DEPARTMENT OF COMPUTER SCIENCE

An Introduction to Side channel attacks and Masking security

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Thursday 4th May, 2023

Abstract

This project aims to provide an introduction to side channel attacks and masking security. The reader is assumed to have an undergraduate knowledge of mathematics, specifically probability, linear algebra and formal proofs. Additionally the reader should be familiar with information theory for chapter 2 which makes heavy use of entropy and mutual information. Technically very little knowledge of cryptography is required to read this but is still recommended as it would serve as motivation for the goals. This project covers, in order, specific attacks, empirical security estimates and masking security. The proofs presented in the literature are often inaccessible to those unfamiliar with the field so frequently proofs are expanded upon or entirely new (simpler) proofs are presented. Additionally the results are presented with a uniform notation to make the overall understandability easier. An emphasis is placed on explaining why definitions are the way the are especially in section 4.4 composition. This is aimed to mitigate the inundation of definitions that can come in some learning materials.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

[redacted] Thursday 4th May, 2023

Contents

Ethics Statement

This project did not require ethical review, as determined by my supervisor, François Dupressoir.

Notation and Acronyms

r.v.	:	Random Variable
iid	:	Independent and identically distributed
$\mathbb{F}$	:	Denotes a Field, $\mathbb{F}_q$ is a finite field with q elements
$\mathcal{M}_{n\times m}(\mathbb{F})$	:	The set of matrices over the field $\mathbb{F}$ of shape $n \times m$
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	:	Calligraphic letters represent sets (infinite of finite)
$\lvert\mathcal{X}\rvert$	:	Denotes the cardinality of the set X (the number of elements in $\mathcal{X}$)
$\lvert x\rvert$	:	For $x \in \mathbb{R}$ $\lvert x\rvert$ is the absolute value of x.
$\lvert M\rvert$	:	For $M \in \mathcal{M}_{n\times n}(\mathbb{F})$ denotes the determinant of the matrix.
M^{-1}	:	For $M \in \mathcal{M}_{n\times n}(\mathbb{F})$ denotes the inverse of the matrix when it exits.
X, Y, Z	:	Capital letters represent r.v.s drawn from the set with the same name
x, y, z	:	Lowercase letters represent elements from sets e.g. $x \in \mathcal{X}$
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	:	Bold lowercase letter represent vectors of elements from sets e.g. $\mathbf{x} \in \mathcal{X}^n$ for $n \in \mathbb{N}$
$\mathbf{x}_i$	:	The i-th element of the vector $\mathbf{x}$
$p(x)$	:	Probability that the r.v. $X = x$
$p(x\lvert y)$	:	The probability that the r.v. $X = x$ given that $Y = y$
$p(x, y)$	:	The probability that $X = x$ and $Y = y$
$H(X)$	:	The entropy of X defined as $\sum_{x\in\mathcal{X}} p(x) \log_2(\frac{1}{p(x)})$
$\mathrm{H}(X\lvert Y)$	:	The conditional entropy of X given Y defined as $\sum_{y\in\mathcal{Y}} p(y) \sum_{x\in\mathcal{X}} p(x\lvert y) \log_2(\frac{1}{p(x\lvert y)})$
$\mathrm{MI}(X;Y)$	:	The mutual information between X and Y defined as $\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log_2(\frac{p(x,y)}{p(x)p(y)})$
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	:	Vectors of iid random variables each distributed as X
$\mathbf{X}_i$	:	The i-th r.v. in the random vector $\mathbf{X}$
$\mathbf{X}_i^j$	:	The slice of the random vector from index i to j inclusive
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	:	Vectors of elements e.g. $\mathbf{x} \in \mathcal{X}^n$
$\mathrm{H}(\mathbf{X})$	:	The joint entropy of the random vector
$\mathbb{E}(X)$	:	The expectation of the r.v. X
$\mathbb{F}_q$	:	a finite field with q elements
AES	:	Advanced Encryption Standard
$x \Vert y$	:	Concatenation of binary strings
$d_H(x)$	:	the Hamming weight of x
$x \leftarrow f()$	:	The variable x gets assigned the result of the function f
$x \xleftarrow{\$} f()$	:	The variable x gets assigned the result of the randomized function f
$x \xleftarrow{\$} \mathcal{X}$	:	The variable x gets assigned a value in $\mathcal{X}$ uniformly at random
$[a, b]$	:	For $a \leq b$ is the set $\{x \in \mathbb{R} : a \leq x \leq b\}$
$[\![a, b]\!]$	:	For $a \leq b$ is the set $[a, b] \cap \mathbb{Z}$
$a \oplus b$	:	For a, b bit strings denotes the bitwise xor between the two
MLP	:	Multi Layer Perceptron

Chapter 1

Introduction

Side channel attacks are a powerful form of attack that adversaries can make use of to break cryptographic implementations. A side channel is any way of observing information of an encryption other than the expected message and ciphertext considered in standard models of security. When considering security of an algorithm this is normally done in the black box model, an adversary gets to encrypt some number of cipher texts and observe the messages, then they make a guess for the secret key. When considering side channel security the adversary can do the same but they also receive the side channel data for every encryption. Some examples of side channels are execution time, power usage or EM emission. The time an algorithm takes to run can be used to, for example, find Diffie-Hellman exponents and factor RSA keys as demonstrated in [Koc96]. Fortunately timing attacks are relatively simple to defeat. By simply inserting dummy operations the algorithm can be made to run in constant time thus preventing timing analysis. A more powerful and difficult to prevent form of attack are power analysis attacks. These attacks measure the power a computer uses as it encrypts some data and from this information they can, for example, recover the secret key used in AES encryption [Ors+04]. It may seem implausible that an adversary could measure the power usage however there are real and important situations that do fit in to this category. A smart card using some cryptographic algorithm to encrypt data could easily have it's power usage monitored if it was either stolen or inserted into a malicious reader. IOT devices can't reliably trust their power source but must still perform encryption to talk over the web. The next chapter will demonstrate how to make use of the power usage to recover sensitive data.

Chapter 2

Attacks

In this chapter we will play the role of an adversary, attacking some implementation. There are two forms of attacks we could use, simple and profiled. In simple attacks there are two main stages, first observe the hardware encrypting some number of messages and measure the power usage while the encryption is being performed. Then perform statistical analysis on this collected data set to make a guess of the secret key. The type of the power usage isn't simply the total power being used over the whole encryption as this typically wouldn't contain enough information to yield a successful attack. In practice the instantaneous power usage is sampled throughout the encryption process to obtain a trace of power vs time for each message encryption that forms the data set. In profiled attacks the adversary also has access to an identical version of the device being attacked they can set the secret key for. We will discuss profiled attacks later but first an example of a simple attack is detailed.

2.1 Correlation Power analysis

.

One type of simple attack is correlation power analysis (CPA) The explanation presented is informed by [BCO04]. CPA works by breaking the key into chunks of enumerable size and assuming that the trace data is correlated with some function of the chunk. Then by computing how correlated that function is with the power usage and selecting the key that has the highest correlation, we obtain a guess for the each key byte.

To formalize this notion we will walk through a correlation analysis attack on AES 128 step by step. By inspecting the specification for AES [ST] an intuitive choice for what each chunk should be presents itself. AES organizes it's 128 bit key into a 4×4 matrix where each entry is one byte. We will denote the matrix as rk_{ij} where $0 \leq i, j \leq 3$ and each $r_{ij} \in$ is thought of as a field element in the finite field of size 2^8. We will use the elements of the matrix as the chunks to enumerate. By looking more at the specification see that the first thing AES does is xor each key bit with each message bit. We assume that the power usage when performing this xor is dependent on the number of bits that change state (from 0 to 1 or 1 to 0). We can express this is as the Hamming distance between the key byte k and the message byte m or equivalently the Hamming weight of $k \oplus m$. Power $\approx A \cdot d_H(k, m) + B$ where $A, B \in \mathbb{R}$ are some unknown constants and the actual power being recorded is perturbed by some form of noise. The hamming weight is chosen as a heuristic because it has been shown to correlate with the leakage of devices in [MD99] and [KJJ99].

Collecting data set: We need some way of interacting with the device as well as some method for measuring the power being drawn. Measuring the voltage across a resistor in series with the power pin of the cpu will allow us to collect the power usage [KJJ99]. Oscilloscopes that can do this at high frequencies can be purchased online for around £100 [Pic]. Assuming that there exists a way to request arbitrary encryptions from the device under attack we proceed as follows. Let $\mathcal{M}$ be the set of messages, $\mathcal{C}$ be the set of ciphertexts and $\mathcal{L}$ be the set of leakage data (i.e. the power usage at each time sample). Let Interact be a function that takes a message, gives it to the device, monitors the power and returns both the ciphertext and leakage, Interact : $\mathcal{M} \to \mathcal{C} \times \mathcal{L}$ simply iterate from 1 to n and call Interact with a uniformly random message m and collect the messages, ciphertexts and leakages obtained. Call the

list of messages $\mathbf{m} \in \mathcal{M}^n$, the list of ciphertexts $\mathbf{c} \in \mathcal{C}^n$ and the list of leakages $\mathbf{l} \in \mathcal{L}^n$ where each $\mathbf{l}$ is a vector of t elements with $\mathbf{l}^i$ being the power usage at the i-th sample time.

Analysing data set: Now the data set is collected we need to analyze it. We will use Pearson's correlation coefficient independently on each point in time to compute the correlation between the hamming weight and power. This produces a graph of correlation vs time for each possible value of a key byte guess. The key byte guess with the highest peak is selected as the final guess.

Definition 1. ***(Pearson's Correlation Coefficient).*** *Given a paired data set* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ *Pearson's Correlation Coefficient is defined as:*

$$PCC(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where $\bar{x} = \sum_{i=1}^{n} x_i$ *and likewise for* y.

Pearson's Correlation Coefficient is 1 if $(\mathbf{x}_i, \mathbf{y}_i)$ all lie on a straight line with positive gradient when plotted as points on the 2d plane, -1 if they all lie on a straight line with negative gradient and 0 if there is no correlation. We don't care if the power is positively or negatively correlated with the hamming weight so we will actually use the absolute value of PCC.

Algorithm 2.1: Correlation power analysis attack

Input: vector of messages $\mathbf{m}$, vector of ciphertexts $\mathbf{c}$, vector of leakages $\mathbf{l}$
Output: Guess k of secret key

```
// For each element in the round key matrix
for r = 1 to 16 do
    // For each possible value of this element
    for g = 0 to 255 do
        // For each time the power was sampled
        for t = 0 to t do
            // Build paired data set
            for i = 0 to n do
                x_i ← d_H(g ⊕ m_i)
                y_i ← l_i^t
            end
            r_g^t = |PCC(x, y)|
        end
        r = max_{t'}(r_g^{t'})
    end
    k = arg max_{g'}(r_{g'})
end
return k = k_0 || ... || k_16
```

There is still one slight problem with this, the correct key byte k and its negation $\neg k$ will correlate exactly the same amount with the data set, so even when the correct guess has the highest correlation an incorrect guess of $\neg k$ will sometimes be found instead. One simple fix is to test all 2^{16} possible combinations of k and $\neg k$, By encrypting a known message ciphertext pair and seeing if the message encrypts to the correct ciphertext. It is also extremely important to make sure that each trace is synchronised in time with all others. This is because the correlation for each time sample is computed independently from all others. This limitation can be mitigated somewhat by down-sampling the data.

On implementing a correlation power analysis attack against an unprotected AES 128 implementation I found that the secret key could be recovered with 1000 traces in around 5 minutes (This was performed as part of COMS30048).

So when the power is correlated with a known function we have a way of using this knowledge to reduce the search space of keys to a manageable size. The hamming weight is a fairly standard choice but there is no guarantee that the power will actually be correlated with this model. In the next section an attack that doesn't have this limitation is introduced.

2.2 Template Attacks

Template attacks (TA) are in a more powerful class of attacks, called profiled attacks. Recall in this scenario we are attacking a device $\mathcal{D}$ through side channel attacks like before, but now we have our own programmable version of $\mathcal{D}$ that we can set the secret key and record traces from. We will call this programmable version $\mathcal{D}'$. In correlation power attacks the power was assumed to be some function of the hamming weight, in a profiled attack $\mathcal{D}'$ will be used to build a statistical model of how the power is actually distributed. This allows template attacks to work against a wider variety of targets. A template is constructed for each possible key and then the templates are used to select the key that has the highest probability of producing the trace data observed from $\mathcal{D}$. The are endless ways the power usage could be modeled, we will detail a Gaussian model which was proposed in [CRR03] which is therefore called a Gaussian Template attack (gTA). Any other statistical model could be used with very little change, but Gaussian templates are detailed due to their popularity and evidence of efficacy. Formally a device $\mathcal{D}$ can be thought of as being in one of $|\mathcal{K}|$ states $S_0, \ldots, S_{|\mathcal{K}|-1}$, which each have an associated random leakage vector $\mathbf{L}_0, \ldots, \mathbf{L}_{|\mathcal{K}|-1}$ each distributed to a (hopefully) unique function. This is where the model is used, assume each $\mathbf{L}_k$ is distributed according to a Multivariate Gaussian function. Each Gaussian can be estimated by calculating the mean and covariance of a data-set collected from $\mathcal{D}'$ for each k.

Definition 2. ***(Multivariate Gaussian Distribution).*** *Let $\mu \in \mathbb{R}^k$ and $\Sigma \in \mathbb{R}^{k\times k}$ be a positive semi-definite matrix. Then for $\mathbf{X} = (X_1, \ldots, X_k)$ we say $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$ (read as $\mathbf{X}$ is distributed Normally or Gaussian with location μ and covariance Σ). The probability distribution function of $\mathbf{X}$ for $\mathbf{x} \in \mathbb{R}^k$ is:*

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \cdot \exp\Big(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\Big)$$

The parameters (μ, Σ) are called a template. Building a template for each possible value of the key is clearly infeasible (this is the same complexity as brute forcing the encryption). We could split the key into some number of separate subkeys just as we did for correlation analysis and build a set of templates for each subkey. Splitting a 128 bit key into 16 subkeys and building a template for each reduces the number of templates from 2^{128} down to $16 \cdot 2^8 = 2^{12}$ which is manageable, we can however do something smarter. We make use of an extend and prune strategy and dynamically building templates as detailed in [CRR03]. By keeping a set of the most likely partial keys and extending it we will have a higher probability of successfully finding the correct key.

Below is pseudo code for a template attack. Let k the secret key $\mathcal{D}$ is using be λ bits in length. First an algorithm for building a template of the first bits of the key from $\mathcal{D}'$ and then and algorithm to attack $\mathcal{D}$.

Algorithm 2.2: Template

Input: $\hat{k}$ bit slice of $n \leq \lambda$ bits, the first n bits of a guess of k
Input: $m \in \mathbb{N}$ the number of traces to take, typically 1000
Output: (μ, Σ) template parameters for $\hat{k}$
for $i = 1$ **to** m **do**
  `// Take padding uniformly at random`
  pad $\xleftarrow{\$} \{0, \ldots, 2^{\lambda-n}\}$
  `// Pad the key to` λ `bits`
  $\hat{k}' \leftarrow \hat{k} \| pad$
  `// Take a message uniformly at random`
  $m \xleftarrow{\$} \mathcal{M}$
  `// Get a leakage vector from` $\mathcal{D}'$ `encrypting` m `with key` $\hat{k}'$`, ignore ciphertext`
  $\_, \mathbf{l}_i \xleftarrow{\$} \mathcal{D}_{\hat{k}'}(m)$
end
`//` μ `is the mean of the leakages`
$\mu \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathbf{l}_i$
`//` Σ `is the covariance of the leakages`
$\Sigma_{ij} \leftarrow \frac{1}{m-1}\sum_{a=1}^{m}(\mathbf{l}_{ai} - \mu_i)(\mathbf{l}_{aj} - \mu_j)$
return (μ, Σ)

Algorithm 2.3: Gaussian Template Attack

Input: Set of traces collected from $\mathcal{D}$, $\mathcal{T}$ of size n
Input: $b \in \mathbb{N}$ the extension length, on each step the key guess will be extended by b bits
Output: Set of guesses $\mathcal{H}$ of secret key k

```
// The hypotheses for the first bits of the key and their templates
```
$\mathcal{H} \leftarrow \emptyset$
```
// Extend until all bits of k have been covered
```
for $r = 0$ **to** $\frac{\lambda}{b}$ **do**
```
    // Stores the extended keys from H
```
    $\mathcal{H}' \leftarrow \emptyset$ `// Extend the current hypotheses`
    if $\mathcal{H} \neq \emptyset$ **then**
```
        // For each current hypothesis
```
        for $(\hat{k}_{r-1}, \_, \_) \in \mathcal{H}$ **do**
```
            // For each possible value of the next bits
```
            for $\hat{k}_r = 0$ **to** 2^b **do**
```
                // Construct a new hypothesis
```
                $\hat{k}'_r \leftarrow \hat{k}_{r-1} \| \hat{k}_r$
```
                // Create a template for this hypothesis
```
                $(\mu'_{\hat{k}_r}, \Sigma'_{\hat{k}_r}) \leftarrow \text{Template}(\hat{k}'_r)$
```
                // Add it to the set of hypotheses
```
                $\mathcal{H}' \leftarrow \mathcal{H}' \cup \{(\hat{k}'_r, \mu'_{\hat{k}_r}, \Sigma'_{\hat{k}_r})\}$
            end
        end
    else
```
        // On the first round there are no hypotheses to extend
```
        for $\hat{k}_r = 0$ **to** 2^b **do**
            $(\mu_{\hat{k}_r}, \Sigma_{\hat{k}_r}) \leftarrow \text{Template}(\hat{k}_r)$
            $\mathcal{H}' \leftarrow \mathcal{H}' \cup \{(\hat{k}_r, \mu_{\hat{k}_r}, \Sigma_{\hat{k}_r})\}$
        end
    end
```
    // Remove unlikely hypotheses to limit complexity
```
    $\mathcal{H} = \text{Prune}(\mathcal{H}', \mathcal{T})$
end
return $\mathcal{H}$

Pruning can work in several ways:

- Take the most likely n hypothesis
- Let some factor $c \in [0, 1]$ and take hypotheses that are greater than $c \cdot P_{\max}$ where $P_{\max}$ is the max over the probabilities in the hypothesis set.
- Scale the probabilities so that they sum to 1, then take hypotheses until the sum of their probabilities is greater than some threshold $c \in [0, 1]$.

The hypotheses in $\mathcal{H}$ can then be tested by encrypting a message with a known ciphertext. It is also important to point out a useful optimization, the leakage vectors $\mathbf{l}$ typically have many components. This can make calculating the covariance a bottleneck for speed. To work around this as suggested in [CRR03], we can down sample the leakage vectors to only include times where there is significant difference between the distributions. To do this first compute the means of the leakages for each possible extension. Then compute pairwise which points have the largest differences between the means. Adding up the pairwise differences and selecting the points that have the most significance. This gives a way to reducing the dimensionality down to a specified size while retaining most information in the trace.

Chapter 3

Information Bounds

3.1 Mutual Information as a security metric

In the previous chapter we have seen how easy it is for the adversary. Now we turn our attention to the task of trying to defend cryptographic algorithms. For this chapter we will be playing the role of an engineer who has just implemented some form of encryption and wants to know how secure said implementation is against side channel attacks. With our newfound knowledge of how to break implementations one approach could be to simply attack our program with various types of attacks and see how many traces are required, however this only provides information about specific adversaries and not against general adversaries. This is important because real adversaries could be performing one of many different attacks and we don't want to have to implement every form of attack possible and then test against each one. What we want is something that provides guarantees against all adversaries. To do this we will consider the optimal adversary who extracts the full amount of information available from every trace.

For this chapter $\mathcal{K}$ will denote the set of possible values of the variable the adversary is attempting to recover. As we saw in the previous chapter when attacking it is typical to target separate parts of the key variable separately. So while the space of keys for AES 128 is $\{0, 1, \dots 2^{128} - 1\}$ since the adversary is targeting byte chunks we would set $\mathcal{K} = \{0, 1, \dots, 2^8 - 1\}$.

Also $\mathcal{L}$ will denote the set of possible leakage values. This will typically look like $\mathcal{L} = \{0, 1, \dots, 2^{\omega}-1\}^D$ where there are D measurements over time, each taking one of 2^{ω} possible values, therefore we have $|\mathcal{L}| = 2^{\omega D}$.

