Low Entropy Side-Channel Secure Hardware Implementations

Jhelum Dhar

Low Entropy Side-Channel Secure Hardware Implementations

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Cryptology and Security

by

Jhelum Dhar
[Roll No: CrS2309]

under the guidance of

Prof. Dr. Ir. Vincent Rijmen
Prof. Bimal Kumar Roy
Daily supervisor: Artemii Ovchinnikov

July 2025

Acknowledgments

I am deeply grateful to everyone who has offered their help and encouragement throughout this semester. To begin with, I would like to express my gratitude to Prof. Bimal Kumar Roy and Prof. Dr. Ir. Bart Preneel for giving me the opportunity to work at COSIC, KU Leuven. My deepest appreciation goes to Prof. Dr. Ir. Vincent Rijmen for his valuable insights and supervision. I am especially thankful to Artemii Ovchinnikov. His constant guidance and advice throughout this project have been invaluable. Finally, I thank my family for being my steady source of emotional support.

Jhelum Dhar

Abstract

The demand for symmetric-key cryptography implemented in hardware is growing due to the increasing need for faster, more efficient, and secure encryption in small devices. However, implementing block ciphers in hardware that are side-channel secure remains a challenging goal. This holds true because there exist sophisticated but well-studied attacks such as Differential Power Analysis, which uses the correlation between power consumption of a device and the information on it to allow attackers with physical access to the cryptographic device to get information about secret data.

Masking is one of the techniques that is used to provide security against side-channel attacks. There are various kinds of masking, including widely recognized Threshold Implementations and Domain-Oriented Masking. However, to mask a secret, one must first generate randomness. Generating secure randomness usually comes at the cost of increased area and time in hardware.

In this master's thesis project, we study ways of reducing or reusing the randomness used in masked hardware implementations of symmetric-key block ciphers and calculate the bounds on the advantage of a threshold probing adversary to determine if the countermeasures preserve security. We then use PROLEAD to verify the probing security and compare its result with our estimations.

Keywords: *Symmetric-Key Cryptography, Linear Cryptanalysis, Side-Channel Security, Threshold Implementations, Probing Security*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Implementing block ciphers in hardware that are secure against side-channel attacks, such as differential power analysis (DPA) [19], is both a crucial and challenging objective. In modern times, tens of billions of embedded devices are used globally and about five to ten per person. These devices are typically implemented using pure hardware or software chips such as ARM Cortex-M or RISC-V cores. In this work, we focus on purely hardware-based implementations. Hardware-based encryption protects sensitive data, as seen in SSDs, car key fobs, smart cards and some IoT devices.

There are many countermeasures to break the correlation between secret data and physical leakage that side-channel attacks exploit. For example, introducing redundant operations and aligning power consumption. Our study is concentrated on the algorithmic countermeasure, Threshold Implementation, introduced by Nikova *et al.* [25] in 2006, based on secret sharing [9], threshold cryptography [13] and Multi-Party Computation protocols [27]. A specific challenge in hardware are glitches, observable pulses that appear on the output when the input changes due to propagation delays. Glitches may break the security of ciphers [20, 21]. But the threshold implementation resists side-channel attacks, even when glitches are present. Here, the secret is split into multiple shares and each operation on the shares is done such a way that it does not reveal any information about the secret. As a result, only a fixed number of randomness is required initially and we do not need new randomness every clock cycle to re-mask the uncovered secrets, like the work [16] does.

The cost and security of generating randomness for uniform masking are often overlooked. Although researchers suggest using strong and recently relatively efficient [12] cryptographic generators, practical implementations usually rely on weaker, less

secure ones. To address this, designers focus on minimizing the number of random bits needed for secure masking, rather than choosing the perfect generator. Especially if we refer to fresh randomness, since multiple modern masking schemes actively exploit refreshing, including auto-generated masked designs [18, 26]. In this thesis, our aim is to reduce initial randomness with a design proposal based on the security model previously introduced [5, 4].

1.2 Goal

In 2023, Dhooghe *et al.* conducted an in-depth analysis of the security of first order threshold implementations for the lightweight block ciphers Prince [11] and Midori64 [1] under non-uniform input conditions [14]. They presented examples of secure and insecure non-uniform input for both ciphers. For Prince insecure input, immediate leakage was seen. For Midori64, they observed a concentrated leakage for the insecure input and the amount of experiments needed to observe the leakage was close to the estimated bound. The secure inputs for both ciphers did not show any leakage during the experiments with FPGA.

Building on their methodology and findings, this work focuses on extending similar security evaluations to two additional lightweight ciphers: Present [10] and Skinny [3]. For this analysis, we implement both ciphers using three shares to design first order secure threshold implementations. To reduce the number of shares required for masking S-box decompositions are used. Then, using linear cryptanalytic techniques, we try to find ways to reuse a limited initial randomness to mask the plaintext in both ciphers. We use Mixed Integer Linear Programming [23] to find the linear trails for Skinny. To evaluate the security of our proposed non-uniform inputs, we compute bounds on the advantage of a threshold probing adversary. These theoretical estimations are then validated using the PROLEAD [24] software.

The rest of the thesis structured as follows. Chapter 2 covers the theoretic background for the work. Chapter 3 and 4 contain the proposed non-uniform inputs and their security analysis for Skinny and Present respectively. The experimental results for those proposed inputs are given in Chapter 5. Finally, Chapter 6 concludes the thesis and indicates a direction for future work.

Chapter 2

Preliminaries

In this chapter, we introduce some basic concepts of linear cryptanalysis, boolean masking and threshold implementation. We also describe the security model used in this work and explain how a trail-based method is applied to evaluate the security of a cipher masked with non-uniform inputs. This chapter is based on the previous works published in the field of side-channel security.

2.1 Linear Cryptanalysis and Fourier Analysis

We first recall some basic notation from linear algebra, which will serve as a basis for the theory that follows.

RECALL:

- $V^{\perp} = \{x \in \mathbb{F}_2^n \mid \forall v \in V,\ v^{\intercal}x = 0\}$ is the orthogonal complement of a subspace V of $\mathbb{F}_2^n$.
- The quotient space $\mathbb{F}_2^n/V^{\perp}$ contains all the cosets of $V^{\perp}$.

To simplify things, an element $x + V^{\perp} \in \mathbb{F}_2^n/V^{\perp}$ will be denoted by x. For $x \in \mathbb{F}_2^n/V^{\perp}$ and $v \in V$, the expression $x^{\intercal}v$ is well defined.

Linear cryptanalysis is a technique to find linear relations between the input and the output of a function. So it is trivially applicable to linear and affine functions, but to apply the analysis to non-linear functions (like S-boxes), we need to use linear approximations. To combine the linear approximations for multiple non-linear

functions located between the input and the output of a cipher, the piling-up lemma [22] is used. Linear cryptanalysis is typically used to determine whether a cipher is secure. The known plaintext attack using linear cryptanalysis [22] reveals some key bits, making key recovery easier for an adversary. Here, however, it is used for a different purpose. It allows to put a quantitative bound on an adversary attempting to apply SCA to a masked cipher. The connection between linear cryptanalysis and Fourier analysis, as studied by Beyne *et al.* [5], plays an important role in the security evaluation presented in this work. Some necessary definitions related to linear cryptanalysis and Fourier analysis are given below.

Definition 1 (Fourier transformation of a function) *Let $V \subseteq \mathbb{F}_2^n$ be a vector space. The Fourier transformation of the complex-valued function $f : V \to \mathbb{C}$ is a function $\hat{f} : \mathbb{F}_2^n/V^\perp \to \mathbb{C}$ defined by*

$$\hat{f}(u) = \sum_{x \in V} (-1)^{u^\intercal x} f(x)$$

where we write u for $u + V^\perp$

Definition 2 (Correlation matrix) *Let $V \subseteq \mathbb{F}_2^n$ e a vector space. The correlation matrix of the function $f : V \to V$ is a real $|V| \times |V|$ matrix with coordinates equal to*

$$C_{v,u}^F = \frac{1}{|V|} \sum_{x \in V} (-1)^{u^\intercal x + v^\intercal F(x)}$$

where, $u, v \in \mathbb{F}_2^n/V^\perp$.

Definition 3 (Bias of a linear approximation) *Let $F : V \to V$, with $V \subseteq \mathbb{F}_2^n$, be a function and $\mathbf{x}$ be a random variable in V, then the bias $\epsilon_{u,v}$ of a linear approximation (u, v) is defined as,*

$$\epsilon_{u,v} = Pr[v^\intercal F(\mathbf{x}) = u^\intercal \mathbf{x}] - \frac{1}{2}$$

The coordinate $C_{v,u}^F$ of the correlation matrix of F is equal to the correlation of a linear approximation on F with input mask u and output mask v. That is, $C_{v,u}^F = 2\Pr[v^\intercal F(\mathbf{x}) = u^\intercal \mathbf{x}] - 1$ for $\mathbf{x}$ uniform random on V.

A linear approximation table denotes the relation between all possible input and output masks. In particular, for a pair of input and output masks, it denotes how closely the function approximates the linear behavior. It is formally defined below.

Definition 4 (Linear approximation table (LAT)) *The linear approximation table (LAT) of a function $F : V \to V$ $[V \subseteq \mathbb{F}_2^n]$ is defined by the coordinates,*

$$LAT_{v,u}^F = 2^n \epsilon_{u,v}$$

2.2 Boolean Masking and Threshold Implementation

As it was mentioned before, one of the frequently used techniques to protect against side-channel attacks is masking. This thesis focuses on block ciphers based on substitution-permutation network constructions, where permutations are linear or affine functions and substitution blocks (S-boxes) are not. To efficiently mask the affine layers, we opt for Boolean masking. This is a technique based on the Boolean XOR operation defined below.

Definition 5 (Boolean Masking of a secret) *Each secret $x \in \mathbb{F}_2$ in the circuit is split into shares $\bar{x} = (x_0, x_1, ..., x_{s_x-1})$ such that $x = \sum_{i=0}^{s_x-1} x_i$ over $\mathbb{F}_2$. This is called Boolean Masking.*

If for a random Boolean masking of a fixed secret, all masking are equally likely, then the random Boolean masking is called uniform. To operate on masked input, one needs a masked algorithm (or a circuit, if we consider hardware implementations) to process the shares in parallel. Masking linear layers is straightforward as they may be replicated times the number of shares and process each part of a secret individually.

Masking a non-linear layer of an algorithm, on the other hand, is challenging because it may be difficult to satisfy security requirements while keeping the output of the circuit correct. For this purpose, we use threshold implementation proposed by Nikova *et al.* [25] and later extended by Bilgin *et al.* [6].

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a part of the circuit and $\bar{F}$ be the corresponding layer in the threshold implementation. The function $\bar{F} : \mathbb{F}_2^{ns_x} \to \mathbb{F}_2^{ms_y}$ is a *masking* of F, where s_x and s_y are the number of shares per input and output bit respectively. For $i \in \{0, ..., s_y - 1\}$, the i^{th} share of $\bar{F}$ is denoted by $F_i : \mathbb{F}_2^{ns_x} \to \mathbb{F}_2^m$. The security of a threshold implementation relies on the following properties of a masking.

Definition 6 (Properties of a masked function [25, 6]) *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function and $\bar{F} : \mathbb{F}_2^{ns_x} \to \mathbb{F}_2^{ms_y}$ be a masking of F. The masking $\bar{F}$ is said to be*

- *correct if $\sum_{i=0}^{s_{F(x)}-1} F_i(x_0, ..., x_{s_x-1}) = F(\sum_{i=0}^{s_x-1} x_i)$ for all $x_0, ..., x_{s_x-1} \in \mathbb{F}_2^n$*
- *non-complete if each F_i depends on at most $s_x - 1$ input shares*
- *uniform if $\bar{F}$ maps a uniform random masking of any $x \in \mathbb{F}_2^n$ to a uniform random masking of $F(x) \in \mathbb{F}_2^m$*

2.3 Glitch-Extended Bounded-Query Probing Security

2.3.1 Security Model

The threshold probing model was introduced by Ishai *et al* [17] in 2003 and in 2020 Beyne *et al* presented a variant of that model where the adversary can only make a bounded number of queries [5]. Since we work on hardware and have to take glitches into account, we use the glitch extended version of Ishai's *et al* probing model introduced by Faust *et al.* [15]. Here, instead of only the probed wire value, the adversary gets to see the values of all wires in a bundle. Since glitch does not propagate through memory gates, a glitch extended probe returns all values leading to the probed wire until a register is reached.

