. In the earlier analysis, we have assumed probability p for slicing attack to succeed at revealing a

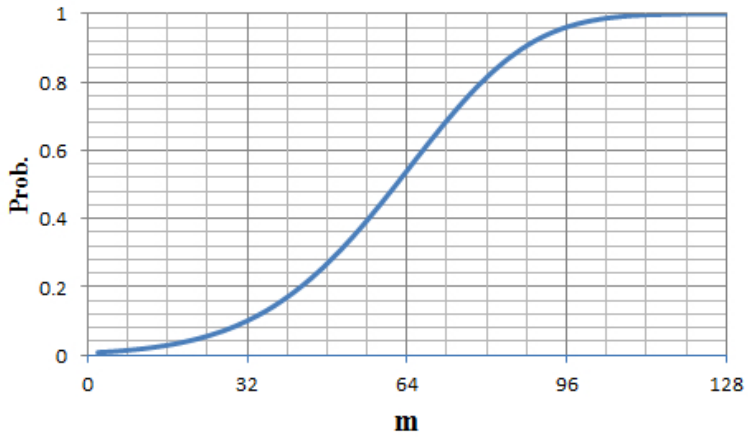


Figure 6.13: The success probability according to m reused random bits when $p = 0.9, t = 91$

specific coded bit b_i. The other possibility due to reuse is that another bit a_l might be revealed through slicing attack with probability p, and it is reused at the bit position of b_i. Eq. (6.5) should be changed into the following equation to account for such reuse:

$$P_{succ\_reuse} = \left(1 - (1 - (p + (1-p)q)^{t+1})^k\right) \tag{6.6}$$

where q is the probability that a reused bit a_l is revealed through slicing attack and is routed to the bit under consideration b_i.

$$q = 1 - \left(1 - \frac{p}{t}\right)^m. \tag{6.7}$$

In Eq. (6.7), $\frac{p}{t}$ is the probability that a reused bit b_i is revealed by slicing attack of another bit a_l. It results from the leakage/slicing attack success probability p of another bit a_l and the probability that the reused bit a_l is routed to b_i's position. Note that a random permutation π_r maps a bit position i to another bit position j with probability $1/t$ over all $t!$ permutations. When slicing memory inspection of a bit fails with probability $(1-p)$, the event that a reuse might reveal needs to be considered resulting in the success probability P_{succ_reuse} to increase by the factor of $(1-p)q$.

Fig. 6.13 shows the success probability parametrized by reuse factor m when p is 0.9 and t is 91. The success probability is 0.1 when the reuse factor m is 30. For $m = 86$, the success probability goes up to 0.9. Fig. 6.17 shows the success probability of our proposed

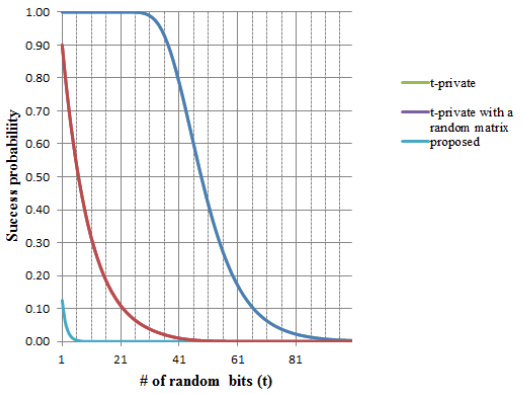


Figure 6.15 The success probability

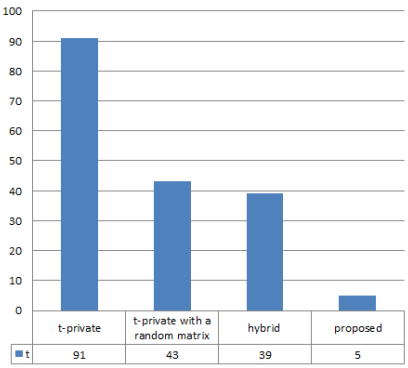


Figure 6.16 The number of random bits(t) when $P_{succ} = 0.0078$

Figure 6.17: Performance comparison between proposed scheme and t-private schemes

memory scheme and t-private schemes. Our proposed schema requires only 5 random bits for $P_{succ} = 0.0078$ as in Fig. 6.17.(b).

Now let us consider the complexity of the $t+1$-bit ripple carry adders used for encoding and decoding in terms of number of logic gates. Since one of the adder operands is a constant, a full adder bit-slice design can be made simpler than the typical full adder. If a constant bit b_0 is 0, the carry-out bit c_1 is $a_0 c_0$ where a_0 and c_0 is an input and a carry-in bit, respectively. The sum bit s_0 is $a_0 \oplus c_0$. If a constant bit b_0 is 1, the carry-out bit c_1 is $a_0 + c_0$ and the sum bit s_0 is $(a_0 \oplus c_0)'$. Only 2 logic gates are needed for a specialized full adder leading to total number of logic gates for the $t+1$-bit adder as $2(t+1)$.

6.5 New Computable And t-private Logic Schema And Gates

Consider an inverter $y = \bar{x}$. If x is encoded with our schema, the incoming $(t+1)$-tuple represents the encoding $(x, \vec{r_x}, \vec{c_x})$. The inverter needs to recode the output, however, with respect to the vector $(y, \vec{r_y}, \vec{c_y})$. This will require first decoding the incoming $(t+1)$-tuple and then recoding it. Had we used the basic encoding schema, this would have revealed x in the open temporarily, open to a probing attack. No bit x_i should be in-flight in the raw form even

momentarily creating a weak link. We overcome this by using $x_i \oplus r_1^i \oplus r_2^i \oplus \cdots \oplus r_t^i$ as MSB in addition.

With this scheme, the MSB of the decoded vector $\vec{d_i} = [x_i \oplus r_t^i \oplus \cdots \oplus r_1^i, r_t^i, \ldots, r_1^i]$ is identical to Ishai encoding of private circuits [Ishai et al. (2003)], and hence can be connected to Ishai's t-private combinational logic gates. The classical t-private scheme has t^2 area and t time overhead. We only save on the random bits by adopting this approach. We however propose a more efficient combinational logic using the decoded vectors which have the same functionality as the traditional logic operation with lower overhead.