Theorem 1. *Let K be the r.v. representing the key and L is the r.v. representing the leakage then this adversary extracts $MI(K; L)$ bits from every trace. Then the number of traces t required to fully recover the key is given as:*

$$t \geq \frac{H(K)}{MI(K; L)}$$

Additionally if the adversary is willing to perform λ bits of work, i.e. checking all values in the range 2^{λ} then the requirement on the entropy is now $H(K|L_1^n) \leq \lambda$ and the bound is now:

$$t \geq \frac{H(K) - \lambda}{MI(K; L)}$$

Proof. To show this let L_1^t be the t traces acquired, the key is known when the entropy of the key given the traces is less than 0. i.e.

$$\begin{aligned} 0 &\geq \mathrm{H}(K|L_1^t) \\ 0 &\geq \mathrm{H}(K) - \mathrm{MI}(K; L_1^t) \\ 0 &\geq \mathrm{H}(K) - t \cdot \mathrm{MI}(K; L) \\ t &\geq \frac{\mathrm{H}(K)}{\mathrm{MI}(K; L)} \end{aligned}$$

To see that $\mathrm{MI}(K; L_1^t) = t \cdot \mathrm{MI}(K; L)$ first write $\mathrm{MI}(K; L_1^t) = \mathrm{H}(L_1^t) - \mathrm{H}(L_1^t|K)$ then:

$$\begin{aligned}
\mathrm{H}(L_1^t) &= \mathrm{H}(L_1) + \sum_{i=1}^{t} \mathrm{H}(L_i|L_1^{i-1}) && \text{(Chain rule for entropy)} \\
&= H(L_1) + \sum_{i=2}^{t} \mathrm{H}(L_i) && \text{(Independence of leakages)} \\
&= t \cdot \mathrm{H}(L) && \text{(Identical distribution of leakages)}
\end{aligned}$$

$$\begin{aligned}
\mathrm{H}(L_1^t) &= \mathrm{H}(K, L_1, \dots L_t) - \mathrm{H}(K) \\
&= \mathrm{H}(K) + \mathrm{H}(L_1|K) + \sum_{i=2}^{t} (\mathrm{H}(L_i|L_1^{i-1}, K)) - \mathrm{H}(K) && \text{(Chain rule for entropy)} \\
&= \mathrm{H}(L_1|K) + \sum_{i=2}^{t} \mathrm{H}(L_i|K) && \text{(Independence of leakages)} \\
&= t \cdot \mathrm{H}(L|K) && \text{(Identical distribution of leakages)}
\end{aligned}$$

$$\implies \mathrm{MI}(K; L_1^t) = \mathrm{H}(L_1^t) - \mathrm{H}(L_1^t|K) = t \cdot (\mathrm{H}(L) - \mathrm{H}(L|K)) = t \cdot \mathrm{MI}(K; L)$$

Which concludes the proof. □

This isn't the only relation between the Mutual information and adversaries, in [Che+19] and [DFS15] the Mutual information is converted into bounds on the success rate of an adversary. This bound on traces is simpler so was presented but the results of the next section for estimating Mutual information can still be applicable to the bounds success rate bounds.

3.2 Estimating Mutual Information

We now have the task of calculating the $\mathrm{MI}(K; L)$, unfortunately calculating it explicitly is not possible as it would require knowledge of the true leakage distribution. This in turn would require modeling the physical properties of the circuits and simulating the current flow. Simulating the current flow would require the physical position of wires and logic gates to be decided which is a step after where we want to normally consider. Therefore in this section we will look at estimating the MI through statistical methods, following the work of [Bro+19] and [Mas+22]. We will start by looking at the equation for mutual information:

$$\mathrm{MI}(K; L) = \mathrm{H}(K) - \mathrm{H}(K|L) = \mathrm{H}(K) - \sum_{k \in \mathcal{K}} p(k) \sum_{l \in \mathcal{L}} p(l|k) \log_2\left(\frac{1}{p(k|l)}\right)$$

If the key is chosen uniformly at random we know the values of $\mathrm{H}(K)$ as well as $p(k) = \frac{1}{|\mathcal{K}|}$ for all k, however we still have no way to calculate $p(l|k)$ or $p(k|l)$. One thing we can do though is sample from the distribution $p(l|k)$ simply by setting our implementation to use the key k and recording the leakage. By doing this repeatedly for every key and equal number of times we can build a set $\mathcal{M}$ of observations where each $(k, l) \in \mathcal{M}$ is a key and the leakage that was observed on encrypting a uniformly random message with that key. From this set we can define the empirical distributions $e_n(k|l)$ and $e_n(l|k)$ that approximate $p(k|l)$ and $p(l|k)$ respectively as:

Definition 3. ***(Empirical Distribution).*** *Let $\mathcal{M}$ be a set of observations of tuples of keys and leakages (k, l) sampled from a distribution $p(l|k)$ by taken k equally across the set $\mathcal{K}$. The empirical distributions $e_n(k|l)$ and $e_n(l|k)$ are defined as:*

$$e_n(l|k) := \frac{|\{k^*, l^* \in \mathcal{M} : l^* = l \wedge k^* = k\}|}{|\{k^*, \_ \in \mathcal{M} : k^* = k\}|}$$
$$e_n(k|l) := \frac{|\{k^*, l^* \in \mathcal{M} : l^* = l \wedge k^* = k\}|}{|\{\_, l^* \in \mathcal{M} : l^* = l\}|}$$

By the strong law of large numbers the distributions functions $e_n(l|k)$ and $e_n(k|l)$ converge to $p(l|k)$ and $p(k|l)$ as $n \to \infty$. Because of this we can hope that simply by substituting $e_n(l|k)$ for $p(l|k)$ and $e_n(k|l)$ for $p(k|l)$ in the formula for MI we obtain a quantity that converges to the true MI. This is indeed the case and to show it however requires a bit of work to prove. The formula derived from this intuition is called the empirical hypothetical information (eHI).

Definition 4. ***(Empirical Hypothetical Information).*** *([Bro+19] equation 19) For a probability distrubtion p and the empirical distribution e_n defined from p the empirical hypothetical information is defined as:*

$$eHI_n(K; L) = H(K) - \sum_k p(k) \sum_l e_n(l|k) \log_2\Big(\frac{1}{e_n(k|l)}\Big)$$

We can also formulate this formula in a more efficient to calculate form. By setting $l_k(i)$ as the i-th leakage observed under the key k and let n_k be the number of leakages observed under key k we have.

$$\mathrm{eHI}_n(K; L) = \mathrm{H}(K) - \sum_k p(k) \frac{1}{n_k} \sum_{i=1}^{n_k} \log_2\Big(\frac{1}{e_n(k|l_k(i))}\Big)$$

This follows by expanding the definition of $e_n(l|k)$ in the sum and noticing that only values of l observed will have non-zero probability assigned to them by $e_n(l|k)$. This now means we can compute the eHI linearly in rate of the number of traces acquired, much better than of the order of $|\mathcal{L}|$.

Before the proof of convergence for the eHI is we dedicate a section to the mathematical background required. The next section isn't required to understand the results obtained, only the proofs behind the results. It is mostly a restatement from [Bro+19] with more elaboration for some proofs.

3.3 Technical Lemmas

Lemma 1. *([Bro+19] lemma 1) Let $\mathcal{M}$ be an ordered set of leakages obtained from key k of size n. With l_i denoting the i-th leakage and let e_n^i be the empirical distribution obtained from the set of observations without the i-th trace then:*

$$\frac{1}{n} \sum_{i=1}^{n} e_n^i = e_n$$

Proof. Let $l \in \mathcal{L}$ then:

$$\left(\frac{1}{n} \sum_{i=1}^{n} e_n^i\right)(l|k) = \frac{1}{n} \sum i = 1^n e_n^j(l|k) \tag{3.1}$$
$$= \frac{1}{n} \sum_{i=1}^{n} \frac{|\{l^* \in \mathcal{M} \setminus l_i | l^* = l\}|}{n-1} \tag{3.2}$$
$$= \frac{1}{n(n-1)} \sum_{i=1}^{n} |\{l^* \in \mathcal{M} \setminus l_i | l^* = l\}| \tag{3.3}$$
$$= \frac{1}{n(n-1)} \cdot (n-1) |\{l^* \in \mathcal{M} | l^* = l\}| \tag{3.4}$$
$$= \frac{1}{n} \cdot |\{l^* \in \mathcal{M} | l^* = l\}| \tag{3.5}$$
$$= e_n(l|k) \tag{3.6}$$
$$\tag{3.7}$$

step from (2.3) to (2.4) follows from the observation that for every index j such that $l_j = l$ will be counted $(n-1)$ times (every-time except when $i = j$ and so the sum is $(n-1)$ times greater than the number of times l actually was observed.

The same logic can be used to show that:

$$\frac{1}{n}\sum_{i=1}^{n} e_n^i(k|l) = e_n(k|l)$$

□

Lemma 2. *([Bro+19] lemma 2) Let $\gamma : [0,1]^t \to \mathbb{R}$ be a convex function, and p be a probability distribution with e_n the empirical distribution created by sampling from p Then for any $n > 1$, we have:*

$$\gamma(p) \leq \mathbb{E}(\gamma(e_n)) \leq \mathbb{E}(\gamma(e_{n-1}))$$

Moreover, if γ is continuous at p and bounded from above on $[0,1]^t$ then:

$$\mathbb{E}(\gamma(e_n)) \to \gamma(p)$$

monotonically with n.

Proof. First from Jensen's inequality if X is a r.v. that takes values in the domain of γ then $\gamma(\mathbb{E}(X)) \leq \mathbb{E}(\gamma(X))$. Also notice that $p = \mathbb{E}(e_n)$ so by direct application of Jensen's inequality we have:

$$\gamma(p) = \gamma(\mathbb{E}(e_n)) \leq \mathbb{E}(\gamma(e_n))$$

Then from Lemma 1 and the convexity of gamma we have that:

$$e_n = \frac{1}{n}\sum_{i=1}^{n} e_n^i \implies \gamma(e_n) = \gamma(\frac{1}{n}\sum_{i=1}^{n} e_n^i) \leq \frac{1}{n}\sum_{i=1}^{n}\gamma(e_n^i)$$

Which can be formally shown by induction on n and the definition of convexity for γ. Moreover since $\mathbb{E}(e_n^j) = \mathbb{E}(e_n^i) = \mathbb{E}(e_{n-1})$ for all i, j we have:

$$\begin{aligned}
\mathbb{E}(\gamma(e_n)) &\leq \mathbb{E}(\frac{1}{n}\sum_{i=1}^{n}\gamma(e_n^i)) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\mathbb{E}(\gamma(e_n^i)) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\mathbb{E}(\gamma(e_{n-1})) \\
&= \mathbb{E}(\gamma(e_{n-1}))
\end{aligned}$$

Now we show convergence of $\gamma(e_n)$ to $\gamma(p)$ under the assumption that γ is continuous at p and uniformly bounded by some M. By continuity of γ at p, for every $\epsilon > 0$ there is a $\delta > 0$ such that $\|e_n - p\| \leq \delta$ implies $|\gamma(e_n) - \gamma(p)| \leq \epsilon$. Also e_n converges in probability to p, meaning that for every (δ, ϵ') there is an N such that $Pr(\|e_n - p\| > \delta) < \epsilon'$ for any $n > N$. As a consequence for $n > N$ we have:

$$Pr(|\gamma(e_n) - \gamma(p)| > \epsilon) < \epsilon'$$

.

Let E be the set of possible values of e_n and $R = \{x \in \mathbb{R}^t : \|x - p\| \leq \delta\} \cup E$. Then for every $n > N$:

$$\begin{aligned}
\mathbb{E}(\gamma(e_n)) - \gamma(p) &= \mathbb{E}(\gamma(e_n) - \gamma(p)) \\
&= \sum_{x \in E} Pr[e_n = x] \cdot |\gamma(x) - \gamma(p)| \\
&= \sum_{x \in R} Pr[e_n = x] \cdot |\gamma(x) - \gamma(p)| + \sum_{x \in E \setminus R} Pr[e_n = x] \cdot |\gamma(x) - \gamma(p)| \\
&\leq \sum_{x \in R} Pr[e_n = x] \cdot \epsilon + \sum_{x \in E \setminus R} Pr[e_n = x] \cdot (M - \gamma(p)| \\
&= \epsilon \sum_{x \in R} Pr[e_n = x] + (M - \gamma(p)) \sum_{x \in E \setminus R} Pr[e_n = x] \\
&= \epsilon Pr[e_n \in R] + (M - \gamma(p)) Pr[e_n \in E \setminus R] \\
&= \epsilon Pr[\|e_n - p\| \leq \delta] + (M - \gamma(p)) Pr[\|e_n - p\| > \delta] \\
&\leq \epsilon(1 - \epsilon') + \epsilon' \big(M - \gamma(p)\big)
\end{aligned}$$

Then combined with $\gamma(p) \leq \mathbb{E}[\gamma(e_n)]$ concludes the proof.

□

Lemma 3. *([Bro+19] lemma 3) Let $y \in \mathbb{R}^m_+$ be a vector of positive entries. Then for any positive $x \in \mathbb{R}^m_+$, we have:*

$$\sum_i y_i \log_2 \left(\frac{x_i}{\sum_j x_j} \right) \leq \sum_i y_i \log_2 \left(\frac{y_i}{\sum_j y_j} \right)$$

with equality if and only if $x_i = k \cdot y_i$ for some $k > 0$.

Proof. Notice that the vectors $x' := \frac{x}{\sum_j x_j}$ and $y' := \frac{y}{\sum_j y_j}$ can be viewed as probability distributions as they are non-negative and sum to one. Also recall the KL-divergence defined as $D_{KL}(y' \| x') = \sum_i \left(y'_i \log_2 \left(\frac{y'_i}{x'_i} \right) \right)$ is non-negative and zero if and only if $x' = y'$

So:

$$0 \leq \sum_i \left(y' \log_2 \left(\frac{y'_i}{x'_i} \right) \right) = \sum_i (y'_i \log_2(y'_i)) - \sum_i (y'_i \log_2(x'_i))$$

Then by multiplying through by $\sum_j y_j$ the desired result follows. □

Lemma 4. *([05] Theorem 2.7.4) The MI$(K; L)$ is a concave function of $p(k)$ for fixed $p(l|k)$ and a convex function of $p(l|k)$ for fixed $p(k)$.*

Proof. For the first part (concavity of MI w.r.t. $p(k)$ when $p(l|k)$ fixed) write:

$$\mathrm{MI}(K; L) = \mathrm{H}(L) - \mathrm{H}(L|K)$$

.

We will show that $\mathrm{H}(L)$ is concave in $p(k)$ and that $\mathrm{H}(L|K)$ is linear in $p(k)$ and then since the sum of a concave function and a linear function is concave we will be done.

Let $g(x) = x \cdot \log_2(x)$, g is convex in x as $g''(x) = \frac{1}{x \cdot \ln(2)} > 0$.

Now write:

$$\mathrm{H}(L) = -\sum_{l \in \mathcal{L}} g(p(l))$$

So $\mathrm{H}(L)$ is concave in $p(l)$ (as it is the negation of a convex function).

Now write $p(l) = \sum_{l \in \mathcal{L}} p(l|k) \cdot p(k)$, as $p(l|k)$ is fixed by assumption we have that $p(l)$ is a linear function of $p(k)$ (and also $p(k)$ is a linear function of $p(l)$)

Therefore $\mathrm{H}(L) = \text{concave}(\text{linear}(p(k)))$ which by checking the derivative with the chain rule shows that $\mathrm{H}(L)$ is concave in $p(k)$.

Now $\mathrm{H}(L|K) = \sum_k p(k) \sum_l p(l|k) log_2\left(\frac{1}{p(l|k)}\right)$ is clearly linear in $p(k)$ as the second sum is completely fixed by assumption.

Therefore we have $\mathrm{MI}(K; L)$ is concave in $p(k)$ for fixed $p(l|k)$.

For the second part, (convexity of $\mathrm{MI}(K; L)$ in $p(l|k)$ when $p(k)$ fixed) write:

$$\mathrm{MI}(K; L) = D_{KL}(p(K, L) \| p(K)p(L))$$

It is known that the KL-Divergence is convex in the pair of it's arguments, i.e. Let u_1, u_2, v_1, v_2 be probability distributions and then $u_\lambda = \lambda u_1 + (1 - \lambda)u_2$ and $v_\lambda = \lambda v_1 + (1 - \lambda)v_2$.

$$D_{KL}(u_\lambda \| u_\lambda) \leq \lambda D_{KL}(u_1 \| v_1) + (1 - \lambda) D_{KL}(u_2 \| v_2)$$

Let $p_1(l|k)$ and $p_2(l|k)$ be probability distributions and L_1 and L_2 be random variables over $\mathcal{L}$ defined by the combination of $p(k)$ and $p_1(l|k), p_2(l|k)$ respectively.

Now set:

$$\begin{aligned}
u_1(k, l) &= p(k)p_1(l|k) = p_1(k, l) \\
u_2(k, l) &= p(k)p_2(l|k) = p_2(k, l) \\
v_1(k, l) &= p(k) \sum_{k'} p(k)p_1(l|k') = p(k)p_1(l) \\
v_2(k, l) &= p(k) \sum_{k'} p(k)p_2(l|k') = p(k)p_2(l) \\
u_\lambda &= \lambda u_1 + (1 - \lambda)u_2 \\
v_\lambda &= \lambda v_1 + (1 - \lambda)v_2
\end{aligned}$$

From which follows:

$$\begin{aligned}
\mathrm{MI}(K; \lambda L_1 + (1 - \lambda)L_2) &= D_{KL}(u_\lambda \| v_\lambda) \\
&\leq \lambda D_{KL}(u_1 \| v_1) + (1 - \lambda) D_{KL}(u_2 \| v_2) \\
&= \lambda \mathrm{MI}(K; L_1) + (1 - \lambda)\mathrm{MI}(K; L_2)
\end{aligned}$$

Which completes the proof. □

Lemma 5. *[Pan03] Let $p(x)$ be a probability distribution defined on domain $\mathcal{X}$ and $e_n(x)$ the empirical distribution derived from p after n observations, then:*

$$\mathbb{E}[D_{KL}(e_n(\cdot) \| p(\cdot)] \leq \log_2(1 + \frac{|\mathcal{X}| - 1}{n})$$

Proof.

$$\begin{aligned}
\mathbb{E}[D_{KL}(e_n(\cdot\|p(\text{cot})] &= \mathbb{E}\Big[\sum_{x\in\mathcal{X}} e_n(x)\log_2\Big(\frac{e_n(x)}{p(x)}\Big)\Big] \\
&\leq \mathbb{E}\Big[\log_2\Big(\sum_{x\in\mathcal{X}} e_n(x)\frac{e_n(x)}{p(x)}\Big)\Big] && \text{(By concavity of the logarithm)} \\
&\leq \log_2\Big(\mathbb{E}\Big[\sum_{x\in\mathcal{X}} e_n(x)\frac{e_n(x)}{p(x)}\Big]\Big) && \text{(By concavity of the logarithm)} \\
&= \log_2\Big(\sum_{x\in\mathcal{X}} \frac{1}{p(x)}\mathbb{E}[e_n(x)^2]\Big)
\end{aligned}$$

Now focusing on $\mathbb{E}[e_n(x)^2]$. We will calculate this from the definition of expectation. Firstly $e_n(x)$ takes values in $\{0, \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \ldots, \frac{n-1}{n}, 1\}$ so $e_n(x)^2$ takes values in $\{0, \frac{1}{n}, \frac{4}{n}, \frac{9}{n}, \ldots, \frac{(n-1)^2}{n}, 1\}$ and the probability that $e_n(x)^2$ takes value $\frac{k^2}{n^2}$ is the probability that $e_n(x)$ takes value $\frac{k}{n}$ which is in turn distributed Binomial(n, k) giving $Pr\big[e_n(x)^2 = \frac{k^2}{n^2}\big] = \binom{n}{k}p(x)^k(1-p(x))^{n-k}$. Then taking the formula for expectation, multiplying values by their probabilities and summing gives:

$$\begin{aligned}
\mathbb{E}[e_n(x)^2] &= \sum_{k=0}^{n} \frac{k^2}{n^2}\binom{n}{k}p(x)^k(1-p(x))^{n-k} \\
&= \frac{p(x)^2(n-1)+p(x)}{n}
\end{aligned}$$

With the summation result calculated by Wolframalpha. Plugging this value into the result form before we obtain:

$$\begin{aligned}
\mathbb{E}[D_{KL}(e_n(\cdot)\|p(\cdot)] &\leq \log_2\Big(\sum_{x\in\mathcal{X}} \frac{1}{p(x)}\frac{p(x)^2(n-1)+p(x)}{n}\Big) \\
&= \log_2\Big(\sum_{x\in\mathcal{X}} \frac{np(x)-p(x)+1}{n}\Big) \\
&= \log_2\Big(\sum_{x\in\mathcal{X}} p(x) - \sum_{x\in\mathcal{X}}\frac{p(x)}{n} + \sum_{x\in\mathcal{X}}\frac{1}{n}\Big) \\
&= \log_2\Big(1-\frac{1}{n}+\frac{|\mathcal{X}|}{n}\Big) \\
&= \log_2\Big(1+\frac{|\mathcal{X}|-1}{n}\Big)
\end{aligned}$$

Which completes the proof. □

3.4 Convergence of eHI

Theorem 2. *([Bro+19] Theorem 5). The expectation of the eHI_n converges monotonically to the MI as $n \to \infty$.*

$$\mathbb{E}(eHI_{n-1}(K;L)) \geq \mathbb{E}(eHI_n(K;L)) \geq MI(K;L)$$

And $\lim_{n\to\infty} eHI_n(K;L) = MI(K;L)$

Proof. $\mathrm{eHI}_n(K;L)$ is the mutual information between the key K and a variable distributed according to the empirical distribution and thanks to Lemma 4 it is convex in $e_n(l|k)$ for fixed $p(k)$. The by applying Lemma 2 on convex functions the result follows. □

The next question to ask is how fast does the eHI converge? The answer unfortunately turns out to be not fast enough.