In the model of Beyne [5], the security of a circuit C with input k against a t-threshold-probing adversary is defined as follows. First, the challenger $\mathcal{C}$ picks a bit $b \in \{0, 1\}$ at random and gives the adversary $\mathcal{A}$ query access to an oracle $\mathcal{O}^b$. The adversary chooses up to t wires of the masked circuit to probe and sends them to the oracle along with the inputs k_0 and k_1. Both the plaintext and the key are considered the input of the circuit. The oracle gives back the values of the glitch-extended probed wire values of the masked circuit with the input k_b. After q queries, the adversary guesses a bit, which is denoted by $\mathcal{A}^{\mathcal{O}^b}$. The advantage of the adversary $\mathcal{A}$ is defined as an equivalent of the left-right security game,

$$Adv_{t-thr}(\mathcal{A}) = |Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - Pr[\mathcal{A}^{\mathcal{O}^1} = 1]|$$

The security model was further extended to a noisy probing variant [4] in 2022, where instead of the probed wire values, the adversary obtains noisy values. This is a

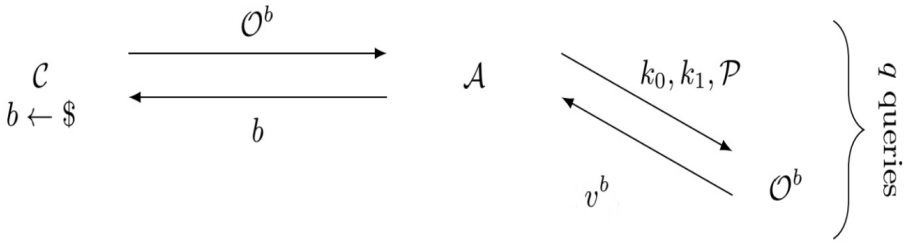


Figure 2.1: [4] The glitch-extended bounded-query probing security model with challenger $\mathcal{C}$, adversary $\mathcal{A}$, a left-right oracle $\mathcal{O}^b$, two inputs k_0, k_1, a set of probes $\mathcal{P}$ and the probed wire values v^b in the circuit $C(k_b)$

more realistic model in hardware security, where physical defaults of devices executing a masked algorithm may unintentionally contribute to the algorithm's security against DPA attacks.

Since in this work, we only use PROLEAD, which relies on the glitch-extended probing model, for the security analysis, the noise is not taken into account. Therefore, we rely on the noiseless model to bound the adversary's advantage in this thesis.

2.3.2 Bound on the advantage

In this subsection, we discuss the bound of the advantage of a glitch-extended bounded-query probing adversary.

We assume that all probed wire values can be divided into two groups, 'good' and 'bad'. The 'good' values jointly reveal nothing about the secret, but the 'bad' values can reveal some secret information. The leakage caused by the 'bad' values can be bounded by some function of ϵ. The definition of the bound on a first-order probing adversary is given below.

Theorem 1 *[5] Let $\mathcal{A}$ be a t-threshold-probing adversary for a circuit C. Assume that for every query made by $\mathcal{A}$ on the oracle $\mathcal{O}^b$, there exists a partitioning (depending only on the probed positions) of the probed wire values into two random variables $\mathbf{x}$ ('good') and $\mathbf{y}$ ('bad') such that*

- *The conditional probability distribution $p_{\mathbf{y}|\mathbf{x}}$ satisfies $\mathbb{E}_{\mathbf{x}}\|\hat{p}_{\mathbf{y}|\mathbf{x}} - \delta_0\|_2^2 \leq \epsilon$.*

- *Any t-threshold-probing adversary for the same circuit C and making the same oracle queries as $\mathcal{A}$, but which only receives the 'good' wire values (i.e. corresponding to $\mathbf{x}$) for each query, has advantage zero.*

The advantage of $\mathcal{A}$ can be upper bounded as,

$$Adv_{t-thr}(\mathcal{A}) \leq \sqrt{2q\epsilon} \tag{2.1}$$

where q is the number of queries to the oracle $\mathcal{O}^b$.

2.4 Fourier Analysis of Masked Function

2.4.1 Restrictions of Masked Functions

For any linear masking scheme, the set $\mathbb{V} \subseteq \mathbb{F}_2^l$ containing all valid masking of $0 \in \mathbb{F}_2^n$ is a vector space and the $\mathbb{F}_2$-linear masking scheme is an algorithm that maps a secret $x \in \mathbb{F}_2^n$ to a random element of the corresponding coset of $\mathbb{V}$. Suppose $\rho : \mathbb{F}_2^n \to \mathbb{F}_2^l$ maps a secret to their corresponding coset representative.

A function $\bar{F} : \mathbb{F}_2^l \to \mathbb{F}_2^l$ is a correct and uniform sharing of $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ if

$$\bar{F}(\rho(x) + \mathbb{V}) = \rho(F(x)) + \mathbb{V} \text{ for all } x \in \mathbb{F}_2^n$$

So, the restriction of $\bar{F}$ to $\rho(x) + \mathbb{V}$ is a well-defined function from $\rho(x) + \mathbb{V}$ to $\rho(F(x)) + \mathbb{V}$.

We define the absolute correlation matrix of the restriction of the function $\bar{F} : \mathbb{V}_a \to \mathbb{V}_b$ next. For this definition, the coset $a + \mathbb{V}$ is denoted by $\mathbb{V}_a$.

Definition 7 (Absolute correlation matrix [5]) *Let $\bar{F} : \mathbb{V}_a \to \mathbb{V}_b$ be a well-defined restriction of a shared function for $\mathbb{V} \subseteq \mathbb{F}_2^l$. The absolute correlation matrix $|C^{\bar{F}}|$ of $\bar{F}$ is defined by the matrix whose coordinates are*

$$|C^{\bar{F}}_{v,u}| = |C^{\bar{F}'}_{v,u}|$$

for $u, v \in \mathbb{F}_2^l/\mathbb{V}^\perp$, where $\bar{F}' : \mathbb{V} \to \mathbb{V}$ defined by, $\bar{F}'(x) = \bar{F}(x + a) + b$.

2.4.2 Correlation of Masking

In Theorem 1, the advantage of a probing adversary was bounded by $\|\hat{p}_{\mathbf{z}} - \delta_0\|_2$ where $p_{\mathbf{z}}$ is the probability distribution of a set of 'bad' values, conditioned on a set of 'good' values. The following theorem links $p_{\mathbf{z}}$ to the linear cryptanalysis of the shared circuit. This helps us to upper bound $\|\hat{p}_{\mathbf{z}} - \delta_0\|_2$ for a masked cipher.

Theorem 2 *[5] Suppose* $\mathbb{V} \subseteq \mathbb{F}_2^l$ *and the restriction of* $x \in \mathbb{V}$ *to the index set* $I = \{i_1, i_2, ..., i_m\}$ *is denoted by* $x_I = (x_{i_1}, ..., x_{i_m}) \in \mathbb{F}_2^{|I|}$. *Let* $F : \mathbb{V}_a \to \mathbb{V}_b$ *be a function. For* $\mathbf{x}$ *uniform random on* $\mathbb{V}_a$ *and* $\mathbf{y} = F(\mathbf{x})$, *let* $\mathbf{z} = (\mathbf{x}_I, \mathbf{y}_J)$, *where* $I, J \subset \{1, ..., l\}$. *The Fourier transformation of the probability mass function of* $\mathbf{z}$ *then satisfies* $|\hat{p}_{\mathbf{z}}(u, v)| = |C^F_{\tilde{v},\tilde{u}}|$, *where* $\tilde{u}, \tilde{v} \in \mathbb{F}_2^l/\mathbb{V}^\perp$ *are such that* $\tilde{u}_I = u$, $\tilde{u}_{[l]\setminus I} = 0$, $\tilde{v}_J = v$ *and* $\tilde{v}_{[l]\setminus J} = 0$

The relation between the linear approximations of F and $\hat{p}_{\mathbf{z}}$ provides a method to upper bound ϵ in Theorem 1 based on linear cryptanalysis.

If $\mathbf{z}$ is the observed value in the glitch-extended probing model and $(\#wires)$ is the number of bits observed by the value $\mathbf{z}$, then $|\text{supp } \hat{p}_{\mathbf{z}}| \leq (2^{(\#wires)})^{\#probes}$ and we can conclude the following,

$$\epsilon := \|\hat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\text{supp } \hat{p}_{\mathbf{z}}| \|\hat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq (2^{(\#wires)})^{\#probes} (|C^F_{v,u}|^{(\#active\_cells)})^2 \quad (2.2)$$

2.5 The Bound with Non-Uniform Inputs

The previous analysis was conducted by Beyne *et al.* [5], focusing on threshold implementations with uniform input. Later Dhooghe *et al.* [14] extended this work to analyze threshold implementations with non-uniform inputs. We apply their approach to other ciphers to verify its reliability and practical applicability.

The threshold implementations of Skinny and Present, presented in the following chapters, do not require fresh randomness for remasking. They only need initial randomness to mask the inputs. Here, we take a smaller amount of randomness compared to what is required for uniform masking and reuse it to mask the plaintext and examine the effect of entropy reduction on the first-order security. We omit the

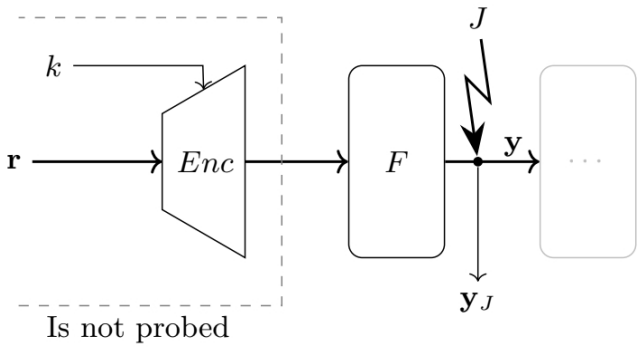


Figure 2.2: [14] *Enc* masking the input k using a few random bits $\mathbf{r}$

key masking for simplicity. This does not affect the outcome of the security evaluation implemented later in the thesis.

A non-uniform function *Enc* [Figure 2.2] is considered which takes the circuit's input and the random bits modeled as shares of zero and as 'bad' values and gives a shared input for the masked circuit.

Suppose $H = F \circ Enc$ be some function that maps the limited randomness to the probed values. Since *Enc* is non-uniform, the function H is also non-uniform. As a result, if for $\tilde{v} \in \mathbb{F}_2^l/\mathbb{V}^\perp$ with $\tilde{v}_J = v$ and $\tilde{v}_{[l]\setminus J} = 0$, $|\hat{p}_{\mathbf{y}_J}(v)| = |C^H_{\tilde{v},0}| \neq 0$, the probe is placed on $\mathbf{y}_J$ may reveal secrets. But the probability of success of the first-order can be bounded by some function of $\epsilon = |\hat{p}_{\mathbf{y}_J}(v)|$

We consider non-zero input linear masks subject to certain conditions depending on how the input is masked. Based on these masks, we identify linear trails over the uniform function F ending in one probe. This allows us to trace the dependency between the probed values and the initially masked input secret. The key is treated as a constant in our analysis, the linear trails do not include key schedule. In this way we can find the weak probing points (if any) and calculate the number of queries or traces needed for a bounded-query probing adversary to achieve an advantage of one and thus observe the leakage at the specified probing location.

Chapter 3

Skinny

Skinny [3] is a tweakable block cipher family. It has many versions that work on different blocks and tweakey sizes. In this chapter, we briefly explain the version of Skinny and it's masking scheme we work with. We also present the hardware architecture of Skinny that we use. Next, we provide two non-uniform inputs and analyze the security for both cases theoretically. The experimental results along with related discussions are given in Chapter 5.

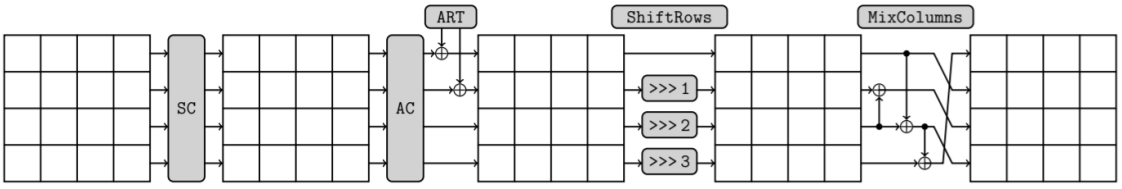


Figure 3.1: [3] SKINNY round function

3.1 Cipher Details

The version of Skinny used in this work operates on a 64-bit state, structured as a 4×4 matrix of 4-bit nibbles, and utilizes a 64-bit tweakey. It consists of 32 rounds, each comprising a non-linear substitution layer based on a 4-bit cubic S-box, followed by a linear diffusion layer. The diffusion layer includes XOR operations with constants and the round tweakey, a shift row transformation, and a mix columns operation. The constants are derived from the output of a 6-bit affine LFSR. In the tweakey schedule, the first 32 bits of the tweakey are XORed with the cipher state, followed by the application of a fixed permutation to the tweakey.