6.5.1 AND operation

Let two encoded bit vectors be $\vec{e_1} = [x_1 \oplus r_t^1 \oplus \cdots \oplus r_1^1, \vec{r_1}] + \vec{c_1}$ and $\vec{e_2} = [x_2 \oplus r_t^2 \oplus \cdots \oplus r_1^2, \vec{r_2}] + \vec{c_2}$ from the memory. They are decoded by the decoder, which are denoted by $\vec{d_1}$ and $\vec{d_2}$. First, consider the simple case in which t is 1. Two decoded bit vectors are $\vec{d_1} = [x_1 \oplus r_1, r_1]$ and $\vec{d_2} = [x_2 \oplus r_1', r_1']$. The result of the AND operation should be $[x_1 \cdot x_2 \oplus r_1'', r_1'']$. How can we obtain the result and r_1'' ? Let us perform the following computation:

$$\begin{aligned}\vec{d_1} \wedge \vec{d_2} &= [(x_1 \oplus r_1) \cdot (x_2 \oplus r_1'), r_1 \cdot r_1'] \\ &= [x_1 \cdot x_2 \oplus r_1 \cdot x_2 \oplus x_1 \cdot r_1' \oplus r_1 \cdot r_1', r_1 \cdot r_1']\end{aligned}$$

$x_1 \cdot x_2 \oplus r_1 \cdot x_2 \oplus x_1 \cdot r_1' \oplus r_1 \cdot r_1'$ in the above equation should be changed into $x_1 \cdot x_2 \oplus r_1 \cdot r_1'$ in order to obtain desired result and thus additional computations are required to remove $r_1 \cdot x_2 \oplus x_1 \cdot r_1'$. We define the AND operation in this case ($t = 1$) as the following equations:

$$\begin{aligned}AND(\vec{d_1}, \vec{d_2}) &= [x_1 \oplus r_1, r_1]\ AND\ [x_2 \oplus r_1', r_1'] \\ &= [(x_1 \oplus r_1) \cdot (x_2 \oplus r_1') \underbrace{\oplus (x_1 \oplus r_1) \cdot r_1' \oplus (x_2 \oplus r_1') \cdot r_1}_{\text{additional computations}}, r_1 \cdot r_1'] \\ &= [x_1 \cdot x_2 \oplus r_1'', r_1'']\end{aligned}$$

where r_1'' is equal to $r_1 \cdot r_1'$.

Let us now increase the value of t to develop our intuition. Two decoded vectors are $\vec{d_1} = [x_1 \oplus \bigoplus r_j, \vec{r}]$ and $\vec{d_2} = [x_2 \oplus \bigoplus r_j', \vec{r'}]$. In this case, the AND operation is equal to the

following equation:

$$
\begin{aligned}
AND(\vec{d_1}, \vec{d_2}) &= [x_1 \oplus \bigoplus r_j, \vec{r}] \; AND \; [x_2 \oplus \bigoplus r'_j, \vec{r'}] \\
&= [(x_1 \oplus \bigoplus r_j) \cdot (x_2 \oplus \bigoplus r'_j) \underbrace{\oplus \left\{(x_1 \oplus \bigoplus r_j) \cdot (\bigoplus r'_j)\right\} \oplus \left\{(x_2 \oplus \bigoplus r'_j) \cdot (\bigoplus r_j)\right\}}_{\text{additional computations (6 operations)}}, \\
&\quad (\bigoplus r'_j) \cdot \vec{r}] \qquad (6.8) \\
&= \left[x_1 \cdot x_2 \oplus \left\{(\bigoplus r_j) \cdot (\bigoplus r'_j)\right\}, (\bigoplus r'_j) \cdot \vec{r}\right]
\end{aligned}
$$

where $\bigoplus r_j = r_1 \oplus r_2 \oplus \cdots \oplus r_t$ and $(\bigoplus r'_j) \cdot \vec{r} = [(r'_1 \oplus \cdots \oplus r'_t)r_1, \ldots, (r'_1 \oplus \cdots \oplus r'_t)r_t]$. The number of gates required is $t + 7$ for $t + 1$ AND gates and 6 additional operations. Thus, the area/gate complexity of this AND operation is $O(t)$. This is more efficient than Ishai's t-private model which has the area complexity of $O(t^2)$ [Ishai et al. (2003)]. Moreover, this computation can be performed in $O(\log t)$ time as opposed to $O(t)$ in the original private circuits.

6.5.2 OR operation

We define the OR operation as follows:

$$
\begin{aligned}
OR(\vec{d_1}, \vec{d_2}) &= [x_1 \oplus \bigoplus r_j, \vec{r}] \; OR \; [x_2 \oplus \bigoplus r'_j, \vec{r'}] \\
&= [\overline{\overline{(x_1 \oplus \bigoplus r_j)} \cdot \overline{(x_2 \oplus \bigoplus r'_j)}} \underbrace{\oplus \left\{\overline{(x_1 \oplus \bigoplus r_j)} \cdot (\bigoplus r'_j)\right\} \oplus \left\{\overline{(x_2 \oplus \bigoplus r'_j)} \cdot (\bigoplus r_j)\right\}}_{\text{additional computations (6 operations)}}, \\
&\quad (\bigoplus r'_j) \cdot \vec{r}] \qquad (6.9) \\
&= \left[(x_1 + x_2) \oplus \left\{(\bigoplus r_j) \cdot (\bigoplus r'_j)\right\}, (\bigoplus r'_j) \cdot \vec{r}\right]
\end{aligned}
$$

An OR gate is a logic dual of an AND gate. Hence, OR operation logic also has the same area complexity of $O(t)$. It has the same structure as the AND operation logic except for the additional NOT gates.

6.5.3 NOT operation

The NOT operation is modeled by the following equations:

$$\begin{aligned} NOT(\vec{d_i}) &= [(x_i \oplus r_1 \oplus \cdots \oplus r_t)', \vec{r}] \\ &= [x_i' \oplus \bigoplus r_j, \vec{r}] \end{aligned}$$

6.5.4 The perfect secrecy

The original secret bit x_i must not be revealed when the adversary probes $t_p \leq t$ nodes in a t-private logic circuit. The t-privacy parameter determines the bounds of probing experiments for perfect secrecy. In Ishai's privacy model, there is no grey zone analysis - you either have perfect secrecy ($p = 0$) or you are unacceptably compromised. We develop a t-private circuit privacy analysis consistent with our memory attack analysis. If the adversary can probe two nodes $(x_1 \oplus \bigoplus r_j^1)$ and $\bigoplus r_j^1$ in the proposed AND or OR logic circuit exactly, x_1 is leaked easily. Assuming that the adversary can access any circuit node equally likely with 100% certainty, the probability that x_1 is learned is given by the following equation:

$$\begin{aligned} P_{succ} &= \frac{\binom{t}{2}}{\binom{n}{t}} \\ &= \frac{t(t-1)t!(n-t)!}{2n!} \end{aligned}$$

where n is the number of total nodes. Since n is much larger than t generally, P_{succ} is very low. For example, when n and t are 100 and 10, respectively, P_{succ} is 2.6×10^{-12}. In order to make P_{succ} close to zero, $(x_i \oplus \bigoplus r_j) \cdot (\bigoplus r_j')$ which consists of two terms in Eq. (6.8) or Eq. (6.9) can be resolved into $\bigoplus\{(x_i \oplus \bigoplus r_j) \cdot r_j'\}$ which consists of t terms.