Theorem 3. *([Mas+22] Theorem 1). Let the leakage space $\mathcal{L} = \{0, \dots 2^\omega - 1\}^D$ i.e. the leakage space consists of D time samples each of resolution ω. Then $|\mathcal{L}| = 2^{\omega D}$. Then the eHI_n satisfies:*

$$MI(K;L) \leq \mathbb{E}[eHI_n(K;L)] \leq MI(K;L) + \frac{|\mathcal{L}||\mathcal{K}|}{n}$$

Moreover,

$$(\mathbb{E}[eHI_n(K;L)] - MI(K;L)) \cdot \frac{n}{|\mathcal{L}||\mathcal{K}|} \xrightarrow{n\to\infty} \frac{1}{2}$$

Proof. First we will rewrite the eHI as:

$$\begin{aligned}\mathrm{eHI}_n(K;L) = \mathrm{MI}(K;L) &\\ + \sum_{k\in\mathcal{K}, l\in\mathcal{L}} &(e_n(k,l) - p(k,l)) \log_2(p(k|l)) \\ + \sum_{l\in\mathcal{L}} &e_n(l) D_{KL}(e_n(\cdot|l) \| p(\cdot|l))\end{aligned}$$

This form of the eHI is obtained through simple expansion of definitions and algebraic manipulation. It is of use to us because it gives a simple form for the difference between the eHI and MI in terms of the KL-Divergence which is a well studied function.

It is clear that the expectation of the first sum over the randomness in the empirical distribution is 0 as the expectation of e_n is p making every term in the sum 0. Now all we need to do is bound the second sum. From the non-negativity of the KL-Divergence and lemma 5 we have:

$$0 \leq \mathbb{E}[D_{KL}(e_n(\cdot|l)\|p(\cdot|l)] \leq \log_2\Big(1 + \frac{|\mathcal{K}| - 1}{n}\Big)$$

Which then implies:

$$\begin{aligned}\mathbb{E}[\mathrm{eHI}_n(K;L) - \mathrm{MI}(K;L)] &= \sum_{l\in\mathcal{L}} \mathbb{E}[e_n(l) \cdot D_{KL}(e_n(\cdot|l)\|p(\cdot|l))] \\ &\leq \sum_{l\in\mathcal{L}} \mathbb{E}[D_{KL}(e_n(\cdot|l)\|p(\cdot|l))] \\ &\leq |\mathcal{L}| \log_2\Big(1 + \frac{|\mathcal{K}| - 1}{n}\Big) \\ &\leq |\mathcal{L}| \frac{|\mathcal{K}| - 1}{n} \\ &\leq \frac{|\mathcal{L}||\mathcal{K}|}{n}\end{aligned}$$

The second statement is proved in [Pan03] Theorem 5. □

One condition that we would like to eliminate is the convergence is only in terms of expectation. For our profiling of an implementation we don't want to have to take many different trace data-sets and average the eHI from each one to obtain a reliable estimate of the MI. Ideally we would be able to take one trace set, compute the eHI from that to obtain an estimate of the MI. To do this we need to prove convergence in a stronger sense than expectation. We will now prove the convergence of the eHI in probability.

Theorem 4. *([Mas+22] Theorem 2). For all $\delta > 0$, the inequality*

$$|eHI_n(K;L) - \mathbb{E}[eHI_n(K;L)]| \leq \log_2(n)\sqrt{\frac{8\ln 4/\delta}{n}}$$

holds with probability greater than $1-\delta$ and

$$\mathbb{E}[|eHI_n - \mathbb{E}[eHI_n]|] \in \Theta\left(\frac{1}{\sqrt{n}}\right)$$

Proof. First let $\widehat{\mathrm{H}(K)} = \sum_l e_n(l)\log_2(\frac{1}{e_n(l)})$ and $\widehat{\mathrm{H}(K,L)} = \sum_k\sum_l e_n(k,l)\log_2(\frac{1}{e_n(k,l)})$ then $\mathrm{eHI}_n(K;L) = \mathrm{H}(K) + \widehat{\mathrm{H}(L)} - \widehat{\mathrm{H}(K,L)}$. Plugging these definitions into $|\mathrm{eHI}_n(K;L) - \mathbb{E}[\mathrm{eHI}_n(K;L)]|$ and the triangle inequality we get:

$$|\mathrm{eHI}_n - \mathbb{E}[\mathrm{eHI}_n]| \leq |\widehat{\mathrm{H}(L)} - \mathbb{E}[\widehat{\mathrm{H}(L)}]| + |\widehat{\mathrm{H}(K,L)} - \mathbb{E}[\widehat{\mathrm{H}(K,L)}]| \tag{3.8}$$

now using McDiarmind's inequality, we have that for all $\epsilon > 0$

$$Pr\Big[|\widehat{\mathrm{H}(L)} - \mathbb{E}[\widehat{\mathrm{H}(L)}]| > \frac{\epsilon}{2}\Big] \leq 2\exp\Big(-\frac{\epsilon^2 n}{8\log_2(n)^2}\Big)$$

and

$$Pr\Big[|\widehat{\mathrm{H}(K,L)} - \mathbb{E}[\widehat{\mathrm{H}(K,L)}]| > \frac{\epsilon}{2}\Big] \leq 2\exp\Big(-\frac{\epsilon^2 n}{8\log_2(n)^2}\Big)$$

Hence:

$$\begin{aligned}
Pr\Big[|\mathrm{eHI}_n - \mathbb{E}[\mathrm{eHI}_n] > \epsilon\Big] &\leq Pr\Big[|\widehat{\mathrm{H}(L)} - \mathbb{E}[\widehat{\mathrm{H}(L)}]| + |\widehat{\mathrm{H}(K,L)} - \mathbb{E}[\widehat{\mathrm{H}(K,L)}]| > \epsilon\Big] && (3.9)\\
&\leq Pr\Big[|\widehat{\mathrm{H}(L)} - \mathbb{E}[\widehat{\mathrm{H}(L)}]| > \frac{\epsilon}{2} \wedge |\widehat{\mathrm{H}(K,L)} - \mathbb{E}[\widehat{\mathrm{H}(K,L)}]| > \frac{\epsilon}{2}\Big] && (3.10)\\
&\leq Pr\Big[|\widehat{\mathrm{H}(L)} - \mathbb{E}[\widehat{\mathrm{H}(L)}]| > \frac{\epsilon}{2}\Big] + Pr\Big[|\widehat{\mathrm{H}(K,L)} - \mathbb{E}[\widehat{\mathrm{H}(K,L)}]| > \frac{\epsilon}{2}\Big] && (3.11)\\
&\leq 2\exp\Big(-\frac{\epsilon^2 n}{8\log_2(n)^2}\Big) + 2\exp\Big(-\frac{\epsilon^2 n}{8\log_2(n)^2}\Big) && (3.12)\\
&= 4\exp\Big(-\frac{\epsilon^2 n}{8\log_2(n)^2}\Big) && (3.13)
\end{aligned}$$

With (2.2) following from (2.1) and the lhs condition being more restrictive than the rhs condition. (2.3) again follows because the rhs condition is less restrictive than the lhs making it more likely. (2.4) follows from the observation that for events A, B $Pr[A \wedge B] = Pr[A]\cdot Pr[B|A] = Pr[B]\cdot Pr[A|B]$ and the fact that $Pr[B|A], Pr[A|B] \leq 1 \implies Pr[A\wedge B] \leq Pr[A], Pr[B] \implies Pr[A\wedge B] \leq Pr[A] + Pr[B]$. Then because Then by plugging in $\epsilon = \log_2(n)\sqrt{\frac{8\ln(4/\delta)}{n}}$ the result follows.

$$\begin{aligned}
Pr\Big[|\mathrm{eHI}_n - \mathbb{E}[\mathrm{eHI}_n] \leq \epsilon\Big] &= 1 - Pr\Big[|\mathrm{eHI}_n - \mathbb{E}[\mathrm{eHI}_n(K;L)]| > \epsilon\Big]\\
&\geq 1 - 4\exp\Big(-\frac{\log_2(n)^2\sqrt{\frac{8\ln(4/\delta)}{n}}^2 n}{8\log_2(n)^2}\Big)\\
&= 1 - 4\exp(-\ln(4/\delta))\\
&= 1 - 4\frac{\delta}{4}\\
&= 1 - \delta
\end{aligned}$$

The second statement is proved in [AK01] section 4.1. □

Unfortunately Theorem 4 implies that the eHI converge slowly, with a rate of $1/\sqrt{n}$ the convergence isn't as fast as desired. To remedy this we move to the next section. It is possible to make use of the eHI it would just require a large number of traces to give an accurate estimate.

3.5 Parametric models

Since the empirical distribution converges too slowly for our purposes we turn our attention to other estimates. One thing to notice is that the attacks detailed before all make use of some kind of parametric model of the distribution rather than estimating the true distribution. We can do the same for our security information estimates.

Definition 5. ***(Δ distance).*** *([Mas+22] Equation 1). let p, m be a probability distributions over a space $(\mathcal{K}, \mathcal{L})$. Then we define a distance metric Δ between the two as:*

$$\Delta_p^m = H(K) - \sum_{k \in \mathcal{K}, l \in \mathcal{L}} p(k,l) \log_2\Big(\frac{1}{m(k|l)}\Big)$$

Note that this in not a true metric as it doesn't equal 0 when p and m are equal. Also note that we can express previous quantities in terms of this function.

$$\mathrm{MI}(K;L) = \Delta_p^p$$

$$\mathrm{eHI}_n(K;L) = \Delta_{e_n}^{e_n}$$

Definition 6. ***(Hypothetical Information).*** *([Bro+19] Equation 7). ($HI(K;L;m)$) let $\mathcal{M}$ be a set of n observations of pairs of keys and leakages (k,l) then let $m \leftarrow \mathcal{M}$ be a model derived from the observations. The $HI(K;L;m)$ is defined as:*

$$\begin{aligned} HI(K;L;m) &= \Delta_m^m \\ &= H(K) - \sum_{k \in \mathcal{K}} m(k) \sum_{l \in \mathcal{L}} m(l|k) \log_2\Big(\frac{1}{m(k|l)}\Big) \end{aligned}$$

Note that the $\mathrm{eHI}_n(K;L) = \mathrm{HI}(K;L;e_n)$. The hypothetical information is the amount of information that could be extracted if the model were the true distribution.

Definition 7. ***(Perceivable Information).*** *([Bro+19] Equation 9). ($PI(K;L;m)$) Let $\mathcal{M}$ be a set of n observations of pairs of keys and leakages (k,l) then let $m \leftarrow \mathcal{M}$ be a model derived from the observations. The $PI(K;L;m)$ is defined as:*

$$\begin{aligned} PI(K;L;m) &= \Delta_p^m \\ &= H(K) - \sum_{k \in \mathcal{K}} p(k) \sum_{l \in \mathcal{L}} p(l|y) \log_2\Big(\frac{1}{m(k|l)}\Big) \end{aligned}$$

The perceivable information is how much information can be extracted from leakages thanks to a model. In general it will be less than the mutual information due to under fitting of the model.

Corollary 1. *([Bro+19] Theorem 6). The perceivable information PI is less than or equal to the mutual information MI*

Proof.

$$\begin{aligned}
\mathrm{PI}(K;L;m) &= \mathrm{H}(K) - \sum_{k\in\mathcal{K}} p(k) \sum_{l\in\mathcal{L}} p(l|k) \log_2\Big(\frac{1}{m(k|l)}\Big) \\
&= \mathrm{H}(K) - \sum_{l\in\mathcal{L}} p(l) \sum_{k\in\mathcal{K}} p(k|l) \log_2\Big(\frac{1}{m(k|l)}\Big) \\
&= \mathrm{H}(K) - \sum_{l\in\mathcal{L}} p(l) \Big[D_{KL}(p(\cdot|l)\|m(\cdot|l)) + \sum_{k\in\mathcal{K}} p(k|l) \log_2\Big(\frac{1}{p(k|l)}\Big)\Big] \\
&= \mathrm{MI}(K;L) - \sum_{l\in\mathcal{L}} p(l) D_{KL}(p(\cdot|l)\|m(\cdot|l)) \\
&\leq \mathrm{MI}(K;L)
\end{aligned}$$

With the last step due to the non-negativity of the KL-Divergence and with equality if and only if the model is equal to the true distribution. It is also important to note that this quantity can't be calculated directly as it makes use of the unknown distribution p. □

Definition 8. ***(Hypothesis class)***, *([Mas+22] Definition 1). A hypothesis class $\mathcal{H}$ is a (possibly infinite) set of models $m : \mathcal{L} \to \mathcal{P}(\mathcal{K})$. Where $\mathcal{L}$ is the space of the leakage data and m maps an observed leakage into a probability distribution over the keys $\mathcal{K}$.*

Some examples of hypothesis classes are the set of Gaussian models, parameterized by mean and standard deviation, or the set of MLP (multi layer perceptron) neural networks.

Definition 9. ***(Learnable Information)***. *([Mas+22] Definition 2). Let $\mathcal{H}$ be a hypothesis class. The learnable information on $\mathcal{K}$ from leakage L using a model from $\mathcal{H}$ is defined as:*

$$LI(K;L;\mathcal{H}) = \sup_{m\in\mathcal{H}} PI(K;L;m)$$

The learnable information is the best that an adversary can do, should they attack with a model in the hypothesis class. Then from theorem 1 an adversary using a model in the hypothesis class $\mathcal{H}$ requires at least $\frac{\mathrm{H}(K)}{\mathrm{LI}(K;L)}$ traces to fully recover the key, and if they are willing to perform λ bits or work they need at least $\frac{\mathrm{H}(K)}{\mathrm{LI}(K;L)}$ traces.

At this point however we still have no way to compute the learnable information, we will introduce a few more definitions and then we will introduce a new metric, the training information. We will then show the training information converges to an upper bound on the learnable information, which in turn allows us to bound the number of traces required.

Definition 10. ***(Learning Algorithm).*** *([Mas+22] Definition 3). A learning algorithm $\mathcal{A}$ for a hypothesis class $\mathcal{H}$ is a function*

$$\mathcal{A} : \Big(\bigcup_{n=1}^{\infty} (\mathcal{K}\times\mathcal{L})^n\Big) \to \mathcal{H}$$

i.e. the algorithms $\mathcal{A}$ takes a set of observations $\mathcal{S}_n$ of size n and returns a model m in the hypothesis class, $m = \mathcal{A}(\mathcal{S}_n)$.

It is also worth noting that an adversary is determined uniquely by their learning algorithm, therefore we will refer interchangeably between adversary and learning algorithms.

Definition 11. ***(Regret).*** *([Mas+22] Definition 4). Let $\mathcal{A}$ be an adversary and $\mathcal{S}_n$ be a set of n observations keys and leakages, the regret of $\mathcal{A}$ is defined as:*

$$R(\mathcal{A}) = MI(K;L) - PI(K;L;\mathcal{A}(\mathcal{S}_n))$$

We have already seen before that PI is less than or equal to the MI so the regret $R(\mathcal{A}) \geq 0$ and is 0 if and only if the algorithm $\mathcal{A}$ outputs the true distribution p.

Definition 12. ***(Training Information).*** *([Mas+22] Definition 5). Let $\mathcal{S}_n$ be a set of n samples drawn from true distribution over (k, l), the training information of $\mathcal{A}$ with n traces is defined as:*

$$\begin{aligned} TI_n(K; L; \mathcal{A}) &= \Delta_{e_n}^{\mathcal{A}(\mathcal{S}_n)} \\ &= H(K) - \sum_k p(k) \sum_l e_n(l|k) \log_2\Big(\frac{1}{(\mathcal{A}(\mathcal{S}_n))(k|l)}\Big) \end{aligned}$$

Since the adversary $\mathcal{A}$ is completely arbitrary there are no guarantees of convergence, however by specifying conditions on $\mathcal{A}$ we can prove convergence properties for the training information.

Definition 13. ***(TI maximizer).*** *([Mas+22] Definition 6). Let $\mathcal{H}$ be a hypothesis class. The TI_n maximizer for the hypothesis class $\mathcal{H}$ is the learning algorithm $\mathcal{A}_\mathcal{H}$ such that $\mathcal{A}_\mathcal{H}(\mathcal{S}_n) = \hat{m}_n$ where $\hat{m}_n$ is defined as:*

$$\hat{m}_n = \operatorname*{argmax}_{m \in \mathcal{H}} \Delta_{e_n}^m$$

Proposition 1. *([Mas+22] Proposition 1). Let $\mathcal{H}$ be a hypothesis class, then the training information for a TI maximizer upper bounds the learnable information in expectation.*

$$\mathbb{E}[TI_{n-1}(K; L; \mathcal{A}_\mathcal{H})] \geq \mathbb{E}[TI_n(K; L; \mathcal{A}_\mathcal{H})] \geq LI(K; L; \mathcal{H})$$

Proof. Let $\mathcal{S}_n$ be the set of observations and $\hat{m}_n$ be the model output by the adversary $\mathcal{A}$, i.e. $\hat{m}_n = \mathcal{A}(\mathcal{S}_n)$

By definition 13 we have for all other models m we have $\Delta_{e_n}^{\hat{m}_n} \geq \Delta_{e_n}^m$. Therefore:

$$\mathbb{E}[\mathrm{TI}_n(K; L; \mathcal{A})] = \mathbb{E}[\Delta_{e_n}^{\hat{m}_n}] \geq \mathbb{E}[\Delta_{e_n}^m] = \Delta_p^m = \mathrm{PI}(K; L; m)$$

with the third step following because Δ_a^b is linear with respect to a. By taking m to be the model that maximizes the PI, since the LI is the maximum of the PI over the models in $\mathcal{H}$, we have that:

$$\mathrm{LI}(K; L; \mathcal{H}) \leq \mathbb{E}[\mathrm{TI}_n(K; L; \mathcal{A}_\mathcal{H})]$$

To show the second inequality we will show that the map $\gamma : x \to \Delta_x^{\hat{m}_p}$ is convex, where we define $\hat{m}_x = \operatorname*{argmax}_{m \in \mathcal{H}} \Delta_x^m$. Then by lemma 2 we will obtain the result.

Let $x, y \in \mathcal{P}(\mathcal{K}, \mathcal{L}), \alpha \in [0, 1]$ then let $z = \alpha x + (1 - \alpha)y$, we show that $\gamma(z) \leq \alpha\gamma(x) + (1 - \alpha)\gamma(y)$. Firstly since Δ_a^b is linear in a we have:

$$\gamma(z) = \Delta_z^{\hat{m}_z} = \alpha\Delta_x^{\hat{m}_z} + (1 - \alpha)\Delta_y^{\hat{m}_t}$$

Then by definition of $\hat{m}_x$ and $\hat{m}_y$ we have $\Delta_x^{\hat{m}_x} \geq \Delta_x^{\hat{m}_z}$ and $\Delta_y^{\hat{m}_y} \geq \Delta_y^{\hat{m}_z}$. Which gives:

$$\gamma(z) \leq \alpha\Delta_x^{\hat{m}_x} + (1 - \alpha)\Delta_y^{\hat{m}_y} = \alpha\gamma(x) + (1 - \alpha)\gamma(y)$$

And therefore γ is convex so when combined with lemma 2 completes the proof.

□

Note that we do not have the convergence property that lemma 2 can provide as while $\Delta_x^{\hat{m}_x}$ is bounded above (specifically by $\mathrm{H}(K; L)$) checking the continuity turns out to be non-trivial as the value of $\hat{m}_x$ depends on x which greatly complicates the calculations.