3.2 Masking Details

We design the threshold implementation of Skinny where the S-box is decomposed, following the techniques from the work of Bilgin *et al* [8], using the same quadratic function twice as follows.

$$S = A_3 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_1$$

with affine functions A_1, A_2, A_3 and the quadratic representative of an affine equivalence class Q_{294}, as described in [7]. These functions are defined as

$$\begin{aligned} A_1(x, y, z, w) &= (y + 1,\ z + 1,\ x,\ y + w) \\ A_2(x, y, z, w) &= (w,\ z,\ y,\ x) \\ A_3(x, y, z, w) &= (x + 1,\ y,\ w + 1,\ x + z + 1) \\ Q_{294}(x, y, z, w) &= (x,\ y,\ z + xy,\ w + xz) \end{aligned}$$

All the functions in Skinny round are shared via direct sharing with three shares. The affine functions here and the other linear round functions of Skinny are computed for each share independently. The masking of the quadratic function is described in Appendix A.

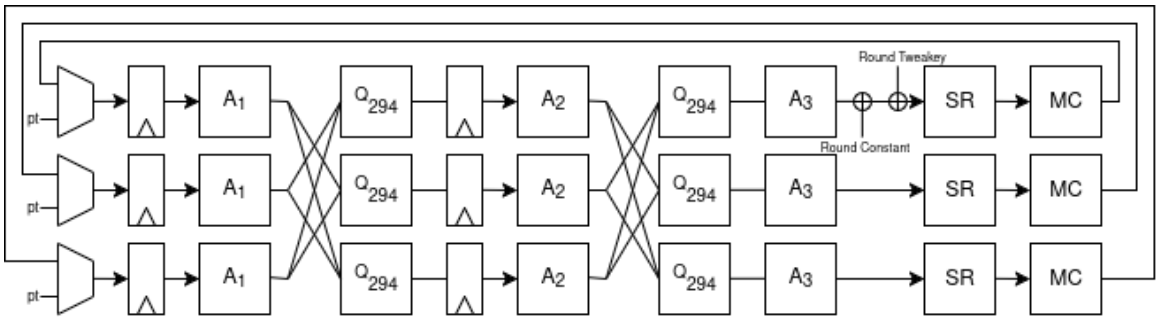


Figure 3.2: Round based architecture of Skinny

The pipelined structure [Figure 3.2] for the first-order secure threshold implementation of Skinny that we use has two register layers. The first one is applied before the S-box layer, whereas the second one is placed inside the S-box separating the internal non-linear layers. We consider the tweakey to be unmasked. So both the round constant and the round tweakey are added to the first share. This design takes two clock cycles to complete one round. So in total 62 clock cycles are needed to complete one encryption. The design was synthesized using the NANGATE 45nm

Library with the Synopsys Design Compiler tool, resulting in an area of 6054 Gate Equivalents (GE).

The maximum absolute correlation of the decomposed S-box of Skinny can be bounded above, as the following lemma suggests.

Lemma 1 (Correlation of Masked S-box of Skinny) *Let $\bar{S} : \mathbb{V}_a \rightarrow \mathbb{V}_b$ be any well-defined restriction of the masked Skinny S-box S for $\mathbb{V} \subseteq \mathbb{F}_2^l$. For any $u, v \in \mathbb{F}_2^l/\mathbb{V}^\perp$ such that $u \neq 0$, the coordinate of the absolute correlation matrix $|C^{\bar{S}}|$ of $\bar{S}$ satisfies $|C^{\bar{S}}_{u,v}| \leq 2^{-2}$.*

Claim 1 *In this implementation of Skinny, if a glitch extended probe is placed in the S-box, it can observe at most five wires.*

Proof:
If we place a glitch extended probe after the layer A_3, in the worst case, it will observe 5 bits.

- In A_3 layer, the fourth output bit of observes **two** input bits.
- In layer Q_{294}, these output bits observe **five** input bits in total.
- Layer A_2 does not affect.

If we place a glitch extended probe after the first Q_{294} layer, in the worst case, it will observe 5 bits.

- In Q_{299} layer, the third or fourth output bit of observes **five** input bits.
- In layer A_1, in both the cases these output bits observe **five** input bits in total.

□

3.3 Linear Trails

As seen in Chapter 2, we apply linear cryptanalysis to analyze the threshold implementations with non-uniform inputs. A nibble (4 bits) is said to be active if non-zero linear approximation is applied to that part. Our target is to find how this

activity propagates. In fact, we use linear trails to calculate the total number of active S-boxes through different rounds for a particular input activity pattern.

Even though the structure of Skinny is quite similar to Prince and Midori64, the MixColumns layer of Skinny makes it very different than those ciphers. Suppose we want to find the activity pattern of the cells at the beginning of Skinny such that at the end of the round only one cell (say, C) will be active. Depending on which row C belongs to, the number of active cells at the beginning changes. This interesting property makes the analysis of Skinny different from the work of Dhooghe *et al.* [14]

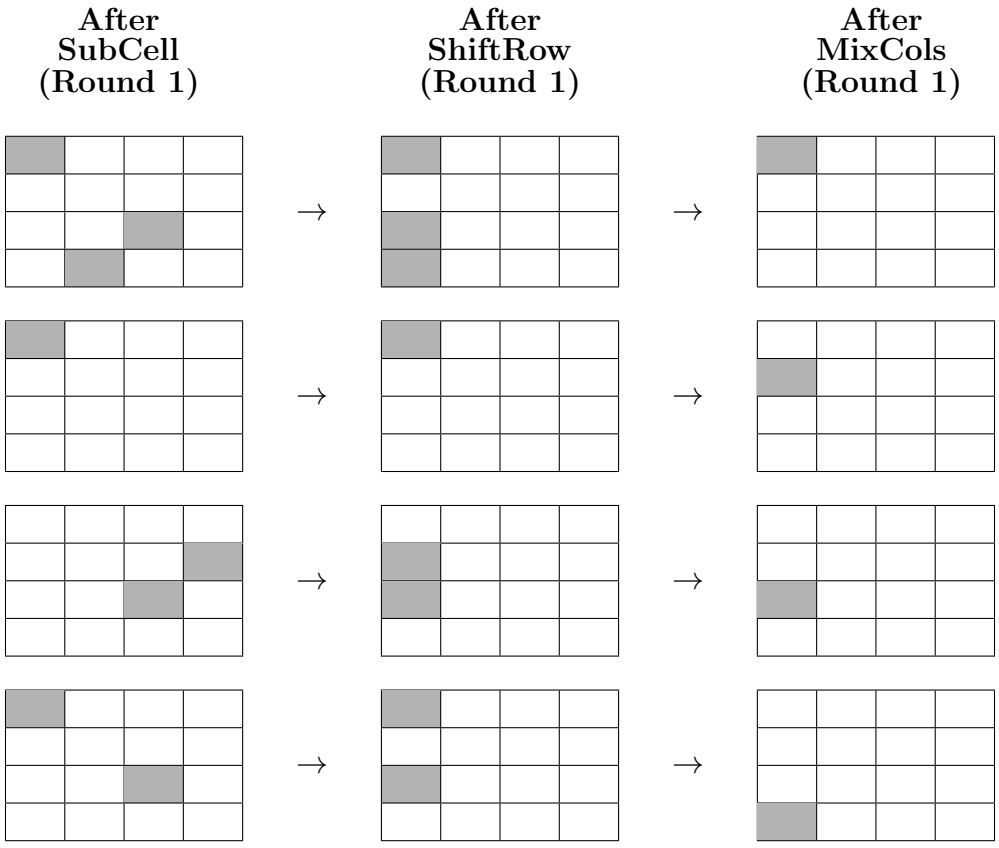


Figure 3.3: The trails of one round Skinny for all four cases with an active cell at the end

Since the maximum number of active S-boxes in one round is three, it follows from Claim 1 that a glitch extended probe can observe at most $3 \times 5 = 15$ wires.

3.4 Threshold Implementation with Non-Uniform Inputs

To mask a 64-bit plaintext using a three-share masking scheme, a total of 128 random bits are typically required. Let a and b be two random bits. A single bit s of the secret plaintext can then be masked by assigning the first share as a, the second share as b, and the third share as $a \oplus b \oplus s$.

Here instead of using 128 bits of randomness, we find patterns in which we can reuse less amount of randomness to mask the plaintext.

Due to the manner in which the plaintext is shared, either none or multiple cells masked with the same randomness are active simultaneously. Our objective is to analyze how these activity patterns propagate through the various round functions of Skinny. When a single probe covers the resulting output activity pattern, we compute an upper bound on the advantage of a bounded-query probing adversary, using Theorem 1.

3.4.1 Insecure non-uniform input

For masking the 64-bit plaintext, we take 32 bits of randomness and reuse them in the following pattern.

r_1	r_2	r_3	r_4
r_4	r_1	r_2	r_3
r_3	r_4	r_1	r_2
r_2	r_3	r_4	r_1

where each $r_i \in \mathbb{F}_2^8$ is random.

Observing the linear trails in Figure 3.3, we see that the output of the input activity pattern in Figure 3.4 can be covered by a single probe after one round.

As mentioned before a glitch extended probe can observe at most 15 wires. Since the probability that the adversary will probe any cell from the last row is $\frac{4}{16} = \frac{1}{4}$ and in this case the total number of active S-boxes is 2, from Equation 2.2 we get

$$\epsilon \approx \tfrac{1}{4} \times 2^{15}((2^{-2})^2)^2 = 2^5 > 1$$

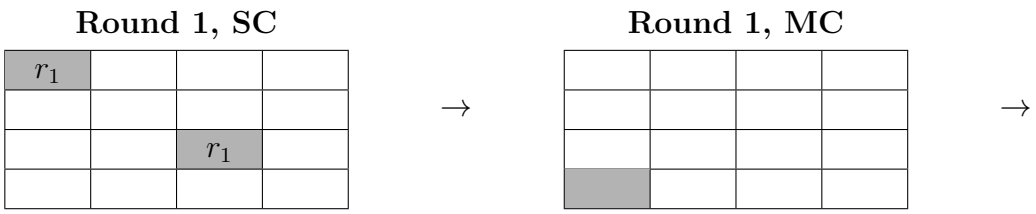


Figure 3.4: Linear trail for one round when a single-bit probe is placed in the active cell at the last row

This implies that the non-uniformity for this input in very high and no valid bound for a probing adversary can be calculated in this case.

3.4.2 Secure non-uniform input

For the next case we again use 32 random bits to mask the plaintext but with a different pattern. Here each cell in a row is masked with the same randomness. That is, the random bits $r_i \in \mathbb{F}_2^8$ are used in the following manner.

r_1	r_1	r_1	r_1
r_2	r_2	r_2	r_2
r_3	r_3	r_3	r_3
r_4	r_4	r_4	r_4

Since all cells from the same row are masked with the same random bits, either none or more than one cell in a row should be activated at the beginning. We check all possible trails of Skinny to find such an activity pattern of the given input for which ends in a single active cell. All the trails for one round are in Figure 3.3 and the trails for two and three rounds are given in Figure B.1. For one and two rounds, no such pattern exists. However, we find a trail for which the following activity pattern results in only one active cell in the first row after three rounds given in Figure 3.5. If any cell from the rest of the rows (second, third or last) is probed after three rounds, the trails can not be activated by the proposed reuse of randomness.

Similarly as the previous case, we notice the following points. The probability that a cell from the first row is probed by the adversary is $\frac{4}{16} = \frac{1}{4}$. A glitch-extended probe can see at most 15 wires. In the trail for three rounds [Figure 3.5], the total number of active S-boxes is 21. Hence ϵ can be bounded using Equation 2.2 as,

$$\epsilon := \|\hat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\text{supp } \hat{p}_{\mathbf{z}}| \|\hat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq \tfrac{1}{4} \times 2^{15}((2^{-2})^{21})^2 = 2^{-71}$$

So using Equation 2.1 we can bound the advantage of a threshold probing adversary as

$$Adv_{2-thr}(\mathcal{A}) \leq \sqrt{2q\epsilon} \leq \sqrt{2q2^{-71}} = \sqrt{q2^{-70}}$$

This implies the advantage of the adversary should be equal to 1 after 2^{70} queries, which is a very large number. So, we say that this non-uniform input is secure.