The perfect secret circuit is defined as a circuit that appears like a pseudo-random number generator. There is no appreciable (poly adversary limited or whatever other restrictions

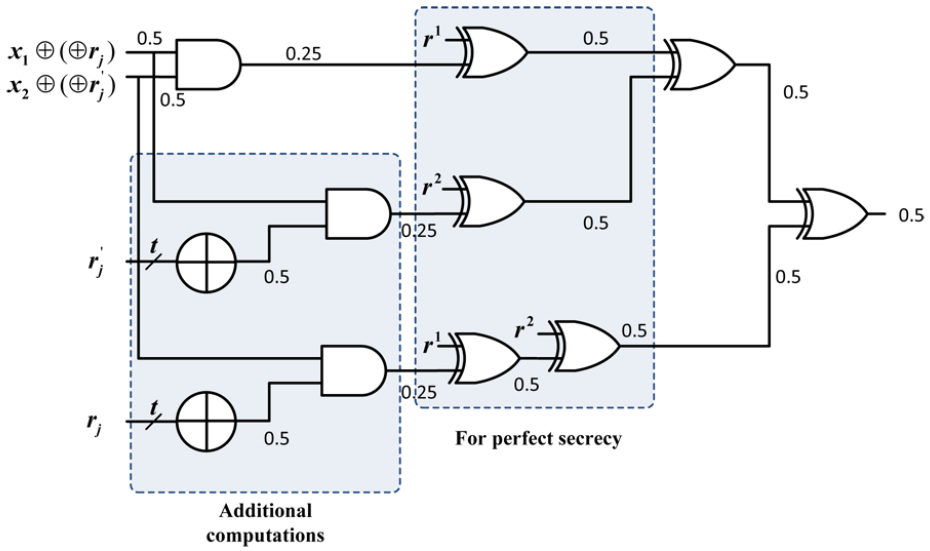


Figure 6.18: An output of AND operation for the perfect secrecy

are placed on the adversary) correlation between inputs and outputs. Given any input, the probability of any output vector should be the same. It does not depend on the input :

$$\Pr[y|x_i] = \Pr[y] \quad \forall x_i.$$

where x_i is the input and y is the output. This is the same property required of encryption functions. For example, the traditional AND gate does not have perfect secrecy since the output depends on inputs. AND-XOR network with a random bit has the perfect secrecy for inputs of AND gates [Park and Tyagi (2012)]. Fig. 6.18 shows the schematic of the first bit of the vector term in Eq. (6.8) which needs the perfect secrecy. For the perfect secrecy, additional XOR gates and new random bits are inserted. Numbers in the logic circuit represents the probability that the node is one. The probability that the output is one is always equal to 0.5, does not depend on inputs. Also, the vector $(\bigoplus r'_j) \cdot \vec{r}$ in Eq. (6.8) should be changed into $[(\bigoplus r'_j)r_t \oplus r''_1, (\bigoplus r'_j)r_{t-1} \oplus r''_1, \ldots, (\bigoplus r'_j)r_2 \oplus r''_{\lceil t/2 \rceil}, (\bigoplus r'_j)r_1 \oplus r''_{\lceil t/2 \rceil}]$ for the perfect secrecy. This technique can also be applied to OR logic circuit for the perfect secrecy in a similar manner. We compare the number of intermediate random bits for the perfect secrecy of three t-private AND circuits which are Ishai's t-private model, our earlier modified t-private model [Park and Tyagi (2012)] and computable t-private model. Table 6.3 shows the comparison of the number of intermediate random bits per AND/OR gate for our HOST scheme, Ishai's t-private

Table 6.3: Number of Random Bits Used for an AND Gate and for an N-gate Circuit

AND Gate	Modified t-private (HOST)	Ishai's t-private	Computable t-private	Computable t-private - perfect secrecy
# of random bits	$\lceil \frac{t+1}{2} \rceil = O(t)$	$\frac{t(t+1)}{2} = O(t^2)$	2	$\lceil \frac{t}{2} \rceil$
N-gate circuit	Modified t-private (HOST)	Ishai's t-private	Computable t-private	Computable t-private - perfect secrecy
# of random bits	Nt	Nt^2	$N * ((t/m) + 2)$	$N * ((t/m) + \lceil \frac{t}{2} \rceil + 2)$

scheme, proposed computable t-private without perfect secrecy, and proposed computable t-private with perfect secrecy. The last two rows show the total number of random bits used among these private schemes for a circuit with N gates.

6.6 Hardware Implementation

Table 6.4: Hardware Implementation on FPGA

	t-private	t-private with R.M	proposed computable and t-private
# keys	10	10	10
# bits of a key	128	128	128
t	63	19	4
P_{succ}	0.14	0.135	0.016
Block RAM	1024 * 80	(35*80) + (304*8)	80*80
# decoded bits per 1 clock	16	16	16
Input bits of decoder	64*16 = 1024	19+16+(19*16) = 339	80
# LUTs	208	25	16
Delay(ns)	1.926	1.998	0.931

We implemented t-private memories including the random matrix method and our proposed computable and t-private memory. We used Xilinx ISE tools for the synthesis and the target device is Xilinx Virtex-5 FPGA (XC5VFX70T-3FF1136). Table 6.4 shows the parameters and the number of used Block RAMs, LUTs and delay for each decoder. In case of t-private memory, 63 random bits are required for $P_{succ} = 0.14$. The stored bits of encoded keys in memory total $nk(t+1) = 10 * 128 * (63+1)$. Since the width of Block RAM in FPGA is limited to 1152 bits, we set the width of the Block RAM to be 1024. Thus, 16 decoded bits (1024 / 64) per 1 clock can be generated and 8 clock cycles are needed for decoding 1 key, which is the reference clock to compare used LUTs and delays for decoders of t-private memories. Since we set the total clock cycles for decoding a key to be 8, 35 bits which include 19 bits for random bits and 16 encoded bits of 16 secret-key bits are released from a block RAM and 304 bits (16×19) also

are output from another block RAM for a random matrix simultaneously.