However we don't actually need convergence to the LI obtain a bound on the number of traces, as $\mathbb{E}[\mathrm{TI}_n] \geq \mathrm{LI}$ we have that the number of traces t to recover the key when the adversary uses a model in $\mathcal{H}$ is bounded as:

$$t \geq \frac{\mathrm{H}(K)}{\mathrm{LI}(K;L;\mathcal{H})} \geq \frac{\mathrm{H}(K)}{\mathbb{E}[\mathrm{TI}_n(K;L\mathcal{A}_\mathcal{H})]}$$

Before Moving on it is worth recapping the results obtained so far. Let $\mathcal{H}$ be a hypothesis class, $\mathcal{A}_\mathcal{H}$ the TI maximizer for $\mathcal{H}$. The bounds can be summed up in a series of inequalities as:

$$\mathrm{PI}(K;L;\mathcal{A}_\mathcal{H}) \leq \mathrm{LI}(K;L;\mathcal{H}) \leq \mathbb{E}[\mathrm{TI}_N(K;L;\mathcal{A}_\mathcal{H})] \leq \mathbb{E}[\mathrm{TI}_{N-1}(K;L;\mathcal{A}_\mathcal{H}]$$

The first inequality comes from Definition 9 as the supremum over the Perceivable information. The second and third from 1.

3.6 Convergence of TI

In [Mas+22] it is shown that for a specific hypothesis class that includes MLP neural networks and Logistic regression models with polynomial degree k the following inequalities hold for the TI_N maximizer $\mathcal{A}_\mathcal{H}$:

$$\mathrm{LI}(K;L;\mathcal{H}) - \tilde{\mathcal{O}}\left(\frac{1}{N}\right) \leq_{\text{h.p.}} \mathrm{PI}(K;L;\mathcal{A}_\mathcal{H}) \leq \mathrm{LI}(K;L;\mathcal{H})$$

$$\mathrm{LI}(Y;L;\mathcal{H}) \leq \mathbb{E}[\mathrm{TI}_N(K;L;\mathcal{A}_\mathcal{H}] \leq \mathrm{LI}(K;L;\mathcal{H}) + \tilde{\mathcal{O}}\left(\frac{1}{N}\right)$$

Where $\tilde{\mathcal{O}}$ is big $\mathcal{O}$ notation that ignores the log terms and $\leq_{\text{h.p.}}$ holds with high probability. This shows that the perceivable information converges to the Learnable information from below and the Training information converges from above. With these results by using a TI maximizer a reliable estimate on the LI can be obtained. This bound can then be translated into either a bound on the number of traces via Theorem 1 or an estimate of the success rate by [Che+19] or [DFS15]. Additionally results are proved for Gaussian templates (which are in a different hypothesis class to the MLP and Logistic regression models).

Chapter 4

Masking

In the previous chapter we saw how to quantify the security an implementation, now we turn our attention to the task of designing implementations. Specifically we ask what measures can we use to turn an insecure design of and mathematical algorithm into a secure one and furthermore how to define security. There are multiple approaches that can be taken, in this chapter we will focus on the technique of masking. The basic idea is to split each sensitive value into d multiple values called secret shares. The goal being that to recover the secret key all d of the shares would need to be recovered making the adversaries job harder. This should force the adversary to estimate not the distributions over one value, but the joint distribution over d values.

The first section will define masking, then define several security models (Noisy leakage, Random probing, Threshold probing) that we will prove reductions between. The final model (Threshold probing) is abstract enough that it can be used to motivate designs for secure circuits.

Definition 14. ***(Masking Scheme).*** *A masking scheme over a set $\mathcal{X}$ is a pair of functions Enc and Dec, where:*

$$Enc : \mathcal{X} \to \mathcal{X}^d \text{is a randomized function called the encoder}$$
$$Dec : \mathcal{X}^d \to \mathcal{X} \text{is a function called the decoder.}$$

With the following conditions on the functions:

- $\forall x \in \mathcal{X} \quad Dec(Enc(x)) = x$
- *Each subset from the output of Enc of size at most $d-1$ is uniformly distributed and independent of x*

We will call $\mathbf{x} = Enc(x)$ a sharing of x and will typically denote sharings as bold lower case letters, the same as the variable they decode to.

The are multiple methods of masking variables, we will focus our attention on arithmetic masking.

Definition 15. ***Arithmetic Masking*** *is a masking scheme over the finite field $\mathbb{F}$ where $q = p^n$ for some prime p. Defined by the decoder as:*

$$Dec(x_1, \ldots, x_d) = x_1 + \cdots + x_d (mod\ q)$$

One method of encoding is:

Algorithm 4.1: Enc

input : $x \in \mathbb{F}$
output: $x_1, \ldots, x_d \in \mathbb{F}$ *such that* $\sum_{i=1}^{n} x_i = x$
for $i = 2$ *to* d **do**
  $x_i \xleftarrow{\$} \mathbb{F}$
end
$x_1 \leftarrow x - (x_2 + \cdots + x_d)$
return $x_1, \ldots x_d$

To see that this encoder does indeed have the property that and subset from the output of Enc of size at most $d-1$ is independent of x first notice that clearly any subset containing only elements from $x_2, \ldots, x_d$ is independent of x. Now to see that subsets containing x_1 are independent of x recall that if $Pr[X = x] = Pr[X = x|Y]$ for all x then X is independent of Y. We will show this by induction on d.

When $d = 2$ we have from Bayes theorem:

$$\begin{aligned} Pr[X = x|X_2 = x_2] &= \frac{Pr[X_2 = x_2|X = x]Pr[X = x]}{Pr[X_2 = x_2]} \\ &= \frac{1/|\mathbb{F}| \cdot Pr[X = x]}{1/|\mathbb{F}|} \\ &= Pr[X = x] \end{aligned}$$

Where X is the r.v. representing the secret x and X_i the r.v. representing the i-th share. The second step follow from:

$$Pr[X_2|X = x] = Pr[X_1 = x - x_2] = 1/|\mathbb{F}|$$

and,

$$\begin{aligned} Pr[X_2 = x_2] &= \sum_{x' \in \mathbb{F}} Pr[x_2 = x_2|X = x'] \cdot Pr[X = x'] \\ &= \sum_{x' \in \mathbb{F}} Pr[X_1 = x' - x_2] \cdot Pr[X = x'] \\ &= 1/|\mathbb{F}| \sum_{x' \in \mathbb{F}} Pr[X = x'] \\ &= 1/|\mathbb{F}| \end{aligned}$$

Now assume that the result holds for less that d, let $X' = X_d +_q X_{d-1}$ and notice that X' is uniformly random over $\mathbb{F}$, then we have that $X_1 = X - (X_2 + \cdots + X_{d-2} + X')$. So as X_1 is the difference between the secret and $d-1$ uniform random variables by the induction hypothesis the result holds and we are done.

Definition 16. ***Boolean Masking*** *refers to arithmetic masking when the field $\mathbb{F}$ is $\mathbb{F}_2$.*

When using Boolean masking it is common not to think about each bit separately but to think of bytes as vectors of bits and each element is masked independently. It is worth noting that Boolean masking can be especially efficient due to it's close alignment with logic gates, addition is equivalent to xor and multiplication is equivalent to and.

Now we have defined masking and gained an intuition as to why it would provide security against side-channel attacks we turn our attention to the task of transforming an algorithm into it's masked representations. The first step of which is formalizing what we mean by an implementation. We will only focus on hardware implementations (as opposed to software) due to the high complexity of software implementations, to this end we start by defining a model for a circuit.

4.1 Circuit Model

Definition 17. ***(Ordered Set).*** *An ordered set $\mathcal{X}$ is a set such that $\forall x \in \mathcal{X} x$ is associated with a unique $n \in \{1, \ldots, |\mathcal{X}|\}$. We will interchangeably refer to elements of $\mathcal{X}$ by their element in $\mathcal{X}$ and by their representation as a natural number.*

If $\mathcal{X}$ and $\mathcal{Y}$ are disjoint ordered set then $\mathcal{Z} = \mathcal{X} \cap \mathcal{Y}$ is a new ordered set where $n \in \{1, \ldots, |\mathcal{X}|\}$ is associated with the same element $x \in \mathcal{Z}$ and $n \in \{|\mathcal{X}| + 1, \ldots, |\mathcal{X}| + |\mathcal{Y}|\}$ is associated with $y \in \mathcal{Z}$ such that $n - |\mathcal{X}|$ is associated with $y \in \mathcal{Y}$.

Definition 18. ***(Circuit).*** *A circuit $\mathcal{C}$ over a finite field $\mathbb{F}$ is a directed acyclic graph where nodes represent logic gates and edges represent wires also equipped with a set of input wires. Formally $\mathcal{C} = (V, E, \mathcal{I}, \mathcal{O})$ where V is an ordered set of gates, E is an ordered set of edges (we will call internal wires) that connect one gate to another; $\mathcal{I}$ is the ordered set of input wires and $\mathcal{O}$ is a set of wires called the outputs. Each input acts as an edge of the graph into a gate but from outside and each output acts as an edge of the graph out from a gate but to outside.*

We will denote the set of wires $\mathcal{W} = E \cap \mathcal{I} \cap \mathcal{O}$. Note that since all of $E, \mathcal{I}, \mathcal{O}$ are ordered set $\mathcal{W}$ is also an ordered set.

Every circuit $\mathcal{C}$ over $\mathbb{F}$ can also be thought of as a function $\mathcal{C} : \mathbb{F}^{|\mathcal{I}|} \to \mathbb{F}^{|\mathcal{O}|}$. Where $\mathcal{C}(\mathbf{m})$ is defined by setting the i-th input wire to have the value $\mathbf{m}_i$ and then propagating the values through the circuit.

Propagation is defined as follows: Each gate waits until it receives a value at each input wire then sends a value on each output wire as a function of the inputs. When all the output wires have taken values the output of the function c_i is defined to be the value of the i-th output wire.

We also give names to some gates that appear frequently.

- Random gates, have no inputs and one output. On each execution the gate provides a uniformly random value.
- Constant gates, have no inputs and one output. On each execution the gate provides the same constant value.
- Copy gates, have one input and two outputs. Both outputs take the value of the input
- Arithmetic gates, have two inputs x and y and one output z. The output z takes the value $x \circ y$ where $\circ$ typically represents one of $+ - \times \div$

The circuits we are considering will be implementing some cryptographic algorithm so natural question to ask is how to represent the key. The key will encoded by the values of constant gates. For example in a circuit implementing AES a natural choice of the field to work over is $\mathbb{F}_{2^8}$, the key is 128 bits long broken into a 4x4 matrix where each element is encoded into the value of one constant gate in the circuit. By $\mathcal{C}_k$ we denote the circuit $\mathcal{C}$ where the constant gates representing the key take the values representing k.

Definition 19. ***(Subcircuit).*** *Let $\mathcal{C}' = (V', E', \mathcal{I}', \mathcal{O}')$ and $\mathcal{C} = (V, E, \mathcal{I}, \mathcal{O})$, we say $\mathcal{C}'$ is a subcircuit of $\mathcal{C}$, $\mathcal{C}' \subset \mathcal{C}$ if (V', E') forms a induced subgraph of (V, E) and the inputs $\mathcal{I}'$ are the inputs in $\mathcal{I}$ that go into gates in V', likewise the outputs $\mathcal{O}'$ are the outputs in $\mathcal{O}$ that go out of gates in V'.*

Definition 20. ***(Masking compiler).*** *Let $Enc : \mathbb{F} \to \mathbb{F}^d$ and $Dec : \mathbb{F}^d \to \mathbb{F}$ be a uniform masking scheme. A masking compiler with respect to Enc and Dec takes a circuit $\mathcal{C} : \mathbb{F}^n \to \mathbb{F}^m$ and outputs a circuit $\mathcal{C}' : (\mathbb{F}^d)^n \to (\mathbb{F}^d)^m$ such that the circuit $\mathcal{C}'$ implements the same function as $\mathcal{C}$ with respect to the masking scheme. The inputs and outputs of $\mathcal{C}'$ are grouped as d dimensional vectors. If we let:*

$$(y^1, \ldots, y^m) = \mathcal{C}(x^1, \ldots, x^n)$$

$$(\mathbf{y}^1, \ldots, \mathbf{y}^m) = \mathcal{C}'(\mathbf{x}^1, \ldots, \mathbf{x}^n)$$

Then $Enc(x^1) = \mathbf{x}^i$ for all i implies $Dec(\mathbf{y}^i) = y^i$ for all i.

Simple compilers work by replacing each wire in $\mathcal{C}$ with a bundle of d wires in $\mathcal{C}'$, where the bundle represents a sharing of the original wire, and replacing each gate with a gadget that implements the same functionality with respect to the sharing. More complex compilers can consider multiple gates at a time and output a more efficient implementation. If a circuit $\mathcal{C}$ with inputs $x^1, \ldots, x^n$ and outputs $y^1, \ldots, x^m$ is compiled into a masked circuit $\mathcal{C}'$ with d shares we group the inputs and outputs of $\mathcal{C}'$ into d-dimensional vectors $\mathbf{x}^1, \ldots, \mathbf{x}^n$ and $\mathbf{y}^1, \ldots, \mathbf{y}^m$. Each constant gate representing part of the secret key will be replaced by a sharing of the value it would have had.

Definition 21. ***(Gadget).*** *([Cas] Definition 1). Given a logic f gate with n inputs and m outputs a d shares gadget for that logic gate is a circuit with nd input wires and md output wires with the inputs grouped into sharings $\left(\mathbf{x}^i\right)_{i=1,\ldots n}$ and the outputs grouped into sharings $\left(\mathbf{y}^i\right)_{i=1,\ldots m}$ that implements the function f with respect to the masking scheme, i.e.*

$$f(x^1, \ldots, x^n) = \Big(Dec(\mathbf{y}^1), \ldots, Dec(\mathbf{y}^m)\Big)$$

Where $\mathbf{x}^i = Enc(x^i)$.

Definition 22. ***(Sequential Composition).*** *([Cas] Definition 2). Let* $G_1 = (V_1, E_1, \mathcal{I}_1, \mathcal{O}_1)$ *and* $G_1 = (V_2, E_2, \mathcal{I}_2, \mathcal{O}_2)$ *be gadgets. If* $|\mathcal{O}_1| = |\mathcal{I}_2|$ *then the sequential composition of* G_1 *and* G_2 *is a new gadget* $G = G_1 \circ G_2 = (V, E, \mathcal{I}, \mathcal{O})$ *defined as:*

- *The set of gates of* G *is the union of the gates in* G_1 *and* G_2, $V = V_1 \cup V_2$.
- *The input wires of* G *are the input wires of* G_1, $\mathcal{I} = \mathcal{I}_1$.
- *The output wires of* G *are the output wires of* G_2, $\mathcal{O} = \mathcal{O}_2$.
- *The internal wires of* G *are the internal wires of* G_1 *union those of* G_2 *and the joining wires* J, $E = E_1 \cup E_2 \cup J$. *The joining wires* J *are the set of wires that connect output of the gate that the* i*-th output of* G_1 *comes from the the input of the gate that the* i*-th input of* G_2 *goes to.*

Definition 23. ***(Parallel Composition).*** *([Cas] Definition 3). Let* $G_1 = (V_1, E_1, \mathcal{I}_1, \mathcal{O}_1)$ *and* $G_1 = (V_2, E_2, \mathcal{I}_2, \mathcal{O}_2)$ *be gadgets. The parallel composition of* G_1 *and* G_2 *is a new gadget* $G = G_1 \| G_2 = (V, E, \mathcal{I}, \mathcal{O})$ *defined as:*

- *The set of gates of* G *is the union of the gates in* G_1 *and* G_2, $V = V_1 \cup V_2$.
- *The input wires of* G *is the union of the inputs to* G_1 *and to* G_2, $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$.
- *The output wires of* G *is the union of the outputs of* G_2 *and of* G_2, $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$.
- *The internal wires of* G *are the internal wires of* G_1 *union those of* G_2, $E = E_1 \cup E_2$.

4.2 Security Models

In this section three security models are presented: The Noisy leakage model, The Random probing model and the Threshold Probing model. Briefly in the noisy leakage the adversary gets a randomized noisy function of each wires value. In the Random probing model the adversary gets the (precise) value of each wire with probability p. In the Threshold probing model the adversary gets to choose at most t wires and then receives the (precise) values of those wires. Each model is more abstract than the one before it making it simultaneously easier to reason and less related to the actual side channel security. After each model is introduced a reduction is presented to the previous giving the result for a circuit $\mathcal{C}$: $\mathcal{C}$ is Threshold probing secure $\implies$ $\mathcal{C}$ is Random probing secure $\implies$ $\mathcal{C}$ is noisy leakage secure, For some values of parameters of the security definition to be specified later.

4.2.1 Noisy Leakage

In this section the δ-Noisy leakage model, where an adversary gets a noisy function of each intermediate value, is presented. To define the model we first need to state some probability definitions and properties. First the Total variation distance is defined, this measures how similar two probability distributions are, we will use it to measure the difference between a random variable and a noisy function of that random variable, the distance between these distributions defines one way of quantifying how noisy a function is.

Definition 24. ***(Total Variation distance).*** *Let* X, Y *be random variables defined over the same space* $\mathcal{X}$. *Define the statistical distance between them as:*

$$D_{TV}(X;Y) = \frac{1}{2} \sum_{x \in \mathcal{X}} |Pr[X = x] - Pr[Y = y]|$$

Additionally if E *and* F *are events we define* $D_{TV}(X|E;Y|F) = \delta(X';Y')$ *where* X' *and* Y' *are distributed by* $Pr[X' = x] = Pr[X = x|E]$ *and* $Pr[Y' = x] = Pr[Y = x|F]$. *If* E *is and event with* $Pr[E] = 1$ *then we drop it and simply write* $\delta(X, Y|F)$

If Z *is a r.v. defined over the set* $\mathcal{X}$ *then we define* $D_{TV}(X;Y|Z) = \sum_{x \in \mathcal{X}} Pr[Z = x] \cdot D_{TV}(X;Y|Z = x)$. *and* $D_{TV}((X;Y)|Z) = \sum_{x \in \mathcal{X}} Pr[Z = x] \cdot D_{TV}(X|Z = x; Y|Z = x)$.

Furthermore we say $X \overset{d}{=} Y$ *if and only if* $D_{TV}(X;Y) = 0$ *and we say* X *is* ϵ*-close to* Y *if* $D_{TV}(X;Y) \leq \epsilon$.

The total variation distance is also know as the statistical distance in some texts. Next properties of the total variation distance are stated.

Lemma 6. *([DDF] Lemma 1). Let X and Y be random variables (not necessarily independent). Let Y' be distributed identically to Y but independent from X then:*

$$D_{TV}(X; X|Y) = D_{TV}((Y; Y')|X)$$

Proof.

$$\begin{aligned}
\delta(X; (X|Y)) &= \frac{1}{2} \sum_{x \in \mathcal{X}} Pr[Y = x] \cdot \sum_{y \in \mathcal{X}} |Pr[X = x] - Pr[X = x|Y = y]| \\
&= \frac{1}{2} \sum_{x,y \in \mathcal{X}^2} |Pr[Y = y] \cdot Pr[X = x] - Pr[Y = y] \cdot Pr[X = x|Y = y]| \\
&= \frac{1}{2} \sum_{x,y, \in \mathcal{X}^2} |Pr[X = x \wedge Y' = y] - Pr[X = x \wedge Y = y]| \\
&= \frac{1}{2} \sum_{x \in \mathcal{X}} Pr[X = x] \sum_{y \in \mathcal{X}} |Pr[Y' = y|X = x] - Pr[Y = y|X = x]| \\
&= \frac{1}{2} \sum_{x \in \mathcal{X}} Pr[X = x] \delta((Y; Y')|X = x) \\
&= \delta((Y; Y')|X)
\end{aligned}$$

□

Definition 25. ***(δ Noisy function).*** *([DDF] Equation 5). Let $Noise : \mathcal{X} \to \mathcal{Y}$ be a randomized function. Let X be a random variable uniformly distributed over $\mathcal{X}$. Noise is called δ-noisy if:*

$$\delta = D_{TV}(X; (X|Noise(X))$$

Corollary 2. *([DDF] Equation 6) If $Noise : \mathcal{X} \to \mathcal{Y}$ is δ noisy then by lemma 6 we have:*

$$D_{TV}((Noise(X); Noise(X'))|X) = D_{TV}(X; (X|Noise(X)) = \delta$$

For X' distributed identically to X and independent of X.