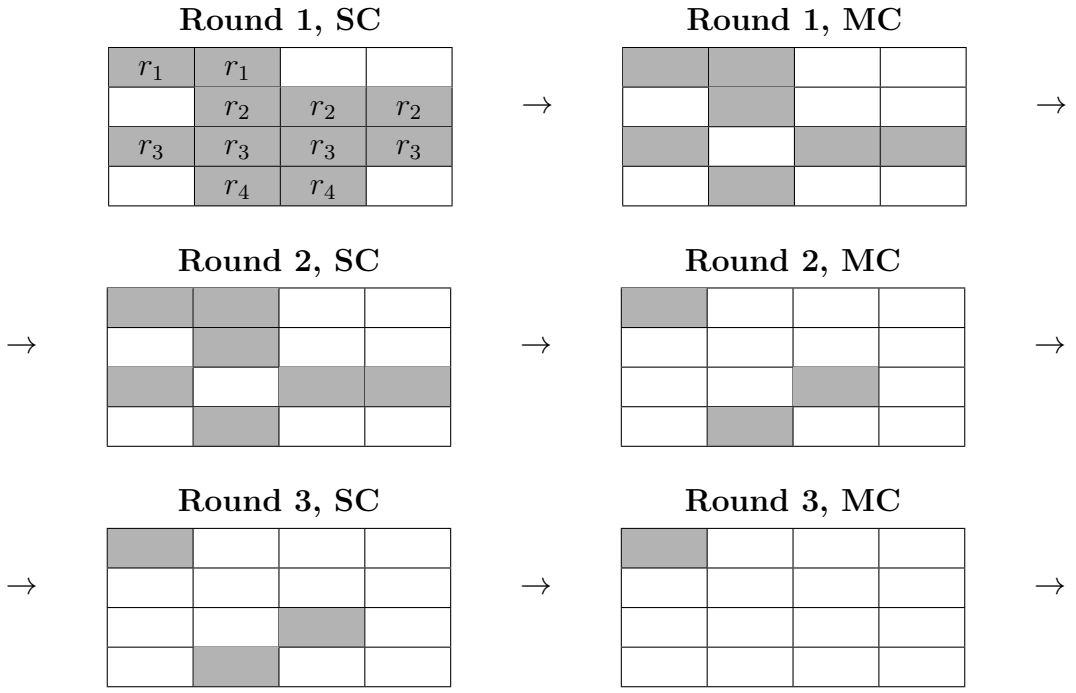


Figure 3.5: Linear trail for three rounds when a single-bit probe is placed in the active cell at the first row

Chapter 4

Present

In this chapter, we describe the cipher and its masking scheme, as well as the hardware architecture used in our work. We then present a theoretical analysis of the threshold implementation of Present [10] with two different non-uniform inputs. The experimental results and related discussions for these cases are provided in Chapter 5.

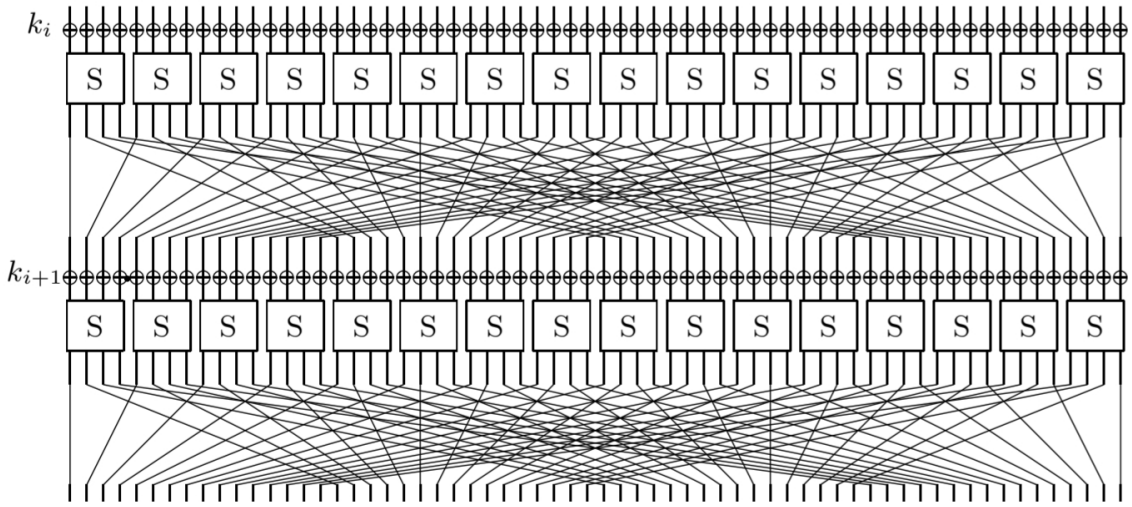


Figure 4.1: [10] The S/P network for Present

4.1 Cipher Details

Present[10] is a lightweight block cipher consisting of 31 rounds. It operates on 64-bit plaintext blocks using an 80-bit key. Each round includes an XOR with the round key, followed by a non-linear substitution layer based on a 4-bit cubic S-box and a linear bit wise permutation. In each round, the 64 leftmost bits of the current 80-bit key are used as the round key. Subsequently, the key is updated using a combination of affine and non-linear transformations.

Since the linear layer in this cipher is a bit permutation and the analysis of the threshold implementation is based on linear trails, the study of Present with non-uniform inputs becomes different than Skinny. To find the linear trails for more than one round, we use the linear approximation table of Present S-box. Because of the bit permutation layer, the number of active S-boxes is not always fixed for same number of rounds. It depends on the wire probed. We take the worst case, that is the minimum number of active S-boxes, to calculate the bound on the advantage of the probing adversary.

To make the security analysis easier, we divide all 16 S-boxes into four groups, as done by Bogdanov *et al.* [10]. Each group contains four S-boxes. Observing the permutation layer of the Present cipher, we note the following.

Observations:

1. The four output bits from a particular S-box enter four distinct S-boxes, each of which belongs to a distinct group in the subsequent round.

2. The four output bits from the q-th S-box of a group become the q-th bit of four distinct S-boxes.

3. The four input bits of an S-box come from the outputs of four different S-boxes from the same group.

4. The input bits to a group of four S-boxes come from 16 different S-boxes.

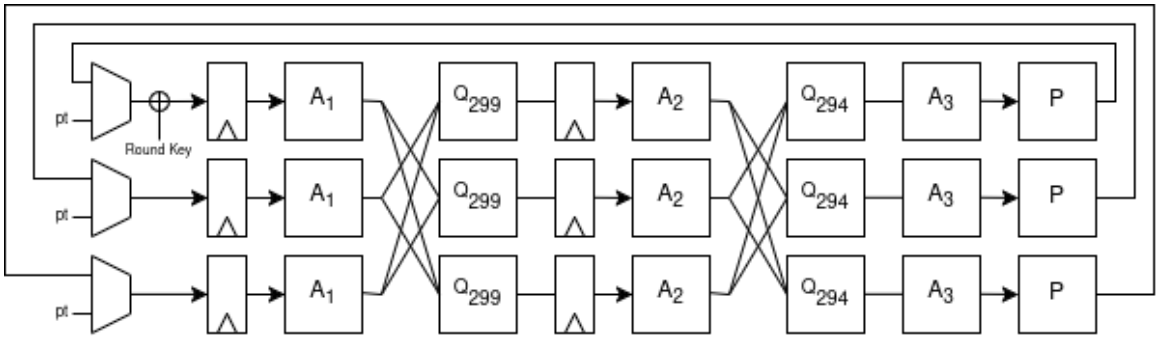


Figure 4.2: Round based architecture of Present

4.2 Masking Details

We use a pipelined structure for the first-order secure threshold implementation of Present. The round based architecture is described in Figure 4.2. Present S-box is a cubic permutation that is decomposed in the following way in this work.

$$S = A_3 \circ Q_{294} \circ A_2 \circ Q_{299} \circ A_1$$

Here A_1, A_2 and A_3 are affine layers and Q_{294}, Q_{299} are quadratic classes. All the affine and quadratic functions are described below.

$$\begin{aligned}
A_1(x,y,z,w) &= (y+z,\ y+w,\ x+w,\ x),\\
A_2(x,y,z,w) &= (y,\ z,\ w,\ x),\\
A_3(x,y,z,w) &= (z+w+1,\ x+z+1,\ x+z+w,\ y),\\
Q_{294}(x,y,z,w) &= (x,\ y,\ z+xy,\ w+xz),\\
Q_{299}(x,y,z,w) &= (x,\ y+xy+xz,\ z+xy+xz+xw,\ w+xy+xw)
\end{aligned}$$

All the functions in Present are shared via direct sharing with three shares. For the affine functions and the permutation layer, each share is processed independently. The masking details for the quadratic functions are given in Appendix A.

In this design, there are two registers. The first one is placed before the S-box layer and the other one separates the non-linear layers inside the S-box. Since the round key is considered as a constant, it is added to the first share. The design needs 62 clock cycles for one encryption, two clock cycles per round. The design is synthesized using the NANGATE 45nm Library with the Synopsys Design Compiler tool and the area is 6631 Gate Equivalents (GE).

The threshold implementation has the following property, where the maximum absolute correlation of the masked S-box can be bounded above.

Lemma 2 (Correlation of Masked S-box of Present) *Let $\bar{S} : \mathbb{V}_a \to \mathbb{V}_b$ be any well-defined restriction of the masked Present S-box S for $\mathbb{V} \subseteq \mathbb{F}_2^l$. For any $u, v \in \mathbb{F}_2^l/\mathbb{V}^\perp$ such that $u \neq 0$, the coordinate of the absolute correlation matrix $|C^{\bar{S}}|$ of $\bar{S}$ satisfies $|C^{\bar{S}}_{u,v}| \leq 2^{-2}$.*

As seen in Chapter 2, we also need the maximum number of bits observed by a glitch-extended probe to calculate the bound of a probing adversary.

Claim 2 *In this implementation of Present, a glitch-extended probe can see at most eight wires.*

Proof:
If we place a glitch extended probe after the layer P, in the worst case, it will observe 7 bits.

- P layer does not affect.
- In A_3 layer, the third output bit of observes **three** input bits.
- In layer Q_{294}, these output bits observe **seven** input bits in total.
- Layer A_2 does not affect.

If we place a glitch extended probe after the layer Q_{299}, in the worst case, it will observe 8 bits.

- In Q_{299} layer, the third output bit of observes all **eight** input bits.
- In layer A_1, these output bits observe **eight** input bits in total.

□

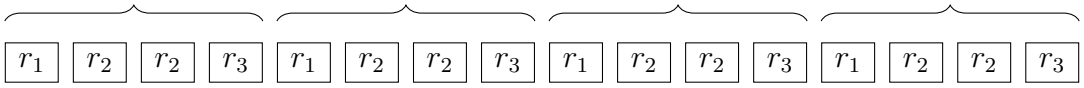
4.3 Threshold Implementation with Non-Uniform Inputs

To mask a nibble (4 bits) of plaintext with three sharing, we need eight bits of randomness. So for uniform three-sharing, a total of 128 bits of randomness are required. Instead of 128 bits, we take less bits of randomness and repeat them to mask the plaintext.

We then analyze the activity patterns through the rounds of masked Present with these inputs. If a nibble masked with r_i is activated, then at least one other nibble that is masked with the same r_i should be activated. We try to find activity patterns following the previous constraint, for which the output pattern can be covered by a single probe.

4.3.1 First Example of Non-uniform input

At first, we use only 24 bits of randomness to mask the plaintext. Each 16 bits of plaintext are masked using the randomness pattern ($r_1r_2r_2r_3$), where each r_i is random and contains 8 bits. Thus, to mask the 64 bits of plaintext r_1, r_2 and r_3 are used the following way.



For one round, a single probe will cover the output pattern only if exactly one S-box is active at the beginning. Clearly, this cannot be the case, as previously explained.

If more than one S-box masked with r_i ($i = 1$, or 3) is activated, they must be from different groups. So, a single probe will not be enough to cover the output pattern after two rounds [Observation 3].

Suppose two S-boxes from the same group masked with r_i ($i = 2$) are activated. In order to cover the output pattern after two rounds with a single probe, only one S-box should be activated in the second round with only one bit output mask. This is only possible if the S-box in the second round has an input mask 0110 [Observation 2]. We see from the LAT of Present [Figure C.1], the biases of the linear approximations with input mask 0110 and one bit output mask are $\frac{1}{2}$. As a result of this discussion, we conclude that the input pattern is secure up to two rounds.

To cover the output pattern using one probe after three rounds, we know that the first round should have at least two active S-boxes and the third round should have exactly one. We have seen that if in round one the S-boxes masked with r_2 from the same group are active, the number of active S-boxes in three respective rounds cannot be two, one, one. If in the first round two S-boxes from different groups are active, in the next round more than one S-box will be activated [Observation 3]. So,

in total at least five S-boxes will be activated in three rounds to be able to observe the output pattern with one probe. An example of such a trail is given in Figure C.2.

Since the maximum absolute correlation of the masked S-box is 2^{-2} and a probe in the S-box sees at 8 bits, we see that [from Equation 2.2]

$$\epsilon := \|\hat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\text{supp } \hat{p}_{\mathbf{z}}| \|\hat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq 2^8((2^{-2})^5)^2 = 2^{-12}$$

So, the advantage of a threshold probing adversary can be bound as [using Equation 2.1]

$$Adv_{2-thr}(\mathcal{A}) \leq \sqrt{2q\epsilon} \leq \sqrt{2q2^{-12}} = \sqrt{q2^{-11}}$$

In the experiment, we should see leakage in the third round after about $2^{11} = 2048$ traces, which is very low.

In the LAT of Present [Figure C.1], we notice the following facts.