Our proposed memory scheme has lower storage needs (only 7% of t-private memory) even though the success probability is almost 10% lower than the t-private memory. Also, the decoder of our proposed memory has lower area and time overhead – specifically it requires 92% lower area, 51% less delay and 36% less area, 53% less delay compared to t-private memory and t-private memory with a random matrix, respectively.

6.7 Conclusion

Side channel attacks and static inspection attacks on silicon chips have necessitated techniques to make circuit implementations resistant (private) to these probes and inspections. t-private circuits protect the privacy of the data in flight during computation. Memories (on-chip or off-chip) however are not protected by t-private circuits.

Valamehr *et al.* [Valamehr et al. (2012)] introduced a few memory protection schemes. We introduce a unified analysis framework to compare these schemes. Effectiveness metrics for these schemes include area/gate count overhead, time overhead, number of random bits needed, and adversary success probability per random bit. In this chapter, we specifically analyzed the storage overhead and the success probability of t-private memories, t-private memories with random matrix (for random bits reuse), and a hybrid private memory.

Ideally, we would like to design a private computing circuit with unified private memory. In such a computing system, data and keys never appear in their raw form, thereby protecting privacy of data and keys. We consider a memory scheme to be computable if the encoded stored keys can be directly used in t-private computations.

Most of the memory schemes presented in Valamehr et al. [Valamehr et al. (2012)] are not computable. The main new interesting technique they develop is to judiciously reuse random bits while still limiting the adversary to low success probability. We develop a new memory schema that is computable, and yet reuses many random bits by bringing in an arithmetic function into encoding. We present the computable and t-private encoding method and cor-

responding logic operations (AND, OR and NOT) suitable for our memory scheme. The new private circuits are more efficient than Ishai's t-private model (only t area overhead compared to t^2 area overhead of Ishai). We verified that our memory model has advantages in performance (the success probability and delay) and area cost by implementing it on FPGA.

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis, the methodology to implement secure hardware design against side-channel attacks has been proposed. Unsafe modules in the cryptographic system are searched by SCA security metrics based on normalized standard deviation, KL divergence or mutual information. If security metrics of any modules are out of the boundary range or threshold, the modules are vulnerable against side-channel attacks. In order to find the boundary or threshold, security metrics are compared with the result value of simulated side-channel attacks such as the successful probability or the successful recognition rate. The range between 0 and allowable successful recognition rate is mapped on the range of security metrics. In order to make more strict boundary, various side-channel attacks using LDA, QDA, naïve Bayes classifier and SVM are performed.

Vulnerable modules are transformed into secure modules by re-synthesizing with secure logic styles such as SABL, WDDL or t-private logic cells. These secure logic styles are satisfied with the secure condition based on the security metrics. Designers can select secure logic style suitable for the hardware specification and constraints.

Memories also should be protected from physical access such as probing to reveal secret information stored in the memory. For the protection, we develop a new computable t-private memory schema which reuses many random bits by bringing in an arithmetic function into encoding. The computable and t-private encoding method can be applied to combinational logic operation. The new private circuits are more efficient than Ishai's t-private model (only t area overhead compared to t^2 area overhead of Ishai). Consequently, the secure logic package including secure logic styles and private memories should be required to implement secure ASIC

or FPGA hardware system against SCA attacks.

7.2 Future Work

There exist several challenging problems in future work in the area of secure hardware implementation. Our graph-based power estimation method using the renewal theory and linear regression may be too time-consuming to estimate power of large-size digital module even though this method is faster than SPICE simulation. For fast and reliable security testing, high performance computing using GPU or hardware accelerators can be an alternative to solve the problem. The graph-based algorithm can be mapped on GPU.

We do not deal with how to generate random bits in this thesis. t-private logic circuits must require a lot of random bits for the perfect security. PUF-based random number generators will be good choice. Also, the distribution of random bits to t-private logic circuits will be significant issue. Ideally, refreshed random bits must be provided to every private circuits in each clock cycle but it causes large power consumption and large area increasing. Efficient distribution of random numbers temporally and spatially should be researched.

APPENDIX A. THE ADVANCED ENCRYPTION STANDARD [FIPS (2001)]

A.1 Algorithm

Ciper(byte in[4*N_b], byte in[4*N_b], word w[N_b*(N_r+1)])
begin
  byte state[4, N_b]
  state = in
  AddRoundkey(state, w[0, N_b-1])
for *round* = 1 to N_r-1 **do**
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundkey(state, w[round*N_b, (round+1)*N_b − 1])
end for
  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w[$N_r * N_b, (N_r + 1) * N_b - 1$])
  out = state
end
// N_r: the number of rounds, N_b : the number of columns (32-bit words) comprising the State

Algorithm 3 Pseudo Code for AES encryption

A.1.1 `SubBytes`

The `SubBytes` step is the only non-linear transformation of the cipher. `SubBytes` is a bricklayer permutation consisting of an S-box applied to the bytes of the state. Fig. A.1 illustrates the effect of the `SubBytes` step on the state. The S-box function should be satisfied with the following conditions:

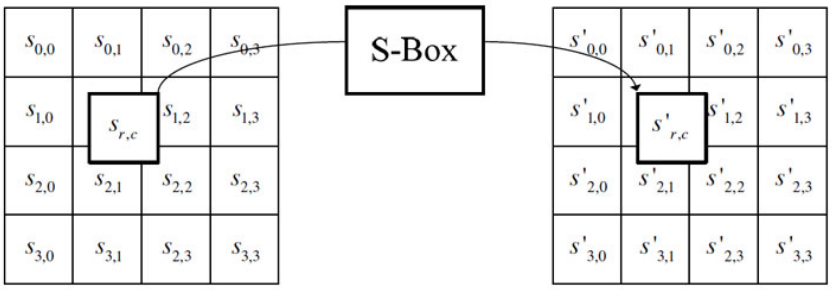


Figure A.1: `SubByte ( )` applies the S-box to each byte of the State

1. The maximum input-out correlation amplitude must be as small as possible.
2. The maximum difference propagation probability must be as small as possible.
3. The algebraic expression of S-box in GF(2^8) has to be complex.

The S-box is defined as the following equations:

$$Sbox(a) = f(g(a))$$

$$g : a \rightarrow b = a^{-1} \ \ (\ mod\ x^8 + x^4 + x^2 + x + 1\ )\ \ in\ \ \mathrm{GF}(2^8)$$

$$b = f(a) : \text{affine transformation}$$

$$\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

A.1.2 `ShiftRows`

The `ShiftRows` step is a byte transposition that cyclically shifts the rows of the state over different offsets. Row 0 is shifted over C_0 bytes, row 1 over C_1 bytes, row 2 over C_2 bytes and row 3 over C_3 bytes, so that the byte at position j in row i moves to position $(j - C_i)$ mod N_b.