Definition 26. ***(p-identity).*** *([DDF] Definition 1). Let $p \in [0, 1]$ and $\varphi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ be a randomized function. φ is called the p identity function if $\varphi(x) \in \{x, \perp\}$, $Pr[\varphi(x) \neq \perp] = p$ and $Pr[\varphi(x) = \perp] = 1-p$.*

The p-identity function will serve as a translation between the noisy leakage model and the random probing model. Recall in the random probing model the adversary receives values of wires with probability p, or equivalently they receive φ of the values where φ is the p identity function.

Lemma 7. *([DDF] Lemma 4). For all randomized functions $Noise : \mathcal{X} \to \mathcal{Y}$ such that $Noise$ is a δ-noisy function for $\delta \leq \frac{1}{|\mathcal{X}|}$. There exists $p \leq \delta \cdot |\mathcal{X}|$ and a randomized function $Noise' : \mathcal{X} \cup \{\perp\} \to \mathcal{X}$, such that for every $x \in \mathcal{X}$ we have:*

$$Noise(x) \stackrel{d}{=} Noise'(\varphi(x))$$

Where $\varphi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ is the p identity function.

Proof. Observe that if we define

$$Pr[\text{Noise}'(x) = y] = \frac{Pr[\text{Noise}(x) = y] - f(y)}{p}$$

and

$$Pr[\text{Noise}'(\perp) = y] = \frac{f(y)}{1-p}$$

For some p and $f(y)$ then we have:

$$\begin{aligned}
Pr[\text{Noise}'(\varphi(x)) = y] &= Pr[\varphi(x) = x] \cdot Pr[\text{Noise}'(x) = x] + Pr[\varphi(x) = \perp] \cdot Pr[\text{Noise}'(\perp) = y] \\
&= p \cdot Pr[\text{Noise}'(x) = x] + (1-p) \cdot Pr[\text{Noise}'(\perp) = y] \\
&= p \cdot \frac{Pr[\text{Noise}(x) = y] - f(y)}{p} + (1-p) \cdot \frac{f(y)}{1-p} \\
&= Pr[\text{Noise}(x) = y] - f(y) + f(y) \\
&= Pr[\text{Noise}(x) = y]
\end{aligned}$$

Which then clearly implies $\text{Noise}(x) \stackrel{d}{=} \text{Noise}'(\varphi(x))$. For this to define a valid probability distribution we must have that each probability is between 0 and 1 and also the probabilities sum to 1:

$$0 \leq Pr[\text{Noise}'(x) = y] \leq 1 \tag{4.1}$$
$$0 \leq Pr[\text{Noise}'(\perp) = y] \leq 1 \tag{4.2}$$
$$\sum_{y \in \mathcal{Y}} Pr[\text{Noise}'(x) = y] = 1 \tag{4.3}$$
$$\sum_{y \in \mathcal{Y}} Pr[\text{Noise}'(\perp) = y] = 1 \tag{4.4}$$
$$0 \leq p \leq 1 \tag{4.5}$$

Condition 4.1 gives:

$$\begin{aligned}
0 \leq Pr[\text{Noise}'(x) = y] \leq 1 &\iff 0 \leq \frac{Pr[\text{Noise}(x) = y] - f(y)}{p} \leq 1 \\
&\iff 0 \leq Pr[\text{Noise}(x) = y] - f(y) \leq 1 \\
&\iff -1 \leq f(y) \leq Pr[\text{Noise}(x) = y]
\end{aligned}$$

With the lower bound in the last step because $Pr[\text{Noise}(x) = y] \geq 0$. Taking $f(y) = \min_{x \in \mathcal{X}}\big(Pr[\text{Noise}(x) = y]\big)$ satisfies condition 4.1. Condition 4.3 gives:

$$\begin{aligned}
1 &= \sum_{y \in \mathcal{Y}} Pr[\text{Noise}'(x) = y] \\
&= \sum_{y \in \mathcal{Y}} \frac{Pr[\text{Noise}(x) = y] - f(y)}{p} \\
&= \frac{1}{p} \sum_{y \in \mathcal{Y}} Pr[\text{Noise}(x) = y] + \frac{1}{p} \sum_{y \in \mathcal{Y}} f(y) \\
&= \frac{1}{p} - \frac{1}{p} \sum_{y \in \mathcal{Y}} f(y) \\
\implies p &= 1 - \sum_{y \in \mathcal{Y}} f(y)
\end{aligned}$$

With the second to last step because the probabilities over Noise sum to 1. Taking p as such also satisfies the condition 4.2 as follows:

$$\begin{aligned}
0 \leq Pr[\text{Noise}'(\perp) = y] \leq 1 &\iff 0 \leq \frac{f(y)}{1-p} \leq 1 \\
&\iff 0 \leq f(y) \leq 1 - p \\
&\iff 0 \leq f(y) \leq 1 - \left(1 - \sum_{y' \in \mathcal{Y}} f(y')\right) \\
&\iff 0 \leq f(y) \leq \sum_{y' \in \mathcal{Y}} f(y')
\end{aligned}$$

Where the last condition is trivially true since $f(y)$ will be counted in the sum to the right, and is greater than 0 by it's definition. We can also show that condition 4.4 is satisfied as follows.

$$\begin{aligned}
\sum_{y \in \mathcal{Y}} Pr[\text{Noise}'(\perp) = y] = 1 &\iff \sum_{y \in \mathcal{Y}} \frac{f(y)}{1-p} = 1 \\
&\iff \sum_{y \in \mathcal{Y}} f(y) = 1 - p \\
&\iff \sum_{y \in \mathcal{Y}} f(y) = 1 - \left(1 - \sum_{y' \in \mathcal{Y}} f(y')\right) \\
&\iff \sum_{y \in \mathcal{Y}} f(y) = \sum_{y' \in \mathcal{Y}} f(y')
\end{aligned}$$

Which is clearly true. Now it is shown that $p \leq \delta \cdot |\mathcal{X}|$.

$$\begin{aligned}
p &= 1 - \sum_{y \in \mathcal{Y}} f(y) \\
&= \sum_{y \in \mathcal{Y}} Pr[Noise(X') = y] - \sum_{y \in \mathcal{Y}} f(y) \\
&= \sum_{y \in \mathcal{Y}} \big(Pr[\text{Noise}(X') = y] - \min_{x \in \mathcal{X}}(Pr[\text{Noise}(x) = y])\big) \\
&= \sum_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} \big(Pr[\text{Noise}(X') = y] - Pr[Noise(x) = y]\big) \\
&\leq \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} \max\big(0, Pr[\text{Noise}(X')] - Pr[\text{Noise}(x) = y]\big) \qquad (4.6) \\
&= \sum_{x \in \mathcal{X}} D_{\text{TV}}(\text{Noise}(x); \text{Noise}(X')) \\
&= |\mathcal{X}| \cdot D_{\text{TV}}(\text{Noise}(X); \text{Noise}(X')|X) \\
&= \delta \cdot |\mathcal{X}| \qquad (4.7)
\end{aligned}$$

The inequality in line 4.6 comes from the fact that the maximum value over x will be included in the sum after so since the over terms are non-negative the sum is larger. The introduction of δ in line 4.7 comes from the assumption that Noise is δ noisy. We have $p \leq \delta \cdot |\mathcal{X}|$ and now show that $p \geq 0$.

$$\begin{aligned} p &= 1 - \sum_{y \in \mathcal{Y}} f(y) \\ &= 1 - \sum_{y \in \mathcal{Y}} \min_{x \in \mathcal{X}} \big(Pr[\text{Noise}(x) = y] \big) \\ &\geq 1 - \sum_{y \in \mathcal{Y}} Pr[\text{Noise}(x_0) = y] \\ &\geq 0 \end{aligned}$$

Where x_0 is any arbitrary element in $\mathcal{X}$. Therefore $0 \leq p \leq \delta \cdot |\mathcal{X}| \leq 1$ which completes condition 4.5 and also the proof. The issue of efficiently sampling from isn't considered instead referring to [DDF] for this. □

We now present a game based notion of security under the δ-Noisy leakage model Let $\mathcal{C} : \mathbb{F}^{nd} \to \mathbb{F}^{md}$ be a masked circuit with inputs $\mathbf{x}^1, \ldots \mathbf{x}^m$ and outputs $\mathbf{y}^1, \ldots, \mathbf{y}^m$. An adversary chooses a sequence of independent noisy functions $(\text{Noise}_1, \ldots, \text{Noise}_{|\mathcal{W}|}$, each $\text{Noise}_i : \mathbb{F} \to \mathcal{L}$ being $\delta_i \leq \delta$ noisy and the inputs to the circuit $x^1, \ldots, x^n$. The adversary then receives the evaluation of the noisy functions on the wires $\text{Noise}_w(\text{val}(w))$ for each $w \in \mathcal{W}$ and makes guess of the secret key. The adversary wins if the guessed key is correct.

Algorithm 4.2: $\text{Exp}^{\text{NL}}_{\mathcal{C}}(\mathcal{A})$

$k \xleftarrow{\$} \mathcal{K}$;
$(\text{Noise}_1, \ldots, \text{Noise}_{|\mathcal{W}|}) \leftarrow \mathcal{A}()$
$(x^1, \ldots, x^n) \leftarrow \mathcal{A}()$
$\mathbf{x}^i \leftarrow \text{Enc}(x^i)$
$\hat{k} \xleftarrow{\$} \mathcal{A}(\text{Noise}(\mathcal{C}_k(\mathbf{x}^1, \ldots, \mathbf{x}^n)))$

Where $\text{Noise}(\mathcal{C}_k(\mathbf{x}^1, \ldots, \mathbf{x}^n)) = \{\text{Noise}_w(\text{val}(w)) | w \in \mathcal{W}\}$ and each $\text{Noise}_w$is a δ noisy function independent from all other $Noise_{w'}$ for $w \neq w'$, and $\text{val}(w) : \mathcal{W} \to \mathbb{F}$ takes a wires and returns the last value that wire took. Essentially the adversary can encrypt arbitrary messages and obtained their ciphertext as well as noisy readings of every intermediate value in the computation. The adversary $\mathcal{A}$ wins if the guess $\hat{k} = k$. The advantage of the adversary is defined as:

$$\text{Adv}^{\text{NL}}_{\mathcal{C}}(\mathcal{A}) = Pr[\text{Exp}^{\text{NL}}_{\mathcal{C}}(\mathcal{A}) : \hat{h} = k]$$

Definition 27. ***(ϵ-δ-Noisy Leakage secure).*** *([DDF]) Let $\delta > 0$. We say a circuit $\mathcal{C}$ is δ-Noisy Leakage secure if for all adversaries $\mathcal{A}$ ϵ^{NL} have $Adv^{NL}_{\mathcal{C}}(\mathcal{A}) \leq \epsilon$.*

It is also important to note that there is a gap between the noisy leakage model and reality, specifically we are assuming that the noisy reading of each intermediate value is independent of all other readings. This is not true in real hardware due to coupling between wires and especially not true in software as multiple values can be in the cpu pipeline at the same time, making the noise on each value highly dependent on other values.

As the noisy leakage model is close to reality it is quite difficult to reason in, therefore we now move on to a more abstract model, the the random probing model.

4.2.2 Random Probing

In the random probing model (RP) the adversary gets the value of each wire with probability p and they get $\perp$ with probability $1 - p$. Next a game based definition of the random probing security model is stated.

Algorithm 4.3: $\text{Exp}^{\text{RP}}_{\mathcal{C}}(\mathcal{A})$

$k \xleftarrow{\$} \mathcal{K}$;
$(x^1, \ldots, x^n) \leftarrow \mathcal{A}()$
$\mathbf{x}^i \leftarrow \text{Enc}(x^i)$
$\hat{k} \xleftarrow{\$} \mathcal{A}(\text{Probes}(\mathcal{C}_k(\mathbf{x}^1, \ldots, \mathbf{x}^n)))$

Where $\text{Probes}(\mathcal{C}) = \{\text{val}(w) \text{ w.p } p, \perp \text{ otherwise } | w \in \mathcal{W}\}$.

We now define the advantage of $\mathcal{A}$ as:

$$\text{Adv}^{RP}_{\mathcal{C}}(\mathcal{A}) = Pr\left[\text{Exp}^{\text{RP}}_{\mathcal{C}}(\mathcal{A}) : \hat{\mathbf{k}} = \mathbf{k}\right]$$

Definition 28. ***(ϵ-p-Random probing secure).*** *([DDF]). $0 < \epsilon, p < 1$. We say a circuit $\mathcal{C}$ is ϵ-δ-Random probing secure if for all adversaries $\mathcal{A}$, $Adv^{RP}_{\mathcal{C}}(\mathcal{A}) \leq \epsilon$.*

Lemma 8. *([DDF] Lemma 5). If a circuit $\mathcal{C}$ is ϵ-p-Random probing secure then it is ϵ-δ-Noisy leakage secure. Where:*

$$\delta = \frac{p}{|\mathcal{X}|}$$

Proof. The proof presented is a outline of the proof presented in [DDF]. The issue of efficiently simulating the noise is not dealt with, refer to [DDF] for full detail. The form of the proof is a reduction. Assume there exists an adversary $\mathcal{A}$ in the δ-Noisy leakage model that wins it's game with advantage ϵ. We will construct an adversary $\mathcal{B}$ in the p-Random probing game with advantage ϵ.

First $\mathcal{B}$ receives the noisy functions requested from $\mathcal{A}$. Each Noise_w is δ noisy so by Lemma 7 There exists $\text{Noise}'_w : \mathbb{F} \cup \{\perp\}$ such that $\text{Noise}'_w(\varphi_w(x)) \stackrel{d}{=} \text{Noise}_w(x)$ for all $x \in \mathbb{F}$ and $w \in \mathcal{W}$ and where φ_w is the $p_w \leq \delta \cdot |\mathbb{F}|$ identity function. The adversary then receives the set of probes which is equivalent to $\varphi_p(\text{val}(w))$ for each $w \in \mathcal{W}$, to meet the conditions of 7 for each value received $\mathcal{B}$ erases the value of wire w with probability $p - p_w$ to obtain $\varphi_w(\text{var}(w))$. Then as $\text{Noise}'_w(\varphi(\text{val}(w))) \stackrel{d}{=} \text{Noise}_w(\text{val}(w))$ by giving the lhs to $\mathcal{A}$ and outputting the same guess as $\mathcal{A}$, when $\mathcal{A}$ succeeds $\mathcal{B}$ success and so the advantage is the same, completing the proof. □

The random probing model is simple enough that security proofs for circuits can be completed directly as is. We won't discuss this yet and first move to the threshold probing model, which is simpler and therefore even easier to reason in.

4.2.3 Threshold Probing

An adversary in the threshold probing model gets to chose a set $\mathcal{P}$ of at most t wires in $\mathcal{W}$ and then receive the value of of these probes for some number of executions of the circuit. Let $\mathcal{C}(\mathbf{x}^1, \ldots, \mathbf{x}^n)_{\mathcal{P}}$ be the set of values of probes in $\mathcal{P}$, $\{\text{val}(w) : w \in \mathcal{P} \subset \mathcal{W}\}$. We now present the game based definition of security under the threshold probing model.

Algorithm 4.4: $\text{Exp}^{\text{TP}}_{\mathcal{C}}(\mathcal{A})$

$k \xleftarrow{\$} \mathcal{K}$;
$(\mathcal{P}, x^1, \ldots, x^n) \leftarrow \mathcal{A}()$
$\hat{k} \xleftarrow{\$} \mathcal{A}(\mathcal{C}_k\left(\text{Enc}(x^1), \ldots, \text{Enc}(x^n)\right)_{\mathcal{P}})$

We define the advantage of $\mathcal{A}$ as:

$$\text{Adv}^{\text{TP}}_{\mathcal{C}}(\mathcal{A}) = Pr[\text{Exp}^{\text{TP}}_{\mathcal{C}}(\mathcal{A}) : \hat{k} = k]$$

Definition 29. ***(ϵ-t-Threshold probing secure).*** *$0 < \epsilon < 1$, $0 < t \leq |\mathcal{W}|$. We say a circuit $\mathcal{C}$ is ϵ-t-Threshold probing secure if for all adversaries $\mathcal{A}$ we have:*
$Adv_{\mathcal{C}}^{TP}(\mathcal{A}) \leq \epsilon$

Lemma 9. *([DDF] Lemma 6). If a circuit C is ϵ-t-Threshold probing secure it is also ϵ-$\frac{t+1}{2|\mathcal{W}|}$-Random probing secure with probability $\geq 1 - q \cdot \exp\left(\frac{|\mathcal{W}| \cdot p}{3}\right)$*

Proof. We give a reduction from the p-Random probing model to the t-Threshold probing model. Let $\mathcal{A}$ be an adversary in the p-Random probing model with $Adv_{\mathcal{C}}^{\mathrm{RP}}(\mathcal{A}) = \epsilon$. We will construct an adversary $\mathcal{B}$ in the t-Threshold probing model with the same advantage as $\mathcal{A}$ with high probability. $\mathcal{B}$ operates by a set of probes randomly according to how the random probing model acts. With high probability this set will be of size less than or equal to t. Formally $\mathcal{P} \leftarrow \{w \text{ w.p. } p, \perp \text{ otherwise }\}$. Then $\mathcal{B}$ requests the values of the probes in $\mathcal{P}$ and gives them to $\mathcal{A}$. $\mathcal{B}$ then returns the same guess as $\mathcal{A}$.

Provided $|\mathcal{P}| \leq t$ $\mathcal{A}$ and $\mathcal{B}$ make the same guess so their advantages are the same. Now we just need to show the probability of this occurring is exponentially small.

Let X be the r.v. that counts the size of the set $\mathcal{P}$. We want to calculate the probability that X is greater than t. Note that $\mathbb{E}(X) = |\mathcal{W}| \cdot p$. And we write:

$$Pr[|\mathcal{P}| > t] = Pr[X > t] = Pr[X > 2\mathbb{E}[X] - 1] = Pr[X \geq 2\mathbb{E}[X]]$$

With the last equality because X takes values in the integers only.

Notice that $X = \sum_{i=0}^{|\mathcal{W}|} W_i$ where W_i is 1 if the i-th wire leaks and 0 otherwise. It is well known that from the chernoff bound ([DP09] Theorem 1.1) we have $Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp(\frac{-\delta^2 \mathbb{E}[X]}{2+\delta})$ for $\delta \geq 0$. By taking $\delta = 1$ we obtain:

$$Pr[|\mathcal{P}| > t] \leq \exp\left(\frac{|\mathcal{W}| \cdot p}{3}\right)$$

And therefore since each invocation of $|\mathcal{P}| > t$ is bounded above by $q * \exp\left(\frac{|\mathcal{W}| \cdot p}{3}\right)$.

Therefore with and exponentially small probability the reduction holds completing the proof. □

From here on we will consider methods of turning an insecure circuit into a secure one in the Threshold probing model and use lemmas 8 and 9 combined to obtain security in the noisy leakage model. Reminder the noisy leakage model by virtue of being closer to reality gives a better judge of how secure the circuit really is but doesn't imply security as it assumes independent leakage, which isn't the case.

4.3 Simulation

Definition 30. ***(Probe Set).*** *([Cas] Definition 5). A set of probes P in a gadget G is a subset of it's wires $\mathcal{W}$, we denote by $G_P((x))$ the tuple containing the values of the wires in P with inputs $\mathbf{x}$.*

Definition 31. ***(Safe probe set).*** *([Cas] Definition 6). A set of probes P is a gadget G is called safe if the distribution of $G_P(Enc(x^1), \ldots, Enc(x^m))$ is the save for every value of $(x^1, \ldots, x^m)$. If a set of probes is not safe it is a successful attack in the sense that information can be extracted.*

This last point warrants some explanation. Specifically if the distribution of the probes values depends on the values of the inputs then since the keys shares (or values that depend on the key) will be inputs to the gadget information about the key (beyond what is available in the black box model) can be extracted and the adversary can use this to extract the key.

Corollary 3. *([Cas] Definition 7). If every set of probes $\mathcal{P}$ of size at most t in a circuit is safe then the circuit is $\frac{1}{|\mathcal{K}|}$-t-Threshold probing secure.*

Proof. As each set of probes is safe any probes an adversary requests is independent from the key. This implies that no information can be extracted from the probes and the best attack left is guessing at random. Since the key is sampled uniformly the best guess is also sampling uniformly which has a probability of $\frac{1}{|\mathcal{K}|}$ of succeeding. □

A couple of small comments on notation, from this point on will will refer to a gadget as being t-Threshold probing secure if every set of probes of size at most t is safe (note the drop of the ϵ, parameter). This is distinct from the gadget being ϵ-t-Threshold probing secure but as we will see later is equivalent for suitable ϵ, see Definition 33 and Proposition 3. The property we care about the inability of the adversary to obtain the key through the probes. Also while we are attempting to ensure security of the key k when considering an isolated gadget we don't know which inputs the shares the key $\mathbf{k}$ will appear, therefore we will assume all inputs and consider security of the inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k$.