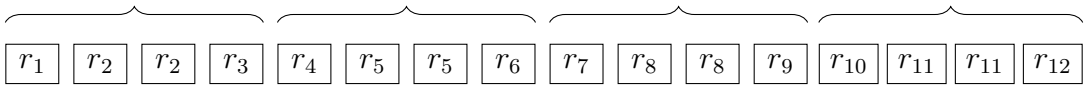
- The biases of the linear approximations with input mask 0110 or 0001 and one bit output mask are $\frac{1}{2}$.
- For any input mask (except the two mentioned above), there is at least one 1-bit output mask such that the bias of the linear approximation is not $\frac{1}{2}$.

The above facts imply that if the first S-box of the i-th group ($i = 1$ or 4) is masked with r and if any other S-box, except the third S-boxes of the groups other than the i-th group, is masked with the same r, we will find a trail activating those two S-boxes masked with r and that end with one probe exactly after the second or the third round, activating three or five S-boxes respectively. If we try to use 32 bits (or less) and repeat them to mask the nibbles of the plaintext, we will always find such a case. So it is not possible to reduce the required randomness 75% and find a non-uniform secure mask for Present.

In the following section, we give another example of a non-uniform input using more randomness than before. However, it remains insecure.

4.3.2 Second Example of Non-uniform input

For the second case, we take 96 random bits to mask the plaintext. The pattern in which these random bits are reused to mask the plaintext is given below. Here $r_i \in \mathbb{F}_2^8$ for $i \in \{1, 2, ..., 12\}$.



The analyses for one and two rounds are similar to the previous case.

For one round, one probe can cover the activity pattern only if exactly one S-box is activated at the beginning of the round. This is not possible since if an S-box masked with r_i is activated, at least one other S-box masked with r_i should be activated.

If both S-boxes masked with the same randomness are activated, then one probe will cover the output pattern after two rounds only if the S-box activated has an input mask 0110 and a one bit output mask. But LAT of Present S-box [Figure C.1] shows that these linear approximations have bias $\frac{1}{2}$.

The S-boxes masked with the same randomness belong to the same group. If both S-boxes masked with the same randomness from the same group are activated, they will either activate a single S-box in the next round or more than one S-boxes from different groups. If in the second round, only one S-box is activated and if it has an input mask 0110 and a one bit output mask, only then the output pattern can be covered by a single probe after the third round. But we have seen that this is not possible. If in the second round two S-boxes from different groups are activated, in the next round always more than one S-box will be activated [Observation 3].

That said, this input pattern is secure up to three rounds.

With the similar analysis we find that the best trail for four rounds, where one probe can cover the output pattern, activates at least 7 S-boxes [Figure C.3 in Appendix C].

Since the maximum absolute correlation of the masked S-box is 2^{-2} and a probe in the S-box sees at 8 bits, we see that [from Equation 2.2]

$$\epsilon := \|\hat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\text{supp } \hat{p}_{\mathbf{z}}| \|\hat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq 2^8((2^{-2})^7)^2 = 2^{-20}$$

So, the advantage of a threshold probing adversary can be bound as [using Equation 2.1]

$$Adv_{2-thr}(\mathcal{A}) \leq \sqrt{2q\epsilon} \leq \sqrt{2q2^{-20}} = \sqrt{q2^{-19}}$$

In the experiment, we should see leakage in the third round after about $2^{19} = 524288$ traces.

Chapter 5

PROLEAD Experiments

PROLEAD is a hardware leakage detection tool. It was introduced by Müller *et al.* [24] in 2022. It performs gate level logic simulation to analyze the statistical independence of the values probed by a robust probing adversary. The tool handles full masked cipher implementations and detects flaws with respect to the occurrence of glitches without the need for any power model. To evaluate leakage, PROLEAD checks whether the observed distributions of intermediate values differ significantly from the ideal distribution. During the evaluation, noise is not considered.

PROLEAD has two modes, normal and compact mode. In normal mode, all observations of the adversary are represented as contingency tables that capture the joint distribution of probe outputs and secrets. In compact mode, the summary of the observations based on their Hamming weight is stored. Although normal mode provides more accurate results, the contingency table becomes large, resulting in high memory usage. Here we use compact mode, since it delivers less time-consuming, computation-friendly verification.

We start by checking the security our implementations against first order glitch-extended probing adversary with *uniform* inputs for both ciphers to verify the security of the proposed masking solutions. Next we apply the proposed non-uniform inputs to those designs and evaluate leakage. To reuse randomness, we add an additional module to our design and exclude the module from the probing list. The results of those experiments are described below.

Cipher	Case	Passed	#Traces	#Cycle	#Round
Skinny	Uniform	Yes	102.4M	NA	NA
	Insecure	No	10240	4,5,6,7,8	2,3,4
	Secure	Yes	102.4M	NA	NA
Present	Uniform	Yes	102.4M	NA	NA
	First Example	No	10240	6,7,8,9,...	3,4,...
	Second Example	No	10240	6,7,8	3,4

Table 5.1: Results of PROLEAD experiments of Skinny and Present with maskings from Section 3.4 and Section 4.3

5.1 Skinny

We take up to 100M simulations for the implementations with the uniform input and the secure non-unifom input. Both implementations do not leak as predicted. The graphs of leakage progression ($log_{10}(p)$) with the number of simulations for both cases are shown in Figure 5.1. The evolution of the curves does not indicate any growth in leakage as well.

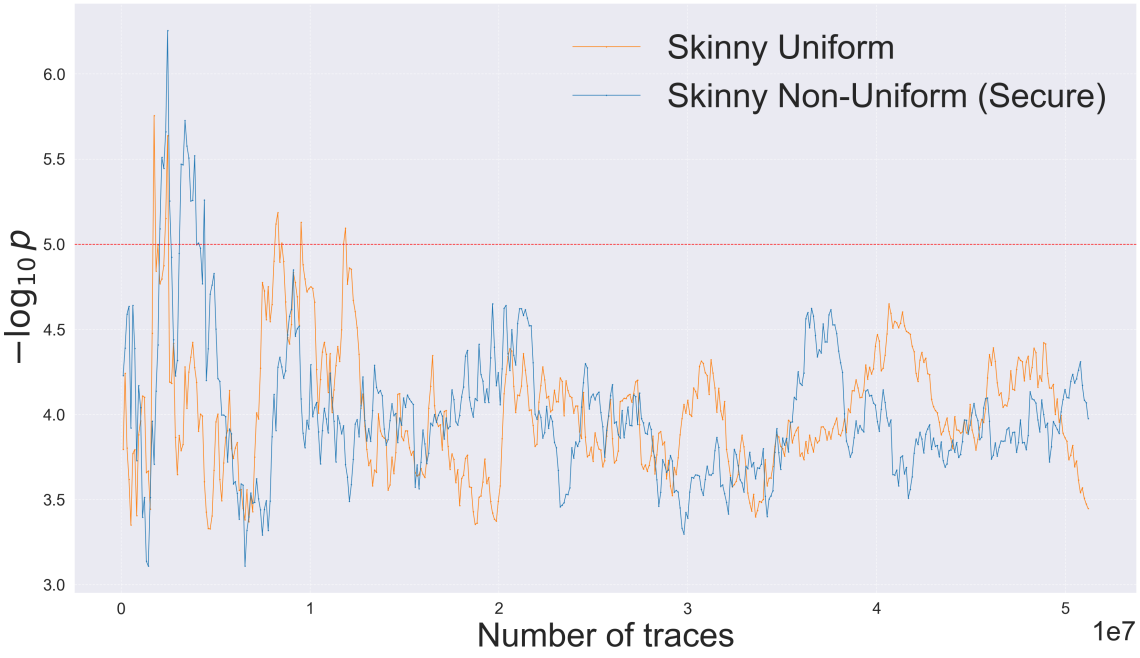


Figure 5.1: Implementation of Skinny with uniform masking and Skinny with (secure) non-uniform masking

For the implementation with the insecure input, we take 10M traces and observe leakage immediately exceeding the threshold for statistical tests by a large margin, concentrated only on the clock cycles 4 to 8. During the cipher's computation progress, the non-uniform randomness is further processed, making it increasingly indistinguishable from the uniform randomness. As a result, no leakage is observed in the later clock cycles.

5.2 Present

To test the Present implementation masked with uniform input, 100M traces were taken. As expected, we do not detect any leakage. The graph of the highest leakage progression ($log_{10}(p)$) with the number of simulations is given in Figure 5.2.

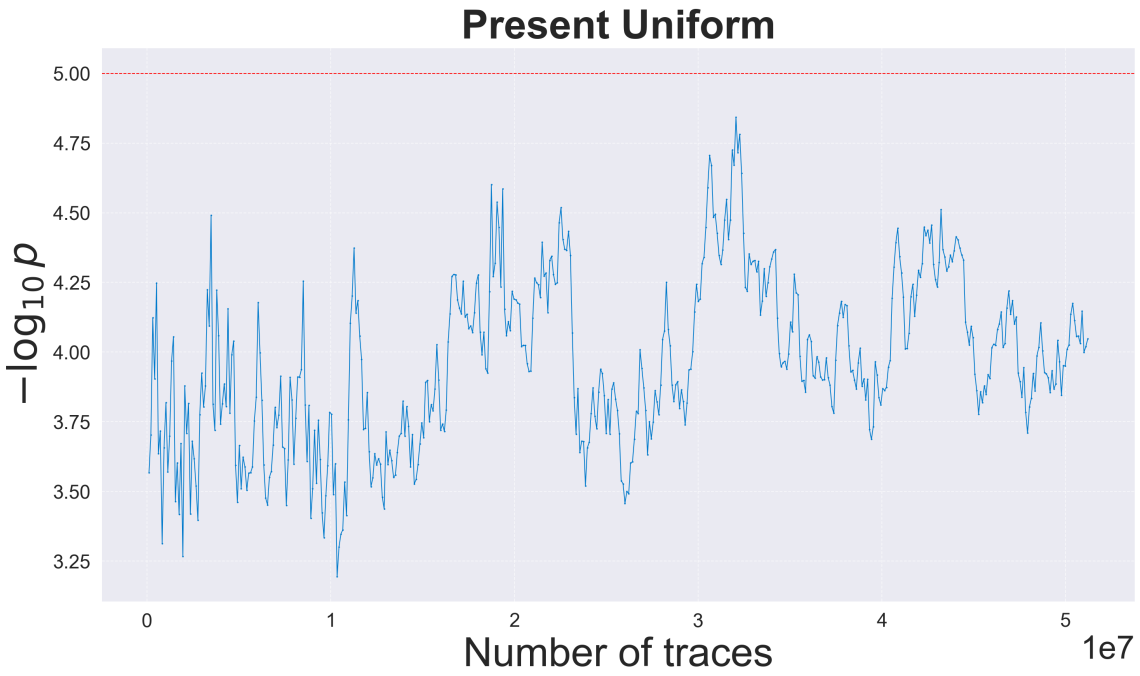


Figure 5.2: Implementation of Present with uniform masking

For the first non-uniform input example [Section 4.3.1], for 100M simulations, we do not observe any leakage before clock cycle 6, supporting our prediction that the implementation masked with this non-uniform input is secure up to two rounds. For round three (clock cycle 6), we take the test with smaller steps and the reports

show that around 1536 simulations the leakage starts to exceed the threshold for the statistical tests [Figure 5.3]. This also proves the correctness of the bound of advantage of an adversary calculated theoretically. For some of the next clock cycles, we see that the leakage grows with the number of simulations. Then, for some clock cycles the amount of leakage oscillates and finally grows. Due to this reuse of masks and the design of the permutation layer, it is possible for the applied masks to cancel out, resulting in the unmasked plaintext being processed later stages explaining the leakage there. The graphs showing how the leakage evolves for some of the later clock cycles are given in Appendix D.