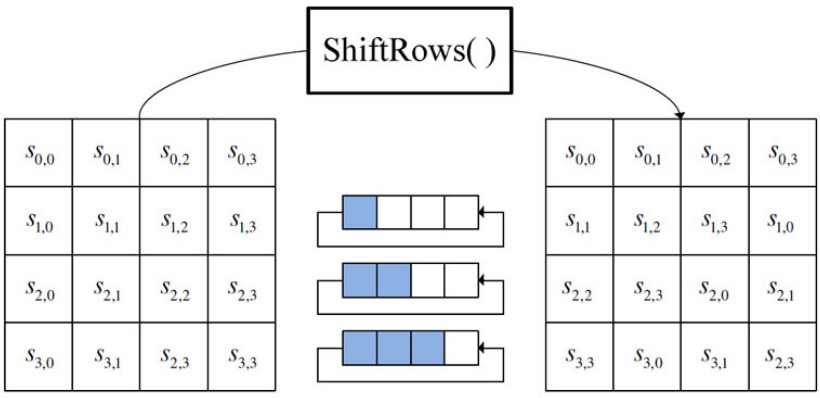


Figure A.2: `ShiftRows ( )` cyclically shifts the last three rows in the State

The shift offsets C_0, C_1, C_2 and C_3 depends on the value of N_b. Table A.1 shows shift offsets depending on N_b. Fig. A.2 illustrates the `ShiftRows` transformation.

Table A.1: `ShiftRows`: shift offsets for different block lengths

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

A.1.3 `MixColumns`

The `MixColumns` step is a bricklayer permutation operating on the state column by column. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$, given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Let $s'(x) = a(x) \cdot s(x) \pmod{x^4 + 1}$. Then

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

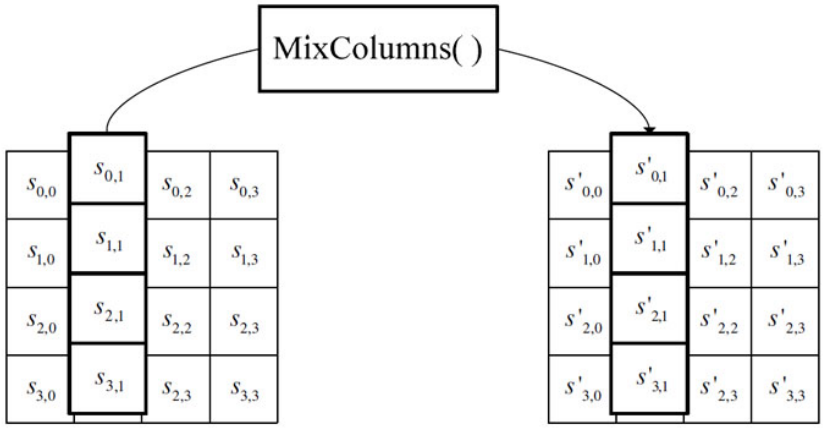


Figure A.3: `MixColumns( )` operates on the State column-by-column

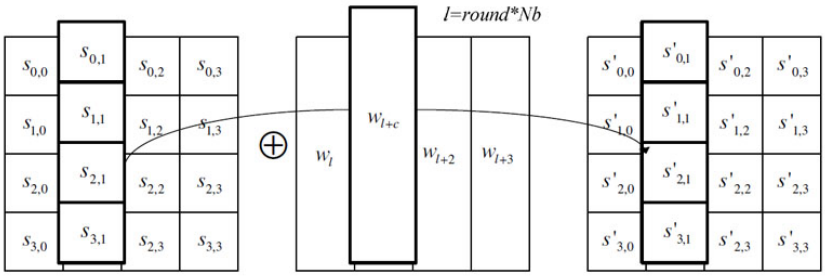


Figure A.4: `AddRoundKey( )` XORs each column of the State with a word from the key schedule

Fig. A.3 illustrates the `MixColumns` transformation.

A.1.4 `AddRoundKey`

The key addition is denoted `AddRoundKey`. In this transformation, the state is modified by combining it with a round key with the bitwise XOR operation. Each round key consists of N_b words from the key schedule. Those N_b words are each added into the columns of the State, such that

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [w_{round*N_b+c}] \quad for \;\; 0 \leq c < N_b.$$

Fig. A.4 illustrates the `AddRoundKey` operation.

A.1.5 Key Schedule

The key schedule consists of two components: the key expansion and the round key selection. Alg. 4 represents pseudo code for key expansion. `SubWord()` is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function `RotWord()` takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, `Rcon[i]`, contains the value given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of x (x is denoted as $\{02\}$) in the field GF(2^8)).

Key Expansion(byte key[4*N_k], word w[N_b*(N_r+1)], N_k)
begin
  word temp
  i = 0
while $i < N_k$ **do**
  w[i] = word (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]
  i = i + 1
end while
  i = N_k
while $i < N_b * (N_r + 1)$ **do**
    tamp = w[i-1]
  if $i \mod N_k = 0$ **then**
    temp = SubWord(RotWord(temp)) xor Rcon[i/N_k]
  else if $N_k > 6$ and $i \mod N_k = 4$ **then**
    temp = SubWord(temp)
  end if
  w[i] = w[i-N_k] xor temp
  i = i + 1
end while
end
// Note that N_k = 4, 6 or 8 when key lengths are 128, 192 or 256 bits, repectively

Algorithm 4 Pseudo Code for Key Expansion

APPENDIX B. TOOL SCRIPTS

B.1 Setup (FreePDK45)