Definition 32. ***(Simulatability).*** *([Cas] Definition 10). Let P be a set of t probes in a gadget G. Let $\mathcal{I}' \subset \mathcal{I}$ be a set of k input shares. A simulator is a randomized function $\mathcal{S} : \mathbb{F}^k \to \mathbb{F}^s$. The set of probes P can be simulated with the set of inputs $\mathcal{I}'$ if and only if there exists a simulator $\mathcal{S}$ such that for any value of the input shares $\mathbf{x}$ of G, the distributions $G_P(\mathbf{x})$ and $\mathcal{S}(\mathbf{x}_{|\mathcal{I}'})$ are equal, where the probability is over the randomness in G and $\mathcal{S}$ and $\mathbf{x}_{|\mathcal{I}'}$ denotes the restriction of $\mathbf{x}$ to the elements designated by $\mathcal{I}'$. The simulatablility set of P denoted $S^G(P)$ is the smallest set of input shares of G such that P can be simulated with $G^G(P)$.*

Proposition 2. *([Cas] Proposition 1). If a set of probes $\mathcal{P}$ in a d-shares gadget G can be simulated with a set of inputs $\mathcal{I}'$ that contain at most $d-1$ shares of each input sharing then $\mathcal{P}$ is safe.*

Proof. For any input sharing $\mathbf{x}^i$ of G let $\mathbf{x}^i_{|\mathcal{I}'}$ be the shares of $\mathbf{x}^i$ that belong to $\mathbf{x}_{|\mathcal{I}'}$, hence $|\mathbf{x}^i_{|\mathcal{I}'} \leq d-1$. We know from definition 14 that $\mathbf{x}^i_{|\mathcal{I}'}$ is independent of x^i (as $\mathbf{x}_{|\mathcal{I}'|}$ has less than d elements) and the same holds for $G_P(\mathbf{x})$, therefore P is safe. □

Now we state a game based security notion of t-Threshold Probing security based on simulatability notions from [BGR18]. We state two experiments ExpReal and ExpSim. In both experiments the adversary selects a set of probes of size at most t as well as inputs to the circuit. In the real experiment the circuit is executed and the values of the probes are returned. In the simulated experiment the simulator is given the probes to simulate but not the inputs. The simulator wins if it outputs the same distribution of $(v_1, \ldots, v_s)$ as the circuit. Let $\mathcal{C}(\mathbf{x}_1, \ldots, \mathbf{x}_n)_{\mathcal{P}}$ be the values in the probes when executing the circuit on inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Algorithm 4.5: ExpReal

$(\mathcal{P}, x_1, \ldots, x_n) \xleftarrow{\$} \mathcal{A}()$
$\mathbf{x}_i \xleftarrow{\$} \text{Enc}(x_i)$
$(v_1, \ldots, v_t) \xleftarrow{\$} \mathcal{C}(\mathbf{x}_1, \ldots, \mathbf{x}_n)_{\mathcal{P}}$
return $(v_1, \ldots, v_t)$

Algorithm 4.6: ExpSim

$(\mathcal{P}, x_1, \ldots, x_n) \xleftarrow{\$} \mathcal{A}()$
$(v_1, \ldots, v_t) \xleftarrow{\$} S(\mathcal{P})$
return $(v_1, \ldots, v_t)$

Definition 33. ***(t-Threshold probing secure).*** *([BGR18] Proposition 1) (Note the drop in ϵ parameter). Let $0 < t \leq |\mathcal{W}|$ We say a circuit $\mathcal{C}$ is t-Threshold probing secure if for every adversary $\mathcal{A}$ that outputs a set of probes of size at most t there exists a simulator S that wins game 8.*

Proposition 3. *If a circuit $\mathcal{C}$ is t-Threshold probing secure secure then the circuit is ϵ-t-Threshold probing secure for $\epsilon = \frac{1}{|\mathcal{K}|}$.*

Proof. If for every adversary there exists a simulator that wins then every set of probes can be simulated. Since the simulator t has no shares of each input sharing by Proposition 2 every set of probes the adversary can choose is safe and therefore by Corollary 3 is ϵ-t-Threshold probing secure. □

Note that simulators don't require any shares of the encoding of x_i as x_i is encoded uniformly so each $\mathbf{x}_{i,j}$ (where $\mathbf{x}_{i,j}$ is the j-th share of the i-th input) is distributed uniformly at random.

So now the task of checking security is equivalent to the task of checking simulatablility for sets of probes. The naive method would be to check every set of at most t wires and compute their distribution. This is infeasible due to scaling problems, the number of sets of probes is $\binom{|\mathcal{W}|}{t}$ where $|\mathcal{W}|$ is typically in $\mathcal{O}(nt^2)$ (where n is the number of non-linear gates [Cas] Page 20), i.e. it grows far to fast to be checked for practical circuits. Additionally the cost of computing the distribution of a set of t elements is $\mathcal{O}(2^k)$ where k is the number of input shares and random gates in the circuit. With direct checking for circuits being infeasible we turn to composition as a solution.

4.4 Composition

Our goal now is define a property of gadgets that implies the gadget is safe *and* transfers across gadget composition, each gadget should be small enough that manually checking a single gadget can be completed circumventing the scaling issue. We motivate our definitions by first stating how to implement affine functions as gadgets and observing security properties obtained by this gadget.

Let $f : \mathbb{F}^n \to \mathbb{F}^m$ be an affine function, i.e. $f(\mathbf{x}) = A \cdot x + \mathbf{b}$ where A is a matrix in $\mathcal{M}_{n\times m}(\mathbb{F})$ and $\mathbf{b} \in \mathbb{F}^m$. Then below we present an algorithm for a gadget of f.

Algorithm 4.7: Sharewise implementation of affine function $f : \mathbb{F}^m \to \mathbb{F}^m$ ([Cas] Algorithm 1)

input : Sharings $\mathbf{x}^1, \ldots, \mathbf{x}^n$ where each $\mathbf{x}^i \in \mathbb{F}^d$
output: Sharings $\mathbf{y}^1 \ldots, \mathbf{y}^m$ where each $\mathbf{y}^i \in \mathbb{F}^d$ and $(y^1, \ldots, y_m) = f(x^1, \ldots, x^n)$
$(\mathbf{y}_1^1, \ldots, \mathbf{y}_1^m) \leftarrow f(\mathbf{x}_1^1, \ldots, \mathbf{x}_1^n)$
for $i = 2$ *to* d **do**
  $(\mathbf{y}_i^1, \ldots, \mathbf{y}_i^m) \leftarrow f(\mathbf{x}_i^1, \ldots, \mathbf{x}_i^n) - f(0, \ldots, 0)$
end
return $(\mathbf{y}^1, \ldots, \mathbf{y}^m)$

Sharewise gadgets (Algorithm 10) can be shown to be correct by simply writing out that $x^i = \mathbf{x}_1^i + \ldots \mathbf{x}_d^i$ and $y^i = \mathbf{y}_1^i + \ldots \mathbf{y}_d^i$ and using the fact that matrix multiplication is linear. We have the nice property that sharewise gadgets are trivially secure with no randomness overhead. Also since the composition of sharewise gadgets is also a sharewise gadget the security transfers across gadget composition.

Proposition 4. *Let G be a sharewise gadget with d shares. G is $(d-1)$-Threshold probing secure.*

Proof. Let $\mathcal{P}$ be a set of probes in G of size at most $d-1$. let $G_1, \ldots, G_d$ be the subcircuits of G each G_i induced by share i. As $|P| \leq d-1$ there exists a G_i with no probe in the subcircuit and therefore the probes in $\mathcal{P}$ are clearly independent of share i. Therefore the probes at depend of a set of shares of x of size at most $d-1$ which we know is independent of x and therefore the probes are independent of x. □

Unfortunately we can't build our whole circuit out of affine functions and trivially obtain security as a composition of affine functions is also affine and thus little use for encryption. We therefore use the property that the i-th input share never interacts with the j-th input share for $i \neq j$ as a guide for our composition properties.

4.4.1 Non composibility of security

Before we jump the gun and start defining composition properties it is worth checking that security in gadgets doesn't trivially compose without any other conditions, we can see that it doesn't by a simple example. We introduce a secure multiplication gadget and compose it with a sharewise gadget (which we have already seen is secure). Then we show that this new gadget is insecure. This specific example is taken from [Cas].

Algorithm 4.8: ISW Multiplication algorithm with d shares ([ISW03] Page 470, [Cas] Algorithm 2)

input : sharings $\mathbf{x}, \mathbf{y}$
output: $\mathbf{z}$ such that $z = x \cdot y$
for $i = 1$ *to* d **do**
  for $j = i + 1$ *to* d **do**
    $r_{ij} \xleftarrow{\$} \mathbb{F}$
    $r_{ji} \leftarrow -r_{ij}$
  end
end
for $i = 1$ *to* d **do**
  $\mathbf{z_i} \leftarrow \mathbf{x_i} \cdot \mathbf{y_i}$
  for $j = 1$ *to* $d, i \neq j$ **do**
    $\mathbf{z}_i \leftarrow \mathbf{z}_i + \left(\mathbf{x}_i \cdot \mathbf{y}_j + r_{ij}\right)$
  end
end
return z

First checking that this does indeed compute a valid sharing of z we have:

$$
\begin{aligned}
\sum_{i=1}^{d} \mathbf{z}_i &= \sum_{i=1}^{d} \Big(\mathbf{x}_i \mathbf{y}_j + \sum_{j=1, j\neq i}^{d} \left(\mathbf{x}_i \mathbf{y}_j + r_{ij} \right) \Big) \\
&= \sum_{i=1}^{d} \Big(\mathbf{x}_i \mathbf{y}_j + \sum_{j=1, j\neq i}^{d} \mathbf{x}_i \mathbf{x}_j + \sum_{j=1, j\neq i}^{d} r_{ij} \Big) \\
&= \sum_{i=1}^{d} \Big(\sum_{j=1}^{d} \mathbf{x}_i \mathbf{y}_j + \sum_{j=1, j\neq i}^{d} r_{ij} \Big) \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbf{x}_i \mathbf{y}_j + \sum_{i=1}^{d} \sum_{j=1, j\neq i}^{d} r_{ij} \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbf{x}_i \mathbf{y}_j \\
&= \left(\mathbf{x}_1 + \cdots + \mathbf{x}_d \right) \cdot \left(\mathbf{y}_1 + \cdots + \mathbf{y}_d \right) \\
&= x \cdot y
\end{aligned}
$$

Proposition 5. *ISW Multiplication (Algorithm 11) with d shares is $(d-1)$-Threshold probing secure.*

Proof. For each probe the most shares of x that the probe can depend on is 1 likewise for y, Therefore at most $|\mathcal{Y}|$ shares of each input are required to trivially simulate the probes by running the computation. If $|\mathcal{Y}| \leq d - 1$ then the simulatability set is of size less than d and by Proposition 2 and 3 we complete the proof. □

Proposition 6. *The sequential composition of t-Threshold probing secure gadgets G_1 and G_2 isn't necessarily t-Threshold probing secure.*

Proof. Let $\mathbb{F} = \mathbb{F}_2$, notice in $\mathbb{F}_2$ squaring is linear as for $f(x) = x^2$ we have that $f(x+y) = (x+y)^2 = x^2 + y^2 + 2xy \equiv x^2 + y^2 = f(x) + f(y) \pmod 2$ and $f(ax) = a^2x^2 \equiv ax^2 = af(x) \pmod 2$.

Let G_1 be sharewise implementation of squaring and G_2 be the ISW multiplication gadget both with d shares. The circuit computing x^3 implemented by first squaring x with G_1 then multiplying x^2 by x is the sequential composition of two $(d-1)$-Threshold Probing secure gadgets yet is not $(d-1)$-Threshold probing secure itself.

By probing the intermediate variable in the ISW multiplication $\mathbf{x}_i \cdot \mathbf{y}_j = \mathbf{x}_i \cdot \mathbf{x}_j^2$ we can probe multiple shares of x at the same time, this works because both inputs to the ISW multiplication dependent on $\mathbf{x}$. Setting the i-th probe to be the intermediate value $\mathbf{x}_i \cdot \mathbf{x}_{i+1}^2$ we have a set of $|P| \leq d-1$ probes that depend on all shares of x and therefore the composition isn't $d-1$ probing secure. □

So we now move to building composition properties.

4.4.2 PINI Composition

Definition 34. ***(NI).*** *([Bar+16] Lemma 2, [Cas] Definition 11). A d-shares gadget G is t-non-interferent (t-NI) if any set of probes P such that $|P| \leq t$ can be simulated using at most t shares of each input sharing.*

This property is motivated by the affine implementation, where if there are at most $d-1$ probes then there must be at least one share of each input sharing that the probes aren't dependent on (i.e. independent of). And therefore the probes are independent of the secret. It provides security but not composition so acts as a step to a stronger property we require.

Proposition 7. *([Cas] Proposition 4). Any t-NI gadget with at least $t+1$ shares is t-Threshold probing secure.*

Proof. Any set of at most t probes can be simulated with t shares of each input sharing and is therefore safe by Proposition 1 □

To define a compositional property we look back to the example of where composition failed and notice that the problem was that shares of different indexes of different inputs can be sampled with one probe. While this is secure if the inputs are independent, if not it breaks security. Therefore we force that the i-th share of each input shouldn't be able to be probed with the j-th share of any other input (not just the j-th share of itself).

Definition 35. ***(PINI).*** *([CS18] Definition 5, [Cas] Definition 12). A d-shares gadget G is t-probe-isolating non-interferent if, for any set $A \subset \{1, \ldots, d\}$ and any set of probes P such that $|A| + |P| \leq t$, there exists a set $B \subset \{1, \ldots, d\}$ with $|B| \leq |P|$ such that the probes P and all output shares of G with index in A can be simulated by the inputs of G with index in $A \cup B$.*

Clearly and t-PINI gadget is also t-NI as by taking $A = \emptyset$ we obtain the condition for G being t-NI. Also any t-PINI gadget is t'-PINI for $t' \leq t$ as the sets A and P in the t'-PINI condition will be smaller than in the t-PINI condition meaning that all the conditions are satisfied.

Corollary 4. *([CS18] Proposition 2). Any t-PINI gadget with at least $t+1$ shares is t-Threshold probing secure.*

Proof. Any t-PINI gadget with at least $t+1$ shares is a t-NI gadget with $t+1$ shares and therefore is secure by Proposition 7 □

Theorem 5. *([CS18] Proposition 3, [Cas] Theorem 1). The parallel composition of t-PINI gadgets is t-PINI.*

Proof. Let G_1 and G_2 be gadgets and let $G = G_1 \| G_2$. Let $\mathcal{P}$ be the set of probes in G and $\mathcal{P}_1 \subset \mathcal{P}$ be the subset of probes in G_1 likewise $\mathcal{P}_2 \subset \mathcal{P}$ be the probes in G_2. Let $A \subset \{1, \ldots, d\}$ be such that $|A| + |\mathcal{P}| \leq t$. Clearly $|A| + |\mathcal{P}_1| \leq t$ so there exists a set $B_1 \leq |\mathcal{P}_1| \leq |\mathcal{P}|$ such that the probes $\mathcal{P}_1$ and outputs of G_1 with inputs in A can be simulated by the inputs of G with index in $A \cup B_1$. Likewise there exists a set B_2 by the same logic. By taking $B = B_1 \cap B_2$ we can simulate all probes in $\mathcal{P}$ and all outputs of G with index in A. Now $|B| = |B_1 \cup B_2| \leq |\mathcal{P}_1 \cup \mathcal{P}_2| = |P|$ as $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ Therefore G is t-PINI. □

Theorem 6. *([CS18] Proposition 3, [Cas] Theorem 1). The sequential composition of t-PINI gadgets is t-PINI.*

Proof. Let G_1 and G_2 be gadgets and let $G = G_1 \circ G_2$. Let $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{P}$ be as defined previously. As G_2 is t-PINI by using the simulator for the probes $\mathcal{P}_2$ gives a set B_2 such that $|B_2| \leq |\mathcal{P}_2|$ and the probes in $\mathcal{P}_2$ and outputs of G_2 in A (which are the outputs of G) can be simulated with inputs in $A \cup B_2$. Now using the t-PINI simulator of G_1 we to simulate the probes $\mathcal{P}_1$ and the outputs in $A \cup B_2$ gives B_1 with $|B_1| \leq |\mathcal{P}_1|$ such that the probes in $\mathcal{P}_1$ and all output shares of G_1 with index in $A \cup B_2$ can

be simulated with the input shares of G_1 with index in $A \cup B2$, can be simulated by input shares of G_1 (which are the inputs shares of G) with index in $A \cup B_1 \cup B_2$. Now let $B = B_1 \cap B_2$ and notice that $|B| = |B_1 \cap B_2| \leq |B_1| + |B_2| \leq |\mathcal{P}_1| + |\mathcal{P}_2| = |P|$ which means that G is t-PINI, concluding the proof. □

Corollary 5. *A masking compiler operating by replacing logic gates by t-PINI gadgets each with $t+1$ shares produces a circuit that is t-Probing secure*

Proof. The resulting circuit will be the parallel and sequential composition of multiple t-PINI gadgets which by Theorems 5 and 6 is also t-PINI and by corollary 3.5.1 is t-Threshold probing secure. □

So now all we need are gadgets that satisfy the PINI property and we can compile our circuits. It is trivial to check that sharewise gadgets satisfy the PINI property.

Proposition 8. *([Cas] Proposition 6). All sharewise gadgets with d shares (those implementing affine functions) are $(d-1)$-PINI.*

Proof. Let G be a sharewise gadget with d shares, P be a set of probes in G and A be a set of output shares. G can be partitioned into d disjoint subcircuits each induced by a unique share. Then at most $|\mathcal{P}|$ of these can have probes in. Let B be the indexes of the subcircuits that have probes in, clearly $|B| \leq |\mathcal{P}|$ and then all probes P and outputs can be simulated by shares in $B \cup A$. □

Now we need to build a multiplication gadget that is PINI. Thinking back to the ISW multiplication the reason it isn't PINI is the shares $\mathbf{x}_i$ interacting directly with $\mathbf{y}_j$ for $i \neq j$ so all we need to do is mask this step with some randomness and obtain the following algorithm.

Algorithm 4.9: PINI multiplication gadget with d shares ([CS18] Algorithm 2, [Cas] Algorithm 4)

input : Sharings $\mathbf{x}, \mathbf{y}$
output: $\mathbf{z}$ such that $z = x \cdot y$
for $i = 1$ *to* d **do**
  for $j = i + 1$ *to* d **do**
    $r_{ij} \xleftarrow{\$} \mathbb{F}$
    $r_{ji} \leftarrow -r_{ij}$
  end
end
for $i = 1$ *to* d **do**
  $\mathbf{z}_i \leftarrow \mathbf{x_i} \cdot \mathbf{y_i}$
  for $j = 1$ *to* $d, i \neq j$ **do**
    $\mathbf{z}_i \leftarrow \mathbf{z}_i + \big((\mathbf{x}_i + 1)r_{ij} + \mathbf{x}_i(\mathbf{y}_j - r_{ij})\big)$
  end
end
return $\mathbf{z}$

First to check the correctness note that $(\mathbf{x}_i + 1)r_{ij} + \mathbf{x}_i(\mathbf{y}_j - r_{ij}) = \mathbf{x}_i \cdot \mathbf{y}_j + r_{ij}$ and then the proof is the same as for the ISW multiplication. Now to check security we have:

Proposition 9. *([Cas] Proposition 7). Algorithm 12 is $(d-1)$-Threshold probing secure.*

Proof. Given a set of probes $\mathcal{P}$ and outputs A to simulate we will build a set B such that $|B| \leq |\mathcal{P}|$ that allows simulation of $\mathcal{P}$ and A when knowing the inputs in $A \cup B$. We first observe what is required to simulate each intermediate term in the computation. Let $\mathcal{I}' = A \cup B$ then we list the knowledge required to simulate intermediate terms:

- Probes on $\mathbf{x}_i \cdot \mathbf{y}_i$ require $i \in \mathcal{I}'$.
- Probes on r_{ij} require no outside knowledge.
- Probes on $\mathbf{x}_i + 1$ require $i \in \mathcal{I}'$.
- Probes on $(\mathbf{x}_i + 1)r_{ij}$ require $i \in \mathcal{I}'$.