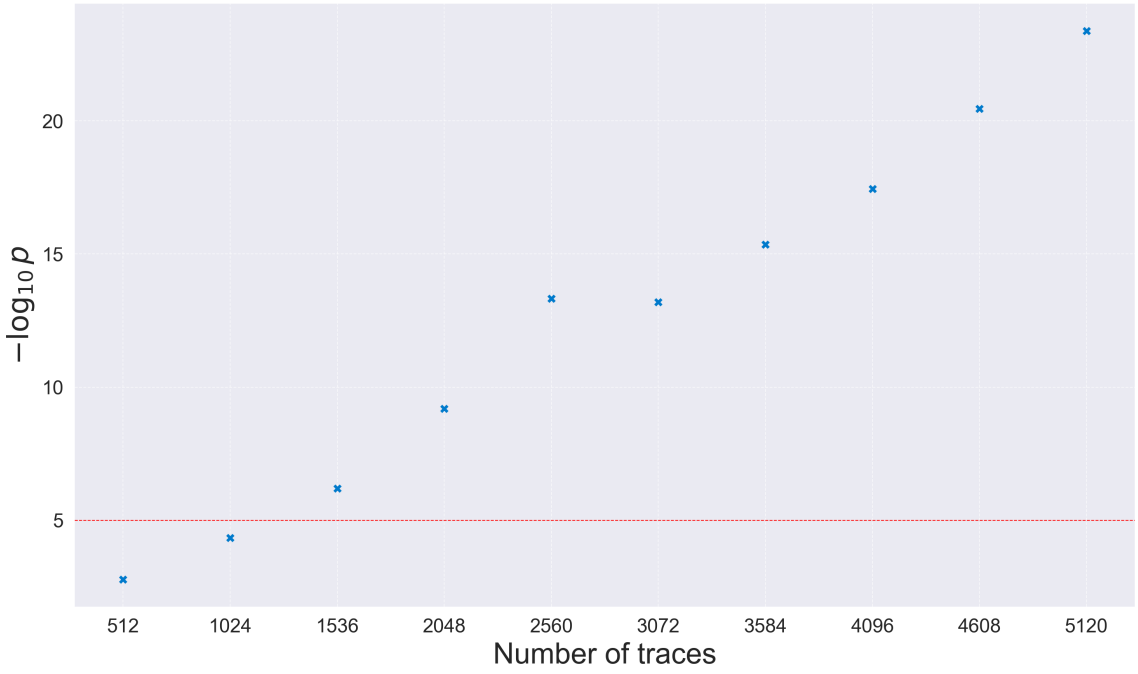


Figure 5.3: Leakage progression in clock cycle 6 for Present with non-uniform masking in Section 4.3.1

For the second example of non-uniform input [Section 4.3.2], we take up to 100M simulations. We observe leakage in the fourth round as predicted by our theoretic analysis. However, the reports indicate the leakage in the third round as well that is not in line with the discussion on the security, provided in Chapter 4. Despite verifying the correctness of the setup and making no changes to the design since the uniform input tests, we continue to observe the leakage after multiple repetitions with slightly altered fixed inputs. There are two possible reasons for that: a bug in a setup utilized for the test or a real leakage that may not be covered by our analysis. To confirm the first reason, multiple tests are required, including tests on a physical device as FPGA with application of TVLA [2], which is out of scope of this work. In

case the first reason is not valid, meaning that the setup worked correctly, we suggest further analyzing Present as a potential future work. Nevertheless, we should denote that the leakage is not observed after the fourth round, verifying the hypothesis about concentrated leakage.

Chapter 6

Conclusion

In this work, our main objective was to check whether the initial randomness required to mask the plaintext for threshold implementations can be reduced while the implementation stays first order secure.

For the cipher Skinny, we successfully achieved our goal. We used 75% less random bits to mask the plaintext in such a way that the bound for the advantage of a threshold probing adversary was so high that in practice it can be considered secure. The PROLEAD experiments support our claim. We also observed that using the same number of randomness in a different pattern caused leakage at a very early stage.

However, for the bit permutation in Present structure, the study was very different in this case. We tested the example of a non-uniform masking with only 24 bits of randomness that stays secure up to the second round, but leaks in the third round immediately. We also saw that if the initial randomness needed to mask the plaintext is reduced by 75% or more, it does not remain first order secure. For the second example, we saw a difference in the theoretic and experimental results. We predict that this could be prevented through the more thorough bit-wise analysis of the trails for Present or ruled out as a setup bug in case of existing TVLA confirmation.

The ultimate goal for this research can be creating an automated tool to do the similar security analysis. Finding ways to reuse fresh randomness with similar analysis can also be an interesting direction to explore.

Acknowledgments. We used the help of ChatGPT for the code, creating the graphs and tables to make our work time efficient, and to improve certain sentences for better clarity. We also thank Tim Beyne for providing us the code to calculate the maximum correlation of masked S-boxes.

Bibliography

[1] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 411–436. Springer (2015)

[2] Becker, G.T., Cooper, J., DeMulder, E.K., Goodwill, G., Jaffe, J., Kenworthy, G., Kouzminov, T., Leiserson, A.J., Marson, M.E., Rohatgi, P., Saab, S.: Test vector leakage assessment (tvla) methodology in practice (2013)

[3] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The skinny family of block ciphers and its low-latency variant mantis. In: Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. pp. 123–153. Springer (2016)

[4] Beyne, T., Dhooghe, S., Moradi, A., Shahmirzadi, A.R.: Cryptanalysis of efficient masked ciphers: Applications to low latency. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 679–721 (2022)

[5] Beyne, T., Dhooghe, S., Zhang, Z.: Cryptanalysis of masked ciphers: A not so random idea. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part I 26. pp. 817–850. Springer (2020)

[6] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Trade-offs for threshold implementations illustrated on aes. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34(7), 1188–1200 (2015)

[7] Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3× 3 and 4× 4 s-boxes. In: International workshop on cryptographic hardware and embedded systems. pp. 76–91. Springer (2012)

[8] Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Tokareva, N.N., Vitkup, V.: Threshold implementations of small s-boxes. Cryptogr. Commun. 7(1), 3–33 (2015), https://doi.org/10.1007/s12095-014-0104-7

[9] Blakley, G.R.: Safeguarding cryptographic keys. In: Managing requirements knowledge, international workshop on. pp. 313–313. IEEE Computer Society (1979)

[10] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9. pp. 450–466. Springer (2007)

[11] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: Prince–a low-latency block cipher for pervasive computing applications. In: Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18. pp. 208–225. Springer (2012)

[12] Cassiers, G., Masure, L., Momin, C., Moos, T., Moradi, A., Standaert, F.X.: Randomness generation for secure hardware masking-unrolled trivium to the rescue. Cryptology ePrint Archive (2023)

[13] Desmedt, Y.: Some recent research aspects of threshold cryptography. In: International Workshop on Information Security. pp. 158–173. Springer (1997)

[14] Dhooghe, S., Ovchinnikov, A.: Threshold implementations with non-uniform inputs. Cryptology ePrint Archive (2023)

[15] Faust, S., Grosso, V., Del Pozo, S.M., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 89–120 (2018)

[16] Groß, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. Cryptology ePrint Archive (2016)

[17] Ishai, Y., Sahai, A., Wagner, D.: 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 23. pp. 463–481. Springer (2003)

[18] Knichel, D., Moradi, A., Müller, N., Sasdrich, P.: Automated generation of masked hardware. Cryptology ePrint Archive (2021)

[19] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. crypto'99, vol. 1666 of lecture notes in computer science (1999)

[20] Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked cmos gates. In: Cryptographers' Track at the RSA Conference. pp. 351–365. Springer (2005)

[21] Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked aes hardware implementations. In: International workshop on cryptographic hardware and embedded systems. pp. 157–171. Springer (2005)

[22] Matsui, M.: Linear cryptanalysis method for des cipher. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 386–397. Springer (1993)

[23] Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C., Yung, M., Lin, D. (eds.) Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011), `https://doi.org/10.1007/978-3-642-34704-7_5`

[24] Müller, N., Moradi, A.: Prolead: A probing-based hardware leakage detection tool. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 311–348 (2022)

[25] Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: International conference on information and communications security. pp. 529–545. Springer (2006)

[26] Wu, L., Fan, Y., Preneel, B., Wang, W., Wang, M.: Automated generation of masked nonlinear components: From lookup tables to private circuits. In: International Conference on Applied Cryptography and Network Security. pp. 319–339. Springer (2024)

[27] Yao, A.C.: Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (sfcs 1982). pp. 160–164. IEEE (1982)

Appendix A

Masking of Quadratic Functions

The masking of the quadratic functions for three shares with $i \in \{0, 1, 2\}$, where the convention is that the superscripts wrap around modulo 3.

The function $Q_{294}(x, y, z, w) = (a,\ b,\ c,\ d) = (x,\ y,\ z + xy,\ w + xz)$ is masked as follows:

$$\begin{aligned}
a_{i-1} &= x_i, \\
b_{i-1} &= y_i, \\
c_{i-1} &= z_i + (x_i y_i + x_i y_{i+1} + x_{i+1} y_i), \\
d_{i-1} &= w_i + (x_i z_i + x_i z_{i+1} + x_{i+1} z_i).
\end{aligned}$$

The function $Q_{299}(x, y, z, w) = (a,\ b,\ c,\ d) = (x,\ y + xy + xz,\ z + xy + xz + xw,\ w + xy + xw)$ is masked as follows:

$$\begin{aligned}
a_{i-1} &= x_i, \\
b_{i-1} &= y_i + (x_i y_i + x_i y_{i+1} + x_{i+1} y_i) + (x_i z_i + x_i z_{i+1} + x_{i+1} z_i), \\
c_{i-1} &= z_i + (x_i y_i + x_i y_{i+1} + x_{i+1} y_i) + (x_i z_i + x_i z_{i+1} + x_{i+1} z_i) \\
&\qquad (x_i w_i + x_i w_{i+1} + x_{i+1} w_i), \\
d_{i-1} &= w_i + (x_i y_i + x_i y_{i+1} + x_{i+1} y_i) + (x_i w_i + x_i w_{i+1} + x_{i+1} w_i).
\end{aligned}$$

Both the masking are uniform and non-complete.

Appendix B

Skinny: Trails for Two and Three Rounds

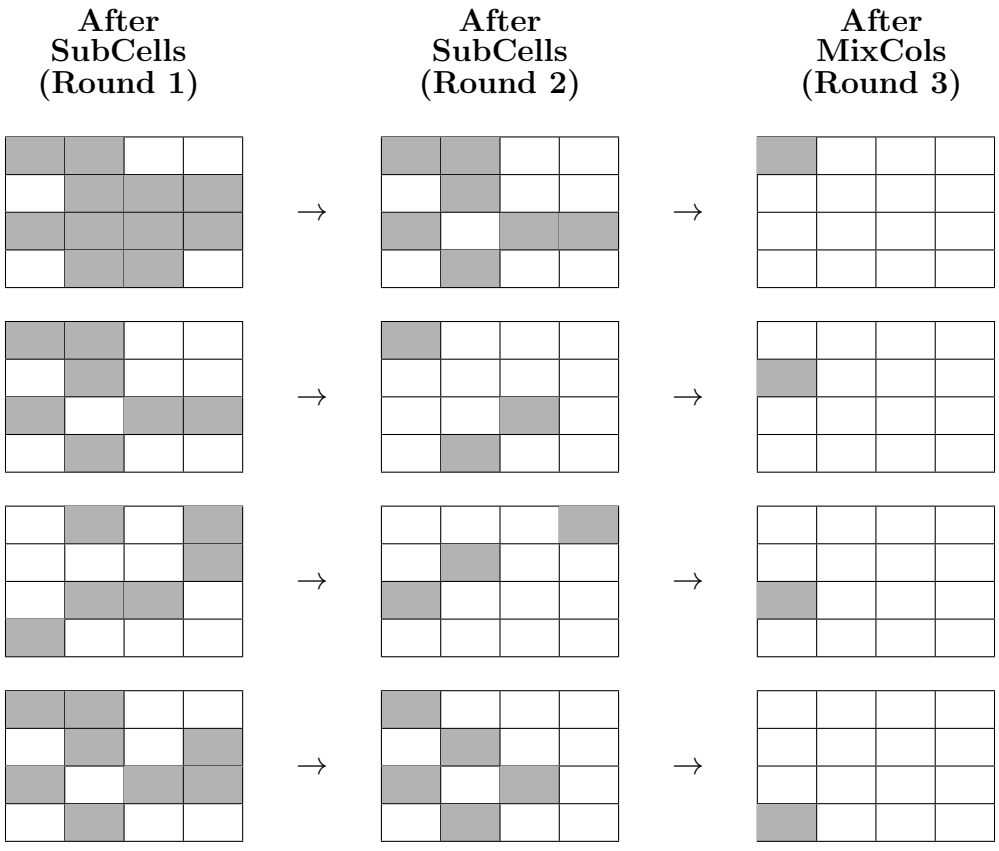


Figure B.1: The trails of Skinny two and three rounds for all four cases with an active cell at the end

Appendix C

Present: LAT and Trails

C.1 Linear Approximation Table

The rows indicate the input mask and the columns indicate the output masks.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	+8	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0
1	+0	+0	+0	+0	+0	-4	+0	-4	+0	+0	+0	+0	+0	-4	+0	+4
2	+0	+0	+2	+2	-2	-2	+0	+0	+2	-2	+0	+4	+0	+4	-2	+2
3	+0	+0	+2	+2	+2	-2	-4	+0	-2	+2	-4	+0	+0	+0	-2	-2
4	+0	+0	-2	+2	-2	-2	+0	+4	-2	-2	+0	-4	+0	+0	-2	+2
5	+0	+0	-2	+2	-2	+2	+0	+0	+2	+2	-4	+0	+4	+0	+2	+2
6	+0	+0	+0	-4	+0	+0	-4	+0	+0	-4	+0	+0	+4	+0	+0	+0
7	+0	+0	+0	+4	+4	+0	+0	+0	+0	-4	+0	+0	+0	+0	+4	+0
8	+0	+0	+2	-2	+0	+0	-2	+2	-2	+2	+0	+0	-2	+2	+4	+4
9	+0	+4	-2	-2	+0	+0	+2	-2	-2	-2	-4	+0	-2	+2	+0	+0
A	+0	+0	+4	+0	+2	+2	+2	-2	+0	+0	+0	-4	+2	+2	-2	+2
B	+0	-4	+0	+0	-2	-2	+2	-2	-4	+0	+0	+0	+2	+2	+2	-2
C	+0	+0	+0	+0	-2	-2	-2	-2	+4	+0	+0	-4	-2	+2	+2	-2
D	+0	+4	+4	+0	-2	-2	+2	+2	+0	+0	+0	+0	+2	-2	+2	-2
E	+0	+0	+2	+2	-4	+4	-2	-2	-2	-2	+0	+0	-2	-2	+0	+0
F	+0	+4	-2	+2	+0	+0	-2	-2	-2	+2	+4	+0	+2	+2	+0	+0