1. Download FreePDK45 design kit

at http://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

2. Make setup script.

```
#!/bin/bash
##########################################################
#  FreePDK Setup Script
#  3/21/2016 by Jungmin Park (jmpark00@iastate.edu)
##########################################################

# Set the CDK_DIR variables
export CDK_DIR=/usr/local/cadence/iclocal/ncsu-cdk-1.6.0.beta
# Set the PDK_DIR variables to the root directory of the FreePDK
    distribution
export PDK_DIR=$PWD/FreePDK45
# Set CDSHOME to the root directory of the Cadence ICOA installstion
export CDSHOME=$IC

if [ ! -f "$PWD/.cdsenv" ]
   then
        cp /remote/ncsu_oa/local/cdssetup/cdsenv $PWD/.cdsenv
fi
```

```
if [ ! -f "$PWD/.cdsinit" ]
   then
  cp $PDK_DIR/ncsu_basekit/cdssetup/cdsinit $PWD/.cdsinit
fi

if [ ! -f "$PWD/cds.lib" ]
   then
  cp $PDK_DIR/ncsu_basekit/cdssetup/cds.lib $PWD/cds.lib
fi

if [ ! -f "$PWD/lib.defs" ]
   then
       cp $PDK_DIR/ncsu_basekit/cdssetup/lib.defs $PWD/lib.defs
fi

if [ ! -f "$PWD/.runset.calibre.drc" ]
   then
       cp $PDK_DIR/ncsu_basekit/cdssetup/runset.calibre.drc $PWD/.runset.
    calibre.drc
fi

if [ ! -f "$PWD/.runset.calibre.lvs" ]
   then
       cp $PDK_DIR/ncsu_basekit/cdssetup/runset.calibre.lvs $PWD/.runset.
    calibre.lvs
fi

if [ ! -f "$PWD/.runset.calibre.lfd" ]
   then
       cp $PDK_DIR/ncsu_basekit/cdssetup/runset.calibre.lfd $PWD/.runset.
    calibre.lfd
```

```
fi

if [ ! -f "$PWD/.runset.calibre.pex" ]
   then
       cp $PDK_DIR/ncsu_basekit/cdssetup/runset.calibre.pex $PWD/.runset.
    calibre.pex
fi

export present=$PYTHONPATH
if [ $present = "" ]
   then
  export PYTHONPATH=$PDK_DIR/ncsu_basekit/techfile/cni
   else
    export PYTHONPATH=$PYTHONPATH:$PDK_DIR/ncsu_basekit/techfile/cni
fi

export MGC_CALIBRE_DRC_RUNSET_FILE=./.runset.calibre.drc
export MGC_CALIBRE_LVS_RUNSET_FILE=./.runset.calibre.lvs
export MGC_CALIBRE_PEX_RUNSET_FILE=./.runset.calibre.pex
```

3. Modify cds.lib file

```
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
DEFINE US_8ths   $CDSHOME/tools/dfII/etc/cdslib/sheets/US_8ths
DEFINE basic     $CDSHOME/tools/dfII/etc/cdslib/basic
DEFINE cdsDefTechLib $CDSHOME/tools/dfII/etc/cdsDefTechLib
DEFINE NCSU_TechLib_FreePDK45  $PDK_DIR/ncsu_basekit/lib/
    NCSU_TechLib_FreePDK45
DEFINE NCSU_Devices_FreePDK45  $PDK_DIR/ncsu_basekit/lib/
    NCSU_Devices_FreePDK45
DEFINE NCSU_Analog_Part   $CDK_DIR/lib/NCSU_Analog_Parts
```

```
DEFINE OSU $PDK_DIR/osu_soc/lib/freepdk45_cells
```

4. Modify .cdsenv file

```
;------------------------------------------------------------------------
; spectre environment variables
;------------------------------------------------------------------------
spectre.envOpts     modelFiles        string   "$PDK_DIR/osu_soc/lib/files/
    gpdk45nm.m"

spectre.envOpts     controlMode       string   "batch"
```

5. Execute Cadence virtuoso.

```
$ source setup.sh
$ virtuoso &
```

B.2 RTL Complier Tcl Script

```
################################################
# Script for Cadence RTL Compiler synthesis
# Use with syn-rtl -f <rtl-script>
################################################

# Set the search paths to the libraries and the HDL files
# Remember that "." means your current directory
set_attribute hdl_search_path {../functional};
set_attribute lib_search_path {../libdir};
set_attribute library [list gscl45nm.lib];
```

```
set_attribute information_level 6;  # See a lot of warnings.

set myFiles [list verilog.v];
set basename AND2X1t1;        # top module
set runname RTL;
#set myPeriod_ps 10000
#set myInDelay_ps 250
#set myOutDelay_ps 250


################################################
# below here shouldn't neet to be changed
################################################

# Analyze and Elaborate the HDL files
read_hdl ${myFiles}
elaborate ${basename}

# Apply Constraints and generate clocks
# set clock [define_clock -period ${myPeriod_ps} -name ${myClk} [
    clock_ports]]
# external_delay -input $myInDelay_ps -clock ${myClk} [find / -port
    ports_in/*]
# external_delay -output $myOutDelay_ps -clock ${myClk} [find / -port
    ports_out/*]

# Sets transition to default values for Synopsys SDC format,
# fall/rise 400 ps
# dc::set_clock_transition .4 $myClk

# check that the design in OK so far
check_design -unresolved
```

```
report timing -lint

# Synthesize the design to the target library
synthesize -to_mapped

# Write out the reports
report timing > ${basename}_${runname}_timing.rep
report gates > ${basename}_${runname}_cell.rep
report power > ${basename}_${runname}_power.rep
report area > ${basename}_${runname}_area.rep

# Write out the structural Verilog and sdc files
write_hdl -mapped > ../encounter/${basename}_${runname}.v
write_sdc > ../encounter/${basename}_${runname}.sdc
```