- Probes on $\mathbf{y}_j - r_{ij}$ require no knowledge as it can be simulated as a uniformly random variable.
- Probes on $\mathbf{x}_i(\mathbf{y}_j - r_{ij})$ require $i \in \mathcal{I}'$.
- Probes on $(\mathbf{x}_i + 1)r_{ij} + \mathbf{x}_i(\mathbf{y}_j - r_{ij}) = \mathbf{x}_i \cdot \mathbf{y}_j + r_{ij}$ require no knowledge as it can be simulated as a uniformly random variable.
- Probes on $\mathbf{z}_i = \mathbf{z}_i + ((\mathbf{x}_i + 1)r_{ij} + \mathbf{x}_i(\mathbf{y}_j - r_{ij})) = \mathbf{z}_i + \mathbf{x}_i\mathbf{y}_j + r_ij$ require no knowledge as it can be simulated as a uniformly random variable.

Therefore for each probe in $\mathcal{P}$ at most 1 index needs to be added to B, also simulating the outputs is equivalent to simulating $\mathbf{z}_i$ which we have already considered and require no inputs additional inputs. Therefore there exists a set B with $|B| \leq |\mathcal{P}|$ allowing simulation by building B by iterating over the probes and adding the index (if any) required to simulate that probe. Which completes the proof. □

The usage of this gadget requires $\frac{d(d-1)}{2}$ fresh random variables and introduces and has $\mathcal{O}(d^2)$ extra multiplications and additions. There are other algorithms that can be used to implement multiplication, for example in [CS19] they introduce a multiplication gadget with better performance for larger number of shares.

When considering electrical circuits we are considering circuits over the field $\mathbb{F}_2$, we will recap how any logic gate can be built from the operations of addition and multiplication and therefore that we can mask any circuit with PINI gadgets.

- x AND $y = x \times_{\mathbb{F}_2} y$
- x XOR $y = x +_{\mathbb{F}_2} y$
- x OR $y = (x$ XOR $y) +_{\mathbb{F}_2} (x$ AND $y)$
- NOT $x = 1 +_{\mathbb{F}_2} x$

A small point worth noting is that NOT can be implemented much more efficiently by simply inverting one share, this is because in $\mathbb{F}_2$ a variable x is equivalent to the parity of it's shares $\mathbf{x}$ so to negate x simply flipping the parity of the shares (by negating one share) does the job.

The PINI property defines one method of composing gadgets, it is worth noting that there are others. We will now briefly detail several for completeness and offer performance comparisons.

4.4.3 ISW Composition

The first composition strategy made use of the ISW multiplication gadget (algorithm 11) and sharewise gadgets to compile circuits. This is not particularly efficient to be t-Threshold probing secure it requires $d = 2t + 1$ shares. (Compared to PINI which requires $d = t + 1$ shares). This can be seen as while the ISW multiplication with d shares isn't $(d-1)$-PINI it is $\frac{d-1}{2}$-PINI.

Proposition 10. *The ISW Multiplication gadget (11) with d shares is $\frac{d-1}{2}$-PINI.*

Proof. Let G be the ISW multiplication gadget with d shares, $\mathcal{P}$ a set of probes in G and A a set out output shares of G such that $|\mathcal{P}| + |A| \leq \frac{d-1}{2}$. Each output in A can be simulated with no knowledge of the inputs as it is distributed uniformly at random. Each probe in $\mathcal{P}$ requires at most 2 input shares to be added to B (when probing $\mathbf{x}_i \cdot \mathbf{y}_j$, so $|B| \leq 2|\mathcal{P}| \leq 2 \cdot \frac{d-1}{2} = d - 1$. Therefore we have that ISW is $\frac{d-1}{2}$-PINI. □

Now since a sharewise gadget with d shares is $(d-1)$-PINI (Proposition 8) it is also clearly $\frac{d-1}{2}$-PINI. Therefore compiling circuits with the ISW composition strategy produces a circuit that is $\frac{d-1}{2}$-PINI (Theorems 6 and 5) which then implies that the circuit is $\frac{d-1}{2}$-Threshold probing secure (Corollary 4).

4.4.4 Refreshed Multiplication

Another strategy for composition comes from the observation that the ISW multiplication (Algorithm 11) is secure as long as the inputs are independent. So by re-randomizing or "refreshing" one of the inputs to the multiplication gadget when needed we can compose these gadgets. Which brings us to the question of how to refresh shares, fortunately we already have an answer, simply multiply by 1. More specifically we use the ISW multiplication again where one of the inputs is the constant sharing of $1 = (1 + 0 + \cdots + 0)$. Formally we define a new multiplication gadget as:

Algorithm 4.10: Refreshed Multiplication gadget with d shares

input : Sharings $\mathbf{x}, \mathbf{y}$
output: $\mathbf{z}$ such that $z = x \cdot y$
$\mathbf{o} \leftarrow (1, 0, \ldots, 0)$ `// Constant sharing of 1`
$\mathbf{x}' \leftarrow (0, \ldots, 0)$ `// Refreshed shares of input` x
for $i = 1$ *to* d **do**
  for $j = i + 1$ *to* d **do**
    $r^0_{ij} \xleftarrow{\$} \mathbb{F}$; $r^0_{ji} \leftarrow -r^0_{ij}$
    $r^1_{ij} \xleftarrow{\$} \mathbb{F}$; $r^1_{ji} \leftarrow -r^1_{ij}$
  end
end
`// Multiply x by o and store into x'`
for $i = 1$ *to* d **do**
  $\mathbf{x}'_i \leftarrow \mathbf{x}_i \cdot \mathbf{o}_i$
  for $j = 1$ *to* $d, i \neq j$ **do**
    $\mathbf{x}'_i \leftarrow \mathbf{x}'_i + (\mathbf{x}_i \cdot \mathbf{o}_j + r^0_{ij})$
  end
end
`// Multiply x' by y and store into z`
for $i = 1$ *to* d **do**
  $\mathbf{z}'_i \leftarrow \mathbf{x}'_i \cdot \mathbf{y}_i$
  for $j = 1$ *to* $d, i \neq j$ **do**
    $\mathbf{z}'_i \leftarrow \mathbf{z}_i + (\mathbf{x}'_i \cdot \mathbf{y}_j + r^1_{ij})$
  end
end
return z

We chose to write out explicitly rather than calling the ISW multiplication twice so that there is no ambiguity in variable names when discussing proofs.

Proposition 11. *Refreshed Multiplication (Algorithm 13) with d shares is $(d-1)$-PINI*

Proof. Let G be the gadget that implements Algorithm 13 with d shares, let P be a set of probes in G and A be a set of output share indexes such that $|P| + |A| \leq d - 1$. For each output index simulating the share $\mathbf{z}_i$ requires no input shares from $\mathbf{x}$ or $\mathbf{y}$ as each $\mathbf{z}_i$ is distributed uniformly at random because it is the sum with a uniformly random variable in r^1. Also $\mathbf{x}'_i$ is independent of $\mathbf{x}_i$ because of similar reasoning. Therefore simply by checking each intermediate variable the most share indices that are required to simulate a probe in P is 1. Building B from P by adding the share index required to simulate the i-th probe (if any) to B for each i we have $|B| \leq |P|$ satisfying the PINI condition. □

In [Cas] Lemma 3 a more detailed proof is presented that shows a PINI refreshed multiplication gadget can be constructed from any multiplication gadget and refresh gadget so long as certain properties are met by the multiplication gadget and refresh gadget.

4.4.5 Comparison

We now present a comparison of the composition schemes we have seen previously. Table 4.4.5 is a table that compares each composition strategy and the performance overhead that they incur to achieve t-Threshold probing security.

Composition Strategy	Shares	Randomness	Add & Sub	Multiplication
PINI (Algorithm 12)	$d = t + 1$	$\frac{d(d-1)}{2}$	$\frac{9d(d-1)}{2}$	$2d(d-1)$
ISW (Algorithm 11)	$d = 2t + 1$	$\frac{d(d-1)}{2}$	$\frac{5d(d-1)}{2}$	d^2
Refreshed (Algorithm 13)	$d = t + 1$	$d(d-1)$	$5d^2 - 3d$	$2d^2$

This is not a complete list, in [Cas] they introduce an alternative multiplication gadget that uses less randomness in exchange for more multiplications. And in [Bel+16] they propose a secure multiplication gadget that uses $d + \frac{d^2}{4}$ random bits, outperforming PINI for $d > 6$. There are also other methods of refreshing sharings that outperform the ISW refresh gadget, in [Bat+16] and [Cas] other refresh gadgets are proposed with randomness usage both in $\mathcal{O}(d \log(d))$. This is achieved by making use of a recursive refresh algorithm.

It is important to note the existence of automatic verification tools to check security in the threshold probing model as well as the PINI and NI properties. maskVerif [Cryb] can automatically check Threshold probing security as well as NI and a stronger property called Strong Non Interference (SNI). IronMask [Crya] can check NI, SNI and PINI properties as well as directly checking Random probing security. It is also important to note that the previous composition strategies are only heuristics and achieve their security by being overly cautious. In the next section it is show how to reduce the t-Threshold probing security to a linear algebra problem. This is done through a sequence of games that are equivalent and finally converting to linear algebra.

4.5 Tight Private Circuits

The Composition properties define several methods of securing a circuit, however none of them are optimal in the sense that more randomness, multiplications or additions are used than required. It is however possible to reduce the threshold probing security even further. Through a series of games the threshold probing security can be reduced to a linear algebra problem which can then be solved producing a tight circuit. By tight it is meant that the number of probes is one less than the number of shares, it is clear that better cannot be achieved in the threshold probing model as if the number of probes t is equal to the number of shares d then trivially probing all shares of an input breaks the security. As a brief outline There are 4 games we will consider, Game 0 is the threshold probing game based on simulatability (Algorithm 8), Game 1 restricts the probes the adversary can request to the inputs and outputs of gadgets, Game 2 flattens the circuit by replacing outputs of multiplications and refresh gadgets with fresh random sharings, Game 3 restricts the probe set further to only the inputs of multiplication gadgets. After each game is introduced equivalence will be shown to the previous. Finally Game 3 will be translated into a linear algebra problem involving the intersection of the images of some matrices. This sequence is described with full detail in [BGR18].

Before the games and reductions are presented the scope is limited somewhat. The field the circuits are working over is set to $\mathbb{F}_2$. The circuits are compiled using the ISW multiplication gadget (Algorithm 11), sharewise gadgets f that are of the form $f : \mathbb{F}_2^2 \to \mathbb{F}$ or $f : \mathbb{F}_2 \to \mathbb{F}$ and refresh gadgets. This choice essentially means that the only logic gates that can appear in a circuit are AND (ISW Multiplication), XOR ($f(x, y) = x \oplus y$), and NOT ($f(x) = 1 \oplus x$) logic gates. For this section let $\mathcal{C}$ be a masked circuit with d shares. Let the inputs to $\mathcal{C}$ be $\mathbf{x}^1, \ldots, \mathbf{x}^n$ where each $\mathbf{x}^i = (\mathbf{x}^i_1, \ldots, \mathbf{x}^i_d) \in \mathbb{F}^d$ such that $\mathbf{x}^j_i \in \mathbb{F}_2$ is the i-th share of the j-th input. Let $\mathbf{x}_i \in \mathbb{F}_d^n$ denote the vector of the i-th share of each input, i.e. $\mathbf{x}_i = (\mathbf{x}^1_i, \ldots, \mathbf{x}^n_i)$. Also as the circuit $\mathcal{C}$ is made of composition of gadgets we assign a unique index to each gadget in $\mathcal{C}$, Let $\mathcal{G}$ be the set of gadget indices. Since the circuit is only made of multiplication, sharewise (arithmetic) and refresh gadgets the set $\mathcal{G}$ can be split into 3 disjoint subsets $\mathcal{G}_m, \mathcal{G}_a, \mathcal{G}_r$ which have the indices of the multiplication, arithmetic and refresh gadgets respectively and $\mathcal{G} = \mathcal{G}_m \cup \mathcal{G}_a \cup \mathcal{G}_r$. Now let $g \in \mathcal{G}$, $\mathcal{I}_g$ and $\mathcal{J}_g$ denote the indices in $\mathcal{W}$ of the (left and right) input wires to gadget g. If the gadget is a refresh or NOT gadget then $\mathcal{J}_g = \emptyset$. Also the set $\mathcal{O}_g$ denotes the set of output wire indices in $\mathcal{W}$ of the gadget g.

4.5.1 Game 1

Now Game 1 is presented. In Game 1 the set of probes $\mathcal{P}$ an adversary can request is limited to be: An input of a refresh gadget; A pair of inputs of a multiplication gadget (from different arguments); An input or output of an arithmetic gadget.

Algorithm 4.11: Game 1: ExpReal($\mathcal{A}, \mathcal{C}$)

$(\mathcal{P}', x^1, \ldots, x^n) \xleftarrow{\$} \mathcal{A}()$
$\mathbf{x}^i \xleftarrow{\$} \text{Enc}(x^i)$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{C}(\mathbf{x}^1, \ldots, \mathbf{x}^n)_{\mathcal{P}'}$
return $(v_1, \ldots, v_q)$

Algorithm 4.12: Game 1: ExpSim($\mathcal{A}, \mathcal{C}, \mathcal{S}$)

$(\mathcal{P}', x^1, \ldots, x^n) \xleftarrow{\$} \mathcal{A}()$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{S}(\mathcal{P}')$
return $(v_1, \ldots, v_q)$

Where $\mathcal{P}'$ must be of the form $\mathcal{P}' = \mathcal{P}'_r \cup \mathcal{P}'_m \cup \mathcal{P}'_a$ with $t = |\mathcal{P}'_r| + |\mathcal{P}'_m| + |\mathcal{P}'_a|$ and

$$\mathcal{P}'_r \subset \bigcup_{g \in \mathcal{G}_r} \mathcal{I}_g$$
$$\mathcal{P}'_m \subset \bigcup_{g \in \mathcal{G}_m} \mathcal{I}_g \times \mathcal{J}_g$$
$$\mathcal{P}'_a \subset \bigcup_{g \in \mathcal{G}_a} \mathcal{I}_g \bigcup_{g \in \mathcal{G}_a} \mathcal{J}_g \bigcup_{g \in \mathcal{G}_a} \mathcal{O}_g$$

The dimension q of the returned values $(v_1, \ldots, v_q)$ then is equal to $q = 2 \cdot |\mathcal{P}'_m| + |\mathcal{P}'_r| + |\mathcal{P}'_a|$. A simulator $\mathcal{S}$ wins Game 1 iff the tuple $(v_1, \ldots, v_q)$ has the same distribution in both experiments.

Proposition 12. *([BGR18] Proposition 4). A circuit $\mathcal{C}$ with d shares is $(d-1)$-Threshold probing secure iff for every adversary $\mathcal{A}$ there exists a simulator t that wins Game 1.*

Proof. We will present a reduction to Game 0 (Algorithm 8) which defines t-Threshold probing security. This is done by considering the probes in Game 0 $p \in \mathcal{P}$ and showing that for every form of p that isn't allowed in $\mathcal{P}'$ that probe can be replaced by at most 1 probe in $\mathcal{P}'$. This is an informal argument, for full details see [BGR18] Proposition 4's proof.

Probes in Multiplication Gadgets. Any probes on the inputs of a multiplication gadget in $\mathcal{P}$ remain unchanged in $\mathcal{P}'$. Probes on the output appear uniformly random so can be simulated trivially and therefore can be removed from $\mathcal{P}'$. Let the multiplication gadget's inputs be $\mathbf{a}$ and $\lfloor$. Probes on the cross term $\mathbf{a}_i \cdot \mathbf{b}_j$ can be replaced by the tuple of probes $(\mathbf{a}_i, \mathbf{b}_j)$ in $\mathcal{P}'$. Therefore every probe in a multiplication gadget can be turned into a probe into $\mathcal{P}'_m$ if needed.

Probes in Refresh Gadgets. Any probes on the inputs of a refresh gadget in $\mathcal{P}$ remain unchanged in $\mathcal{P}'$. Outputs appear uniformly random so can be ignored and probes on internal variables can depend on at most 1 input share so can be replaced by a probe on that input share.

Probes in Arithmetic Gadgets. Any probes on the inputs or outputs remain unchanged. If the gadget implements a function $f : \mathbb{F}_2^2 \to \mathbb{F}$ then probes on intermediate values that only contain one input get replaced by probes on that input. Probes on intermediate values that contain both inputs get replaced by probes on the corresponding output.

Therefore given an adversary in Game 0 that requests probes $\mathcal{P}$ we can build an adversary in Game 1 that requests probes $\mathcal{P}'$ (dependent on $\mathcal{P}$) such that simulating $\mathcal{P}$ is equivalent to simulating $\mathcal{P}'$. This means that given an adversary $\mathcal{A}_0$ that wins Game 0 we can build an adversary $\mathcal{A}_1$ that wins Game 1. Also Given $\mathcal{A}_1$ that wins Game 1 the set of probes $\mathcal{P}'$ can be turned back into $\mathcal{P}$ (only the multiplication probes need to be reverted back to internal probes) giving $\mathcal{A}_0$ that wins Game 0. Therefore we have: $\mathcal{C}$ is $(d-1)$-Threshold probing secure $\iff$ For all $\mathcal{A}$ there exists $\mathcal{S}$ that wins Game 0 $\iff$ for every $\mathcal{A}$ there exists $\mathcal{S}$ that wins Game 1. □

4.5.2 Game 2

Now Game 2 is described. Recall in Game 2 the circuit is flattened so that outputs of multiplications and refresh gadgets are replaced with new, uniformly random inputs to the circuit. Specifically flatten

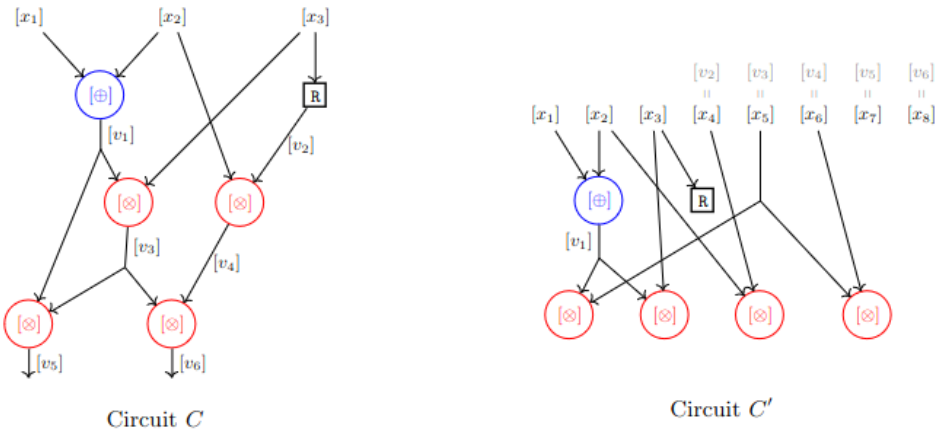

Figure 4.1: Example of the Flatten transform ([BGR18] Figure 7)

is a transform that acts as follows. It takes a circuit $\mathcal{C}$ with n inputs and returns a circuit $\mathcal{C}'$ with $N = n + |\mathcal{G}_m| + |\mathcal{G}_r|$ inputs. The set of gadgets in $\mathcal{C}'$ is the same as in $\mathcal{C}$. Wires that do not originate from a multiplication or refresh gadget remain unchanged. Wires that originate from a refresh or multiplication gadget are replaced with a new input to the circuit (giving the extra inputs $N > n$). Flatten is defined so that the index of a gadget or wire in $\mathcal{C}'$ remains the same as in $\mathcal{C}$. An example of the flatten transform from [BGR18] is below.

Now with the Flatten transform described the second game can be presented.

Algorithm 4.13: Game 2: $\text{ExpReal}(\mathcal{A}, \mathcal{C})$

$\mathcal{C}' \leftarrow \text{Flatten}(\mathcal{C})$
$(\mathcal{P}', x^1, \ldots, x^N) \xleftarrow{\$} \mathcal{A}()$
$\mathbf{x}^i \xleftarrow{\$} \text{Enc}(x^i)$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{C}'(\mathbf{x}_1, \ldots, \mathbf{x}_N)_{\mathcal{P}'}$
return $(v_1, \ldots, v_q)$

Algorithm 4.14: Game 2: $\text{ExpSim}(\mathcal{A}, \mathcal{C}, \mathcal{S})$

$\mathcal{C}' \leftarrow \text{Flatten}(\mathcal{C})$
$(\mathcal{P}', x^1, \ldots, x^N) \xleftarrow{\$} \mathcal{A}()$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{S}(\mathcal{P}')$
return $(v_1, \ldots, v_q)$

The constraints and description of $\mathcal{P}'$ and q remain unchanged from Game 1. The simulator $\mathcal{S}$ wins Game 2 iff the distribution of the tuple $(v_1, \ldots, v_q)$ has the same distribution in both experiments.