Figure C.1: Linear Approximation Table (LAT) for Present S-box

C.2 Linear Trails for Present

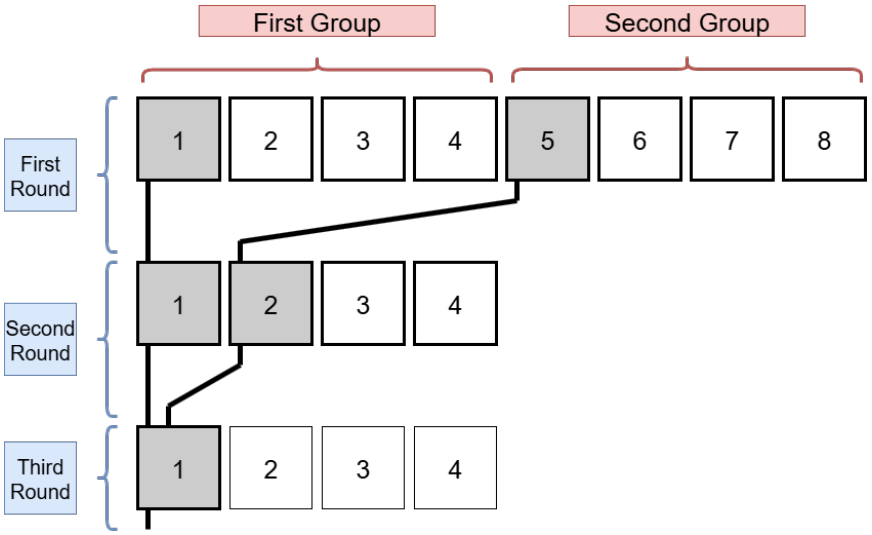


Figure C.2: Best trail for three rounds activating five S-boxes for the first non-uniform input in 4.3.1

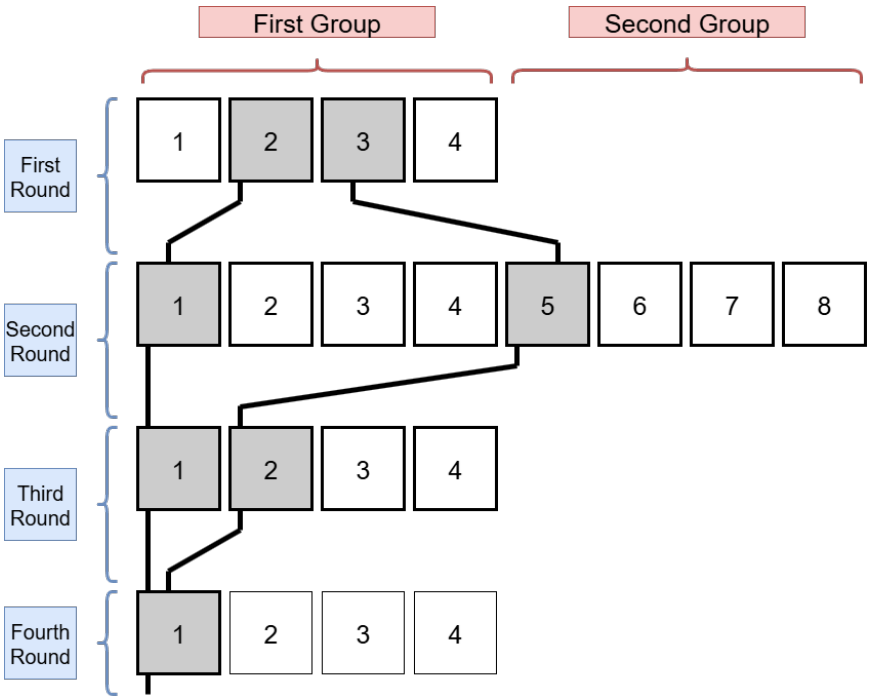


Figure C.3: Best trail for four rounds activating seven S-boxes for the second non-uniform input in 4.3.2

Appendix D

PROLEAD: Tables and Graphs

D.1 Tables from PROLEAD reports

D.1.1 Uniform Inputs

Table D.1: Evaluation Info: Uniform Masking

Cipher	#Standard Probes	#Extended Probes	Security Order	#Probing Set	Maximum #Probes per Set
Skinny	46662	70224	1	46266	25
Present	57630	89590	1	49130	14

Table D.2: Evaluation results: Skinny Uniform Masking

Elapsed Time	Required Ram	Processed Simulations	Probing Set with highest Information Leakage	-log_{10}(p)	Status
71.634446 s	7.969004 GB	102400	`mux_pt2[46] (4)`	3.795819	OKAY
144.651318 s	8.231148 GB	204800	`mux_pt1[13] (30)`	4.242760	OKAY
220.531922 s	8.231148 GB	307200	`mux_pt2[35] (1)`	3.802872	OKAY
.....					
33797.467970 s	8.231148 GB	50995200	`\round_Sbox_decomp_ from_Q294_ first_1[44] (9)`	3.508595	OKAY
33864.956402 s	8.231148 GB	51097600	`\round_Sbox_decomp_ from_Q294_ first_1[44] (9)`	3.474759	OKAY
33931.210566 s	8.231148 GB	51200000	`mux_pt3[1] (30)`	3.447474	OKAY

Table D.3: Evaluation results: Present Uniform Masking

Elapsed Time	Required RAM	Processed Simulations	Probing Set with Highest Information Leakage	-log_{10}(p)	Status
32.709491 s	10.292112 GB	102400	`ciphertext3[3] (34)`	3.567592	OKAY
64.794571 s	10.292112 GB	204800	`ciphertext2[55] (62)`	3.702537	OKAY
96.970094 s	10.292112 GB	307200	`ciphertext3[33] (16)`	4.123171	OKAY
.....					
30381.553168 s	10.292112 GB	102195200	`ciphertext3[48] (24)`	5.000785	LEAKAGE
30410.077457 s	10.292112 GB	102297600	`ciphertext3[48] (24)`	4.845446	OKAY
30438.949881 s	10.292112 GB	102400000	`ciphertext3[48] (24)`	4.853469	OKAY

D.1.2 Non-Uniform Inputs

Table D.4: Evaluation Info: Skinny Non-Uniform Inputs

Cipher	#Standard Probes	#Extended Probes	Security Order	#Probing Set	Maximum #Probes per Set
Skinny	50556	65802	1	41778	24
Present	57630	73610	1	41055	13

Table D.5: Evaluation results: Skinny (Insecure)

Elapsed Time	Required RAM	Processed Simulations	Probing Set with Highest Information Leakage	$-\log_{10}(p)$	Status
57.360254 s	7.686788 GB	102400	`\uut_mux_pt3[60] (4)`	inf	LEAKAGE
116.502428 s	7.752324 GB	204800	`\uut_mux_pt3[61] (4)`	inf	LEAKAGE
173.557319 s	7.817860 GB	307200	`\uut_mux_pt3[61] (4)`	inf	LEAKAGE
.....					
5526.666311 s	7.817860 GB	10035200	`\uut_mux_pt3[61] (4)`	inf	LEAKAGE
5578.387022 s	7.817860 GB	10137600	`\uut_mux_pt3[61] (4)`	inf	LEAKAGE
5629.935321 s	7.817860 GB	10240000	`\uut_mux_pt3[61] (4)`	inf	LEAKAGE

Table D.6: Evaluation results: Skinny (Secure)

Elapsed Time	Required RAM	Processed Simulations	Probing Set with Highest Information Leakage	$-\log_{10}(p)$	Status
56.212279 s	7.753684 GB	102400	\uut_mux_pt1[57] (45)	4.228449	OKAY
113.798281 s	7.819220 GB	204800	\uut_mux_pt3[24] (55)	4.390328	OKAY
175.130282 s	7.819220 GB	307200	\uut_mux_pt2[48] (17)	4.589605	OKAY
.....					
52623.688284 s	7.819220 GB	102195200	\uut_round_Sbox_decomp_from_Q294_first_2[28] (20)	3.769765	OKAY
52678.202404 s	7.819220 GB	102297600	\uut_round_Sbox_decomp_from_Q294_first_2[28] (20)	3.851361	OKAY
52729.345287 s	7.819220 GB	102400000	\uut_round_Sbox_decomp_from_Q294_first_2[28] (20)	3.931028	OKAY

Table D.7: Evaluation results: Present (First Non-Unifom Example)

Elapsed Time	Required RAM	Processed Simulations	Probing Set with Highest Information Leakage	$-\log_{10}(p)$	Status
25.079097 s	8.667848 GB	102400	\uut_round_func_Sbox_decomp_from_Q299_2[57] (6)	inf	LEAKAGE
50.413631 s	8.667848 GB	204800	\uut_round_func_Sbox_decomp_from_Q299_3[60] (6)	inf	LEAKAGE
75.268276 s	8.667848 GB	307200	\uut_round_func_Sbox_decomp_from_Q299_3[60] (6)	inf	LEAKAGE
.....					
25149.866701s s	8.667848 GB	102195200	\uut_round_func_Sbox_decomp_from_Q299_3[60] (6)	inf	LEAKAGE
25175.004107 s	8.667848 GB	102297600	\uut_round_func_Sbox_decomp_from_Q299_3[60] (6)	inf	LEAKAGE
25200.399231 s	8.667848 GB	102400000	\uut_round_func_Sbox_decomp_from_Q299_3[60] (6)	inf	LEAKAGE

Table D.8: Evaluation results: Present (Second Non-Uniform Example)

Elapsed Time	Required RAM	Processed Simulations	Probing Set with Highest Information Leakage	$-\log_{10}(p)$	Status
163.313425 s	8.472224 GB	102400	\uut_round_func_ Sbox_decomp_from_ Q299_2[57] (6)	inf	LEAKAGE
326.293271 s	8.603296 GB	204800	\uut_round_func_ Sbox_decomp_from_ Q299_3[60] (6)	inf	LEAKAGE
479.198426 s	8.668832 GB	307200	\uut_round_func_ Sbox_decomp_from_ Q299_3[60] (6)	inf	LEAKAGE
.....					
100734.146654 s	8.668832 GB	102195200	\uut_round_func_ Sbox_decomp_from_ Q299_3[60] (6)	inf	LEAKAGE
100822.035424 s	8.668832 GB	102297600	\uut_round_func_ Sbox_decomp_from_ Q299_3[60] (6)	inf	LEAKAGE
100911.087427 s	8.668832 GB	102400000	\uut_round_func_ Sbox_decomp_from_ Q299_3[60] (6)	inf	LEAKAGE

D.2 Graphs related to Present with first example non-uniform masking from Section 4.3.1