B.3 Encounter Script

B.3.1 Configuration file (`encounter.conf`)

```
#################################################
#                                               #
#  FirstEncounter Input configuration file      #
#                                               #
#################################################

# Specify the name of your toplevel module
set my_toplevel  AND2X1t1
set RTL  _RTL
#################################################
# No changes required below
#################################################
```

```
global env
#set OSU_FREEPDK $env(PDK_DIR)/osu_soc

global rda_Input
set rda_Input(ui_netlist) $my_toplevel$RTL.v
set rda_Input(ui_timingcon_file) $my_toplevel$RTL.sdc
set rda_Input(ui_topcell) $my_toplevel

set rda_Input(ui_netlisttype) {Verilog}
set rda_Input(ui_ilmlist) {}
set rda_Input(ui_settop) {1}
set rda_Input(ui_celllib) {}
set rda_Input(ui_iolib) {}
set rda_Input(ui_areaiolib) {}
set rda_Input(ui_blklib) {}
set rda_Input(ui_kboxlib) ""
set rda_Input(ui_timelib) "../libdir/gscl45nm.tlf"
set rda_Input(ui_smodDef) {}
set rda_Input(ui_smodData) {}
set rda_Input(ui_dpath) {}
set rda_Input(ui_tech_file) {}
set rda_Input(ui_io_file) ""
set rda_Input(ui_buf_footprint) {buf}
set rda_Input(ui_delay_footprint) {buf}
set rda_Input(ui_inv_footprint) {inv}
set rda_Input(ui_leffile) "../libdir/gscl45nm.lef"
set rda_Input(ui_core_cntl) {aspect}
set rda_Input(ui_aspect_ratio) {1.0}
set rda_Input(ui_core_util) {0.7}
set rda_Input(ui_core_height) {}
set rda_Input(ui_core_width) {}
set rda_Input(ui_core_to_left) {}
```

```
set rda_Input(ui_core_to_right) {}
set rda_Input(ui_core_to_top) {}
set rda_Input(ui_core_to_bottom) {}
set rda_Input(ui_max_io_height) {0}
set rda_Input(ui_row_height) {}
set rda_Input(ui_isHorTrackHalfPitch) {0}
set rda_Input(ui_isVerTrackHalfPitch) {1}
set rda_Input(ui_ioOri) {R0}
set rda_Input(ui_isOrigCenter) {0}
set rda_Input(ui_exc_net) {}
set rda_Input(ui_delay_limit) {1000}
set rda_Input(ui_net_delay) {1000.0ps}
set rda_Input(ui_net_load) {0.5pf}
set rda_Input(ui_in_tran_delay) {120.0ps}
set rda_Input(ui_captbl_file) {}
set rda_Input(ui_cap_scale) {1.0}
set rda_Input(ui_xcap_scale) {1.0}
set rda_Input(ui_res_scale) {1.0}
set rda_Input(ui_shr_scale) {1.0}
set rda_Input(ui_time_unit) {none}
set rda_Input(ui_cap_unit) {}
set rda_Input(ui_sigstormlib) {}
set rda_Input(ui_cdb_file) {}
set rda_Input(ui_echo_file) {}
set rda_Input(ui_qxtech_file) {}
set rda_Input(ui_qxlib_file) {}
set rda_Input(ui_qxconf_file) {}
set rda_Input(ui_pwrnet) {vdd}
set rda_Input(ui_gndnet) {gnd}
set rda_Input(flip_first) {1}
set rda_Input(double_back) {1}
set rda_Input(assign_buffer) {0}
```

```
set rda_Input(ui_pg_connections) [list \
                        {PIN:vdd:} \
                        {PIN:gnd:} \
                              ]
set rda_Input(PIN:vdd:) {vdd}
set rda_Input(PIN:gnd:) {gnd}
```

B.3.2 tcl file (encounter.tcl)

```
#####################################
# Run the design through Encounter
#####################################

# Setup design and create floorplan
loadConfig ./encounter.conf
#commitConfig

# Create Initial Floorplan
floorplan -r 1.0 0.85 0 0 0 0

# Create Power structures
#addRing -spacing_bottom 5 -width_left 5 -width_bottom 5 -width_top 5 -
    spacing_top 5 -layer_bottom metal5 -width_right 5 -around core -center
    1 -layer_top metal5 -spacing_right 5 -spacing_left 5 -layer_right
    metal6 -layer_left metal6 -nets { gnd vdd }

# Place standard cells
amoebaPlace

# Route power nets
sroute -noBlockPins -noPadRings
```

```
# Perform trial route and get initial timing results
trialroute
#buildTimingGraph
#setCteReport
#reportTA -nworst  10 -net > timing.rep.1.placed

# Run in-place optimization
# to fix setup problems
#setIPOMode -mediumEffort -fixDRC -addPortAsNeeded
#initECO ./ipo1.txt
#fixSetupViolation
#endECO
#buildTimingGraph
#setCteReport
#reportTA -nworst  10 -net > timing.rep.2.ipo1

# Run Clock Tree Synthesis
#createClockTreeSpec -output encounter.cts -bufFootprint buf -invFootprint
    inv
#specifyClockTree -clkfile encounter.cts
#ckSynthesis -rguide cts.rguide -report report.ctsrpt -macromodel report.
    ctsmdl -fix_added_buffers

# Output Results of CTS
#trialRoute -highEffort -guide cts.rguide
#extractRC
#reportClockTree -postRoute -localSkew -report skew.post_troute_local.
    ctsrpt
#reportClockTree -postRoute -report report.post_troute.ctsrpt

# Run Post-CTS Timing analysis
#setAnalysisMode -setup -async -skew -autoDetectClockTree
```

```
#buildTimingGraph
#setCteReport
#reportTA -nworst  10 -net > timing.rep.3.cts

# Perform post-CTS IPO
#setIPOMode -highEffort -fixDrc -addPortAsNeeded -incrTrialRoute  -restruct
    -topomap
#initECO ipo2.txt
#setExtractRCMode -default -assumeMetFill
#extractRC
#fixSetupViolation -guide cts.rguide

# Fix all remaining violations
#setExtractRCMode -detail -assumeMetFill
#extractRC
#if {[isDRVClean -maxTran -maxCap -maxFanout] != 1} {
#fixDRCViolation -maxTran -maxCap -maxFanout
#}

#endECO
#cleanupECO

# Run Post IPO-2 timing analysis
#buildTimingGraph
#setCteReport
#reportTA -nworst  10 -net > timing.rep.4.ipo2

# Add filler cells
addFiller -cell FILL -prefix FILL -fillBoundary

# Connect all new cells to VDD/GND
globalNetConnect vdd -type tiehi
```

```
globalNetConnect vdd -type pgpin -pin vdd -override

globalNetConnect gnd -type tielo
globalNetConnect gnd -type pgpin -pin gnd -override

# Run global Routing
globalDetailRoute

# Get final timing results
#setExtractRCMode -detail -noReduce
#extractRC
#buildTimingGraph
#setCteReport
#reportTA -nworst  10 -net > timing.rep.5.final

# Output GDSII
streamOut  final.gds2 -mapFile ../libdir/gds2_encounter.map -stripes 1 -
    units 1000 -mode ALL
saveNetlist -excludeLeafCell final.v

# Output DSPF RC Data
rcout -spf final.dspf

# Run DRC and Connection checks
verifyGeometry
verifyConnectivity -type all

win

puts "************************************"
puts "* Encounter script finished        *"
puts "*                                  *"
```

```
puts "* Results:                                *"
puts "* --------                               *"
puts "* Layout:   final.gds2                   *"
puts "* Netlist:  final.v                      *"
puts "* Timing:   timing.rep.5.final           *"
puts "*                                        *"
puts "* Type 'exit' to quit                    *"
puts "*                                        *"
puts "******************************************"
```

BIBLIOGRAPHY

Agrawal, D. and Aggarwal, C. C. (2001). On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*.

Alioto, M., Poli, M., and Rocchi, S. (2010). A general power model of differential power analysis attacks to static logic circuits. *IEEE Trans. Very Large Scale Integr. Syst.*, 18(5):711–724.

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Basel Halak, Julian Murphy, A. Y. (2013). Power balanced circuits for leakage-power-attacks resilient design. Cryptology ePrint Archive, Report 2013/048. http://eprint.iacr.org/.