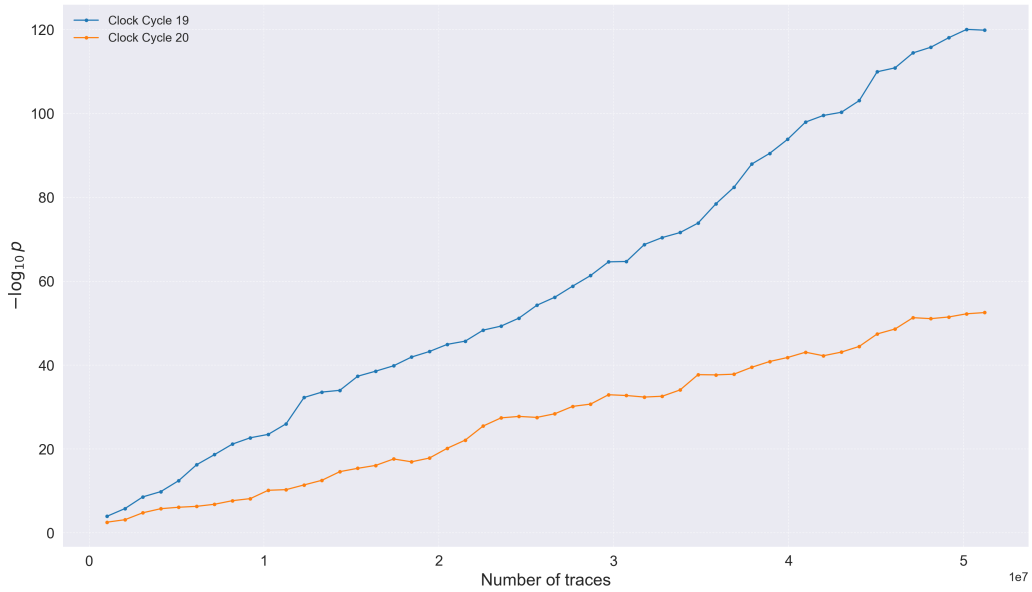


Figure D.1: Leakage progression in clock cycle 19 and 20

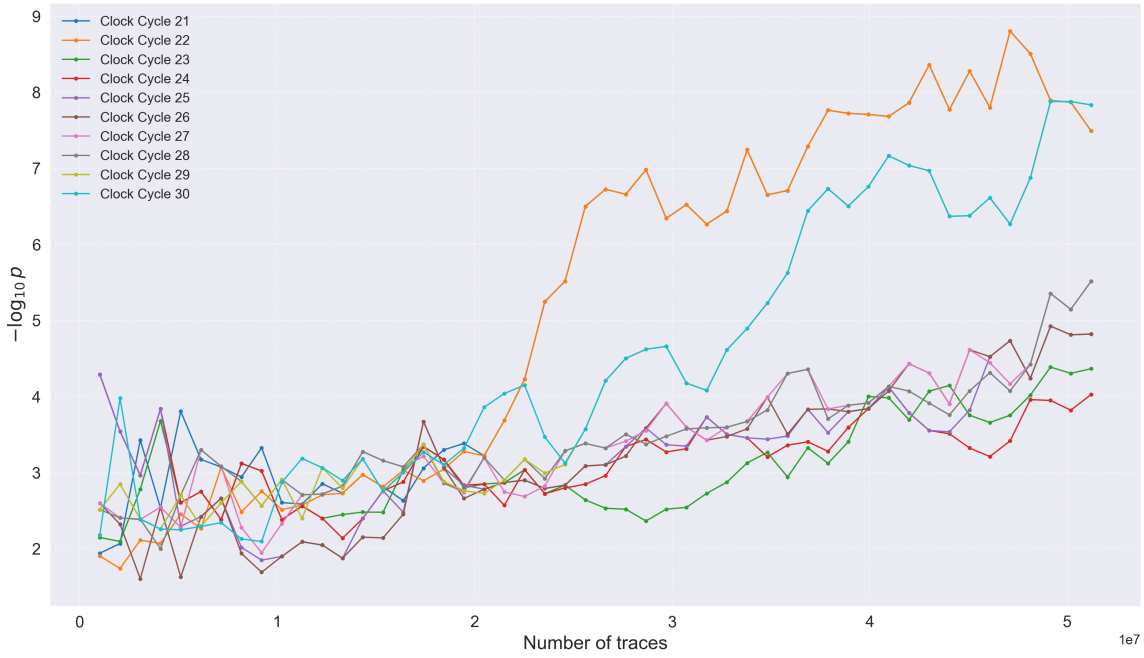


Figure D.2: Leakage progression in clock cycle 21 to 30

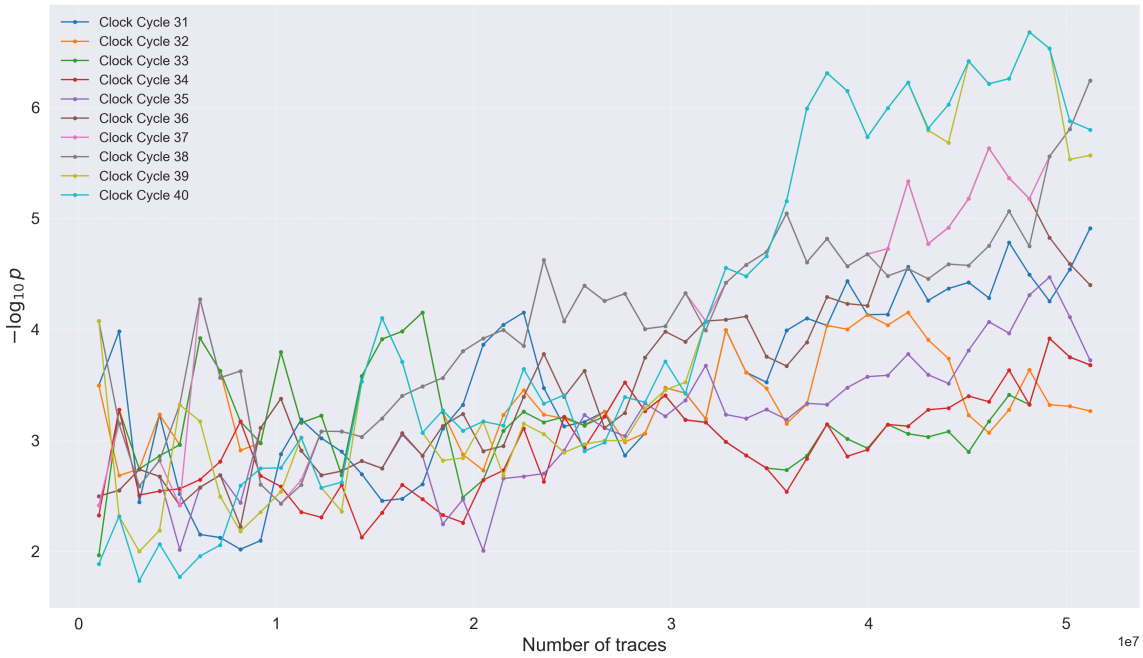


Figure D.3: Leakage progression in clock cycle 31 to 40

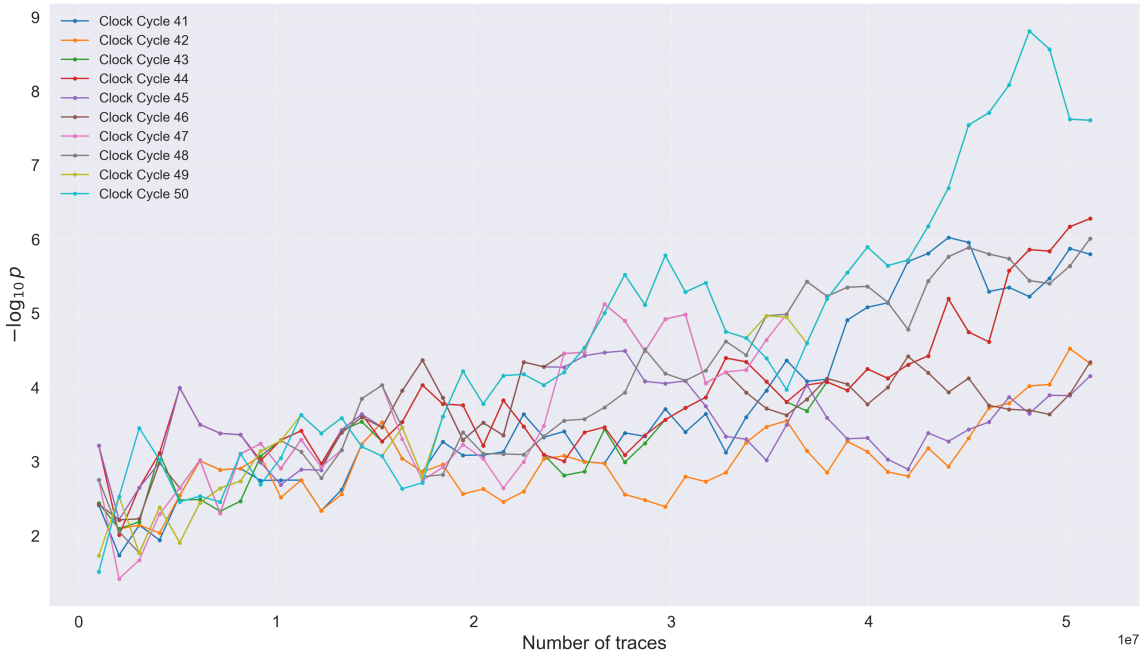


Figure D.4: Leakage progression in clock cycle 41 to 50

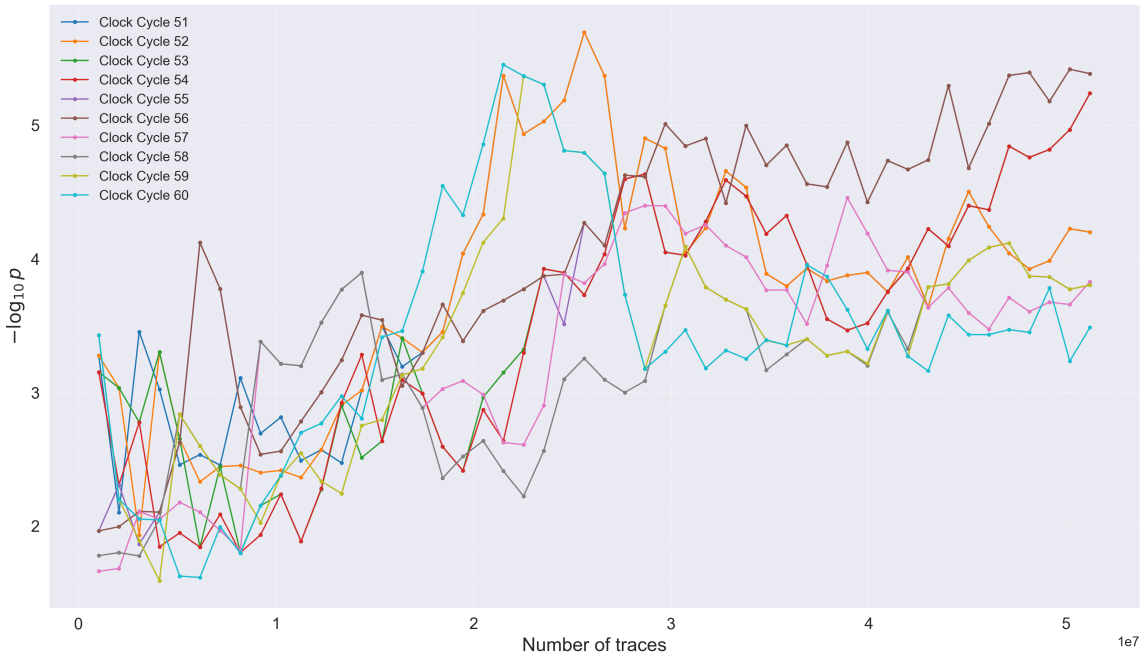


Figure D.5: Leakage progression in clock cycle 51 to 60

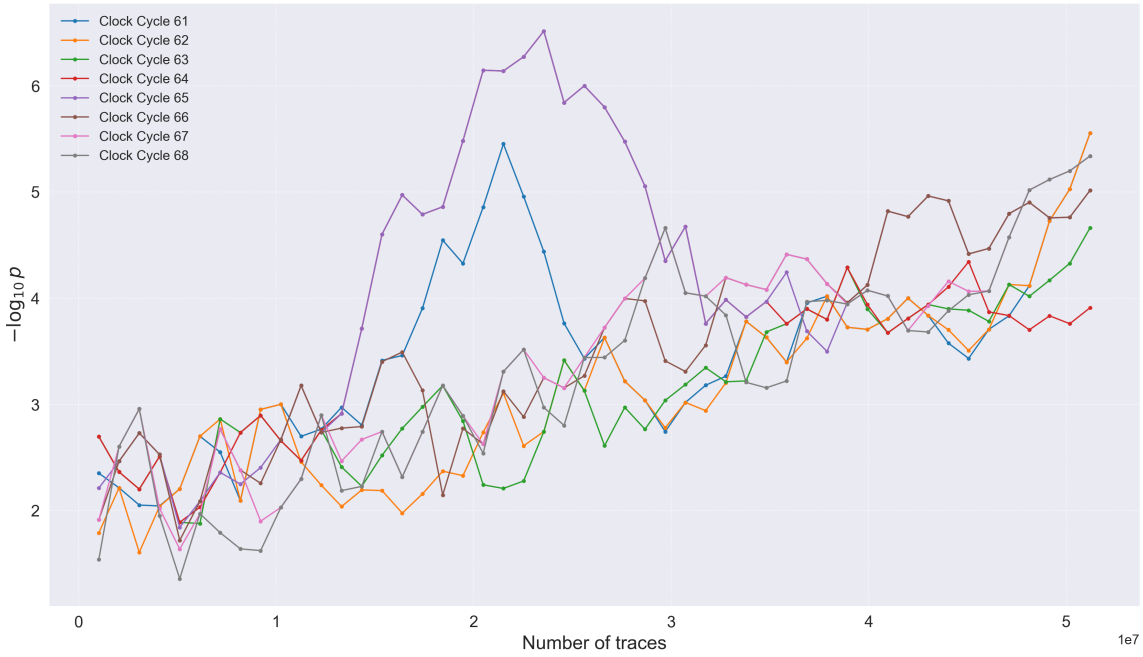


Figure D.6: Leakage progression in clock cycle 61 to 68