Fei, Y., Ding, A. A., Lao, J., and Zhang, L. (2014). A statistics-based fundamental model for side-channel attack analysis. Cryptology ePrint Archive, Report 2014/152. http://eprint.iacr.org/.

FIPS (2001). Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES).

Group, T. C. (2013). Trusted Platform Module Specification and Architecture. Online at http://www.trustedcomputinggroup.org/resources/tpm_main_specification/.

Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Cal, J. A., Feldman, A. J., and Felten, E. W. (2008). Least we remember: Cold boot attacks on encryption keys. In *In USENIX Security Symposium*.

Ishai, Y., Sahai, A., and Wagner, D. (2003). Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology*

Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer.

Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397. Springer-Verlag.

Leuven, K. (2011). Ls-svmlab v1.8. Online at http://www.esat.kuleuven.be/sista/lssvmlab/.

Mac, F., Standaert, F.-X., and Quisquater, J.-J. (2007). Information theoretic evaluation of side-channel resistant logic styles. In Paillier, P. and Verbauwhede, I., editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 427–442. Springer.

Mangard, S. (2005). Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints. In *Systems ? CHES 2005, 7th International Workshop*, pages 172–186. Springer.

Mangard, S., Oswald, E., and Popp, T. (2007). *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mangard, S., Pramstaller, N., and Oswald, E. (2005). Successfully attacking masked aes hardware implementations. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer.

Mathai, A. and Provost, S. (1992). *Quadratic Forms in Random Variables*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis.

Messerges, T. S. (2000). Securing the aes finalists against power analysis attacks. In *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, Lecture Notes in Computer Science, pages 150–164. Springer.

Messerges, T. S., Dabbish, E. A., Sloan, R. H., and Member, S. (2002). Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51:541–552.

Micali, S. and Reyzin, L. (2003). Physically observable cryptography. In *TCC 2004, LNCS*, pages 278–296. Springer.

Micheli, G. D. (1994). *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition.

Mohyuddin, N., Pakbaznia, E., and Pedram, M. (2008). Probabilistic error propagation in logic circuits using the boolean difference calculus. In *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pages 7 –13.

Monteiro, J. C., Devadas, S., Ghosh, A., Keutzer, K., and White, J. K. (1997). Estimation of average switching activity in combinational logic circuits using symbolic simulation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 16(1):121–127.

Najm, F. N. (1994). A survey of power estimation techniques in vlsi circuits. *IEEE Trans. Very Large Scale Integr. Syst.*, 2(4):446–455.

NCSU (2011). Version 1.4 of freepdk45 kit. Online at http://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

Nelson, R. D. (1995). *Probability, stochastic processes, and queueing theory - the mathematics of computer performance modeling*. Springer.

OSU (2008). Osu freepdk45 kit. Online at http://vlsiarch.ecen.okstate.edu/flow/#.

Park, J. and Tyagi, A. (2012). t-private logic synthesis on fpgas. In *HOST*, pages 63–68. IEEE.

Park, J. and Tyagi, A. (2014a). t-private systems: Unified private memories and computation. In *Security, Privacy, and Applied Cryptography Engineering - 4th International Conference, SPACE 2014, Pune, India, October 18-22, 2014. Proceedings*, pages 285–302.

Park, J. and Tyagi, A. (2014b). Towards making private circuits practical: DPA resistant private circuits. In *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2014, Tampa, FL, USA, July 9-11, 2014*, pages 528–533.

Park, J. and Tyagi, A. (2016). Security metrics for power based SCA resistant hardware implementation. In *29th International Conference on VLSI Design and 15th International Conference on Embedded Systems, VLSID 2016, Kolkata, India, January 4-8, 2016*, pages 541–546.

Prouff, E. and Rivain, M. (2007). A generic method for secure Sbox implementation. *Information Security Applications*, pages 227–244.

Quisquater, J.-J. and Samyde, D. (2001). Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, E-SMART '01, pages 200–210, London, UK, UK. Springer-Verlag.

Reed, I. (1954). A class of multiple-error-correcting codes and the decoding scheme. *Information Theory, IRE Professional Group on*, 4(4):38 –49.

S. Kullback and R. A. Leibler (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Samyde, D., Skorobogatov, S., Anderson, R., and Quisquater, J.-J. (2002). On a new way to read data from memory. In *Proceedings of the First International IEEE Security in Storage Workshop*, SISW '02, pages 65–, Washington, DC, USA. IEEE Computer Society.

Sasao, T. and Fujita, M., editors (1996). *Representations of Discrete Functions*. Kluwer Academic Publishers, Norwell, MA, USA.

Satoh, A., Morioka, S., Takano, K., and Munetoh, S. (2001). A compact rijndael hardware architecture with s-box optimization. In Boyd, C., editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer.

Sentovich, E., Singh, K., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P., Brayton, R. K., and Sangiovanni-Vincentelli, A. L. (1992). Sis: A system for sequential circuit synthesis. Technical report, EECS Department, University of California, Berkeley.

Standaert, F.-X., Malkin, T. G., and Yung, M. (2009). A unified framework for the analysis of side-channel key recovery attacks. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 443–461, Berlin, Heidelberg. Springer-Verlag.

Tiri, K., Akmal, M., and Verbauwhede, I. (2002). A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pages 403–406.

Tiri, K. and Verbauwhede, I. (2004). A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '04, pages 10246–, Washington, DC, USA. IEEE Computer Society.

Tiri, K. and Verbauwhede, I. (2005). A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 3*, DATE '05, pages 58–63, Washington, DC, USA. IEEE Computer Society.

Tyagi, A. (2005). Energy-privacy trade-offs in vlsi computations. In *Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings*, volume 3797 of *Lecture Notes in Computer Science*, pages 361–374. Springer. A version titled Energy-Privacy-Time Tradeoffs in VLSI Computations under revision for IEEE Trans. on Computers.

Valamehr, J., Chase, M., Kamara, S., Putnam, A., Shumow, D., Vaikuntanathan, V., and Sherwood, T. (2012). Inspection resistant memory: architectural support for security from physical examination. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 130–141, Washington, DC, USA. IEEE Computer Society.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, NY, USA.

Vapnik, V. N. (1998). *Statistical Learning Theory.* Wiley-Interscience.

Wasserman, L. (2006). *All of Nonparametric Statistics (Springer Texts in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Weste, N. and Harris, D. (2010). *CMOS VLSI Design: A Circuits and Systems Perspective.* Addison-Wesley Publishing Company, USA, 4th edition.