Proposition 13. *([BGR18] Proposition 4). A masked circuit $\mathcal{C}$ with d shares is $(d-1)$-Threshold probing secure if and only if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ that wins game 2.*

Proof. We present a reduction to Game 1 (which then implies via 12 Threshold probing security).

First show:

$$\forall \mathcal{A}_2 \exists \mathcal{S}_2 \text{ that wins Game 2} \implies \forall \mathcal{A}_1 \exists \mathcal{S}_1 \text{ that wins Game 1}$$

Given $\mathcal{A}_1$ define $\mathcal{A}_2$ such that $\mathcal{A}_2$ makes the same choice of probes. (Recall Flatten doesn't change the indexing of the circuit). The first n inputs $\mathcal{A}_2$ give match those given by $\mathcal{A}_1$. The additional inputs that $\mathcal{A}_2$ gives are such that they match decoded intermediate values in $\mathcal{C}$. Because the outputs of multiplication and refresh gadgets appear uniformly random and the additional inputs $x^{n+1}, \ldots, x^N$ are encoded with the uniform encoder the execution of $\mathcal{C}$ is equivalent to the execution of $\mathcal{C}'$ on the inputs that $\mathcal{A}_2$ gives. Then if there exists $\mathcal{S}_2$ that simulates the probes in $\mathcal{P}'$ it can also be used in Game 1.

Now show:

$$\forall \mathcal{A}_1 \exists \mathcal{S}_1 \text{ that wins Game 1} \implies \forall \mathcal{A}_2 \exists \mathcal{S}_2 \text{ that wins Game 2}$$

For an adversary $\mathcal{A}_2$ that returns $\mathcal{P}'$ and $(x^1, \ldots, x^N)$ define $\mathcal{A}_1$ such that $\mathcal{A}_1$ returns the same probe set and the truncated input of $\mathcal{A}_2$. By assumption there exists $\mathcal{S}_1$ that wins Game 1 and by the above logic $\mathcal{S}_1$ can be used to simulate the probes in Game 2 as well. □

4.5.3 Game 3

Now Game 3 can be presented. Recall in Game 3 the probes are limited to tuples of inputs on the multiplication gadgets. The experiments are as follows:

Algorithm 4.15: Game 3: ExpReal$(\mathcal{A}, \mathcal{C})$

$\mathcal{C}' \leftarrow \text{Flatten}(\mathcal{C})$
$(\mathcal{P}'', x^1, \ldots, x^N) \xleftarrow{\$} \mathcal{A}()$
$\mathbf{x}^i \xleftarrow{\$} \text{Enc}(x^i)$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{C}'(\mathbf{x}_1, \ldots, \mathbf{x}_N)_{\mathcal{P}'}$
return $(v_1, \ldots, v_q)$

Algorithm 4.16: Game 3: ExpSim$(\mathcal{A}, \mathcal{C}, \mathcal{S})$

$\mathcal{C}' \leftarrow \text{Flatten}(\mathcal{C})$
$(\mathcal{P}'', x^1, \ldots, x^N) \xleftarrow{\$} \mathcal{A}()$
$(v_1, \ldots, v_q) \xleftarrow{\$} \mathcal{S}(\mathcal{P}')$
return $(v_1, \ldots, v_q)$

Where $\mathcal{P}''$ must be of the form $\mathcal{P}'' = \mathcal{P}'' \subset \cup_{g \in \mathcal{G}_m} \mathcal{I}_g \times \mathcal{J}_g$ with $t = |\mathcal{P}''|$ and
The dimension q of the returned values $(v_1, \ldots, v_q)$ then is equal to $q = 2 \cdot |\mathcal{P}'|$. A simulator $\mathcal{S}$ wins Game 3 iff the tuple $(v_1, \ldots, v_q)$ has the same distribution in both experiments.

Proposition 14. *([BGR18] Proposition 6). A masked circuit $\mathcal{C}$ with d shares is $(d-1)$-Threshold probing secure if and only if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ that wins game 3.*

Proof. We present a reduction to Game 2 (which then implies via 12 and 13 Threshold probing security). For a more formal argument based on linear algebra see [BGR18].

First show:

$$\forall \mathcal{A}_3 \exists \mathcal{S}_3 \text{ that wins Game 3} \implies \forall \mathcal{A}_2 \exists \mathcal{S}_2 \text{ that wins Game 2}$$

This will be done by showing the contrapositive

$$\exists \mathcal{A}_2 \forall \mathcal{S}_2 \text{ loses Game 2} \implies \exists \mathcal{A}_3 \forall \mathcal{S}_3 \text{ loses Game 3}$$

First the set of probes $\mathcal{P}'$ that $\mathcal{A}_2$ returns it must be dependent on at least one of the inputs x^i. Without loss of generality let this input be x^1. Also there must be at least one probe on the input to a multiplication gadget that reveals shares of x^1 otherwise the probes in $\mathcal{P}'$ would form a subset of size less than or equal to $d-1$ of shares of x^1 which is independent of x^1 by definition of the uniform encoder. This is because if the probes on shares of x^1 are all on the inputs to refresh gadgets or inputs & outputs of arithmetic gadgets each probe can only reveal at most one share of x^1 and $|\mathcal{P}'| = d-1$. We will show that given the set of probes $\mathcal{P}'$ how to build $\mathcal{P}''$ that can be returned by $\mathcal{A}_3$ to win against all adversaries. For any probe in $\mathcal{P}'$ on the input to a refresh gadget can be moved to either the input to a arithmetic (sharewise) gadget or the input of a multiplication gadget in $\mathcal{P}''$. This is because refresh gadgets are only used when a variable is being used in more than one place, otherwise the refresh would be superfluous. Now we are left with a set of probes on the inputs to multiplication gadgets and the inputs & outputs of arithmetic gadgets. For each probe on an input to an arithmetic gadget that reveals a share of x^1 can be moved to an input of a multiplication gadget that reveals the same share of x^1 (because from above x^1 must be inputted into a multiplication gadget). If the share doesn't reveal a share of x^1 but another input x^i it may or may not be possible to move this to reveal the same share of x^i but in fact the probe can simply be removed from $\mathcal{P}''$ without effecting the probes set dependence on x^1. The same logic applies to the probes on outputs of arithmetic gadgets. Therefore the probe set $\mathcal{P}'$ can be transformed into $\mathcal{P}''$ retaining dependence on x^1 which $\mathcal{A}_3$ can return and will be unsimulatable, meaning $\mathcal{A}_3$ wins it's game.
Now show:

$$\forall \mathcal{A}_2 \exists \mathcal{S}_2 \text{ that wins Game 2} \implies \forall \mathcal{A}_3 \exists \mathcal{S}_3 \text{ that wins Game 3}$$

For any adversary $\mathcal{A}_3$ in Game 3 define $\mathcal{A}_2$ such that it returns the same probes and inputs. Then by assumption there exists a simulator that wins Game 2 which can be directly used to simulate the probes in Game 3. □

4.5.4 Linear Algebra

Now it is shown how to reformulate Game 3 into a linear algebra problem. First a sequence of matrices is defined from the Flattened circuit in Game 3, then winning Game 3 is formulated in terms of properties of these matrices. Recall $\mathcal{G}_m$ is the set of multiplicative gadgets in the circuit $\mathcal{C}$. The probes in $\mathcal{P}''$ that $\mathcal{A}_3$ returns can be uniquely identified by 3 indices (g, i, j) where g is the index of the multiplicative gadget i is the share of the left input revealed and j is the share of the right input revealed. Let $\mathbf{a}^g = (\mathbf{a}_1^g, \ldots, \mathbf{a}_d^g)$ be the left input of the g-th multiplication gadget and $\mathbf{b}^g = (\mathbf{b}_1^g, \ldots, \mathbf{b}_d^g)$ be the right input of the g-th multiplication gadget. Each $\mathbf{a}^g$ and $\mathbf{b}^g$ is a linear function of the inputs $\mathbf{x}^1, \ldots, \mathbf{x}^N$. Again recall that $\mathbf{x}_k$ denotes the vector of the k-th share of each inputs $(\mathbf{x}_k^1, \ldots, \mathbf{x}_k^N)$. Then the we can write $\mathbf{a}^g = (\boldsymbol{\alpha}^g \cdot \mathbf{x}_1, \ldots, \boldsymbol{\alpha}^g \cdot \mathbf{x}_d)$ and $\mathbf{b}^g = (\boldsymbol{\beta}^g \cdot \mathbf{x}_1, \ldots, \boldsymbol{\beta}^g \cdot \mathbf{x}_d)$ where $\alpha^g, \beta^g \in \mathbb{F}_2^N$ are uniquely defined by the sequence of arithmetic gadgets that feed into the multiplication. All this is really saying is that the k-th share of the input to a multiplication gadget is a linear function of the k-th share of each input to the circuit, and also that it is the same linear function for share k and share l. Now the sequence of matrices $M_1, \ldots, M_d$ is defined. The matrix M_i is built from rows where each row is α^g or β^g for some g. The rows of M_i are all the α^g such that the tuple $(g, i, \_) \in \mathcal{P}'$ and all the β^g such that the tuple $(g, \_, i) \in \mathcal{P}'$. Each matrix $M_i \in \mathcal{M}_{N \times t_i}(\mathbb{F}_2)$ for some $0 \geq t_i \leq 2 * t$. t_i is determined by how many probes exist on the i-th share.

Proposition 15. *([BGR18] Lemma 2). For any inputs $(x^1, \ldots, x^N) \in \mathbb{F}_2^N$ the probed variables $(v_1, \ldots, v_q) = \mathcal{C}(\curvearrowleft^1, \ldots, \curvearrowleft^N)_{\mathcal{P}''}$ can be perfectly simulated if and only if the matrices M_i satisfy*

$$Im(M_1^T) \cap \cdots \cap Im(M_d^T) = \emptyset$$

Proof. First show that a non-null intersection implies the variables cannot be simulated. Assume that $\mathbf{w} \in \mathrm{Im}(M_1^T) \cap \cdots \cap \mathrm{Im}(M_d^T) \subset \mathbb{F}_2^N$. Then by definition

$$\mathbf{w} = M_1^T \cdot \mathbf{u}_1^T = \cdots = M_d^T \cdot \mathbf{u}_d^T$$

or equivalently

$$\mathbf{w} = \mathbf{u}_1 \cdot M_1 = \cdots = \mathbf{u}_d \cdot M_d$$

For $\mathbf{u}_i \in \mathbb{F}_2^{t_i}$ where t_i is the number of probes on the i-th share of variables as defined above.

Now without loss of generality let the output variables $\mathbf{v} = (v_1, \ldots, v_q)$ be ordered such that $\mathbf{v}$ is the concatenation of d vectors $\mathbf{v}_1 \ldots \mathbf{v}_d$ such that $\mathbf{v}_i = M_i \cdot \mathbf{x}_i$. This can be done because each v_i is equal to $\boldsymbol{\alpha}^g \cdot \mathbf{x}_i$ or $\boldsymbol{\beta}^g \cdot \mathbf{x}_i$ for some i by reordering the vector $\mathbf{v}$ terms can be grouped so that all those depending on the first share vector $\mathbf{x}_1$ are first in the vector $\mathbf{v}$ and so on. Dividing $\mathbf{v}$ by shares then yields the expression in terms of the matrices M_i.

Now combining both of the previous statements and setting $\mathbf{x} = (x^1, \ldots, x^N)$ we have :

$$\sum_{i=1}^{d} \mathbf{u}_i \cdot \mathbf{v}_i = \sum_{i=1}^{d} \mathbf{u}_i \cdot M_i \cdot \mathbf{x}_i = \sum_{i=1}^{d} \mathbf{w} \cdot \mathbf{x}_i = \mathbf{w} \cdot \sum_{i=1}^{d} \mathbf{x}_i = \mathbf{w} \cdot \mathbf{x}$$

This then implies that the distribution of the tuple $(v_1, \ldots, v_q) = (\mathbf{v}_1 \| \ldots \| \mathbf{v}_d)$ is dependent on $\mathbf{x}$ and therefore cannot be simulated without knowledge of $\mathbf{x}$.

Now show that a null intersection implies simulatability. First by definition of how the uniform encoder functions (Algorithm 4) we can write $\mathbf{x}_1 = \mathbf{x} + \mathbf{x}_2 + \cdots + \mathbf{x}_d$ and each $\mathbf{x}_2, \ldots, \mathbf{x}_d$ is distributed uniformly at random. From this it is clear that the distribution of $\mathbf{v}_2, \ldots, \mathbf{v}_d$ is also uniformly random. Now consider an element of $\mathbf{v}_1$, this element is the dot product $\mathbf{r} \cdot \mathbf{x}_1$ where $\mathbf{r}$ is a row of M_1. By assumption of the null intersection there exists a matrix M_i such that $\mathbf{r} \notin \mathrm{Im}(M_i^T)$ (and $i \neq 1$ as $\mathbf{r}$ is trivially in the image of M_1^T). Then this means that $\mathbf{r} \cdot \mathbf{x}_i$ is a uniform random variable independent of $\mathbf{v}_2, \ldots, \mathbf{v}_d$. Then $\mathbf{r} \cdot \mathbf{x}_1 = \mathbf{r} \cdot (\mathbf{x} + \mathbf{x}_2 + \cdots + \mathbf{x}_d)$ which is distributed uniformly at random (because of the term $\mathbf{r} \cdot \mathbf{x}_i$ in the sum). Since this argument applies for each element of $\mathbf{v}_1$ every element of $(v_1, \ldots, v_1)$ can be simulated without knowledge of $\mathbf{x}$, completing the proof. □

Proposition 16. *([BGR18] Equation 6). If the intersection of the set of matrices in non empty then it is the span of a single vector.*

$$Im(M_1) \cap \cdots \cap Im(M_d) = \langle \mathbf{w} \rangle$$

For $\mathbf{w} \in \mathbb{F}_2^N$ *with* $\mathbf{w} = \alpha^{\mathbf{g}}$ *or* $\mathbf{w} = \boldsymbol{\beta}^g$ *for some* g.

Proof. First it is shown that one matrix M_i must consist of only one row. This is the case because for the matrices to describe a successful probing attack there must be probes on every share index. When choosing the shares to probe with the left inputs a there are $d-1$ probes to choose so one share index will be missed, without loss of generality let this be the first share. Then when choosing shares to probe with the right inputs b the first share must be probed leaving $d-2$ remaining probes and d positions. This results in 2 share indices that are only probed by one probe and therefore a matrix M_i that only has one row.

Therefore one matrix $M_i = \mathbf{w}^t$ for some vector $\mathbf{w} \in \mathbb{F}_2^N$. Then $\mathrm{Im}(M_i^T) = \mathrm{Im}(\mathbf{w}) = \langle \mathbf{w} \rangle$. For the intersection to be non-null $\mathbf{w}$ must be in the image of every other matrix. This then implies that the span of $\mathbf{w}$ is a subset of each of the other images making the intersection equal to the span of $\mathbf{w}$. Also $\mathbf{w}$ is equal to $\boldsymbol{\alpha}^g$ or $\boldsymbol{\beta}^g$ by virtue of being a row of M_i. □

If the intersection of the images of the matrices is indeed the span of some vector $\mathbf{w}$ we say there is a probing attack on $\mathbf{w}$. Therefore an adversary can win Game 3 if an only if $\mathrm{Im}(M_1^T) \cap \ldots \mathrm{Im}(M_d^T) = \langle \mathbf{w} \rangle$, for $\mathbf{w} \in \{\boldsymbol{\alpha}^g, \boldsymbol{\beta}^g\}_g$.

4.5.5 Method

The method proposed in [BGR18] works as follows. Loop over all vectors $\mathbf{w} \in \{\boldsymbol{\alpha}^g, \boldsymbol{\beta}^g\}_g$ and check if there exists a probing attack on $\mathbf{w}$ (if $\mathcal{P}''$ can be constructed such that $\mathrm{Im}(M_1^T) \cap \cdots \cap \mathrm{Im}(M_d^T) = \langle \mathbf{w} \rangle$). For each $\mathbf{w}$ an iterative algorithm is used. Two sequences of sets are defined and iterativly built until one of two conditions are met or until the iteration has occurred m times where m is the number of multiplication gadgets in the circuit. The first sequence of sets $(\mathcal{G}_i)_i$ starts with

$$\mathcal{G}_1 = \{g : \mathbf{w} = \boldsymbol{\alpha}^g\} \cup \{g : \mathbf{w} = \boldsymbol{\beta}^g\}$$

The set $\mathcal{G}_1$ is the set of indices of multiplicative gadgets that have $\mathbf{w} \cdot \mathbf{x}$ as an input. The the next sequence is the set $\mathcal{O}_1$ defined as

$$\mathcal{O}_1 = \{\boldsymbol{\beta}^g : \mathbf{w} = \boldsymbol{\alpha}^g\} \cup \{\boldsymbol{\alpha}^g : \mathbf{w} = \boldsymbol{\beta}^g\}$$

This set is referred to as free vector operands. If an adversary spends one probe on a gadget $g \in \mathcal{G}_1$ such that $\mathbf{w} = \boldsymbol{\alpha}^g$ then $\boldsymbol{\beta}^g$ can also be added to any matrix M_j to get the value $\boldsymbol{\beta} \cdot \mathbf{x}_j$. Then if $\mathbf{w} \in \langle \mathcal{O}_1 \rangle$ the adversary can combine several vector operands to make $\mathbf{w} \in \mathrm{Im}(M_j^T)$ without directly adding $\mathbf{w}$ to M_j. The free vectors can also be combined with outer multiplications which leads to the iterative definition:

$$\mathcal{G}_{i+1} = \{g : \boldsymbol{\alpha}^g \in \mathbf{w} + \langle \mathcal{O}_i \rangle\} \cup \{g : \boldsymbol{\beta}^g \in \mathbf{w} + \langle \mathcal{O}_i \rangle\}$$

and

$$\mathcal{O}_{i+1} = \{\boldsymbol{\beta}^g : \boldsymbol{\alpha}^g \in \mathbf{w} + \langle \mathcal{O}_i \rangle\} \cup \{\boldsymbol{\beta}^g : \boldsymbol{\beta}^g \in \mathbf{w} + \langle \mathcal{O}_i \rangle\}$$

On each step two conditions are tested:

- if $\mathcal{G}_i = \mathcal{G}_{i-1}$, then there is no probing attack on $\mathbf{w}$ so iteration for $\mathbf{w}$ stops and moves to the next value of $\mathbf{w}$.
- if $\mathbf{w} \in \langle \mathcal{O} \rangle$, then there is an attack on $\mathbf{w}$

If a probing attack is found then **True** is returned with the sequence of $\mathcal{G}_i, \mathcal{O}_i$ as proof. If all vectors $\mathbf{w}$ are tested without finding an attack then **False** is returned. The point where **True** is returned indicates a fault in the multiplication gadget g that $\mathbf{w} = \boldsymbol{\alpha}^g$ or $\mathbf{w} = \boldsymbol{\beta}^g$ being tested. This fault can be fixed by inserting a refresh gadget before the multiplication. By running the algorithm and finding all multiplication gadgets that contain faults then inserting refresh gadgets before the faulty multiplications yields a circuit that is $(d-1)$-Threshold probing secure ([BGR18] Proposition 10).

4.6 Further Reading

4.6.1 Robust Probing Model

The security models previously presented aren't the only forms of security worth considering. Important artifacts of digital circuits that aren't considered by the Threshold probing model are glitches and transitions. Glitches are an unavoidable artifact of digital circuits. When considering digital logic we treat wires as carrying either 0 or 1, however in reality the current values are not discrete in $\{0, 1\}$ but take infinitely many values between. Logic gates fluctuate between values which then propagate to subsequent logic gates, potentially accumulating information about the secret values until the path reaches a register. To accommodate this an the Glitch-Robust probing model each probe reveals values back until a register is reached. Transitions are the property that when a logic gate changes values it's power usage will depend on both the new value and the old value. To account for this in the Transition-Robust probing model the circuit model is adjusted to have layers for each time step, a probe receives both the value at the current time step and the value before. Accounting for both of these phenomena in the Glitch+Transition Robust a probe is expanded to account for transitions then expanded again to account for glitches. These models are formally defined in [CS21] and [Cas] as well as definition compositional properties similar to the composition strategies in the Threshold probing model are introduced and proved secure.

4.6.2 Random Probing Model (Again)

As well as proving security in the Threshold probing model security can be directly proved in the random probing model. The techniques presented in [Cas+21] provide two methods to proving security directly in the random probing model. This is attempted through statistically esitimating the probability of success of a probing attack. Then an object called a probe distribution table (PDT) is defined. Each gadget has a probe distribution table which can be turned into security bounds in the random probing model. Additionally the PDT of the composition of two gadgets can be computed efficiently from the PDTs of the constituent gadgets and then turned into a bound on the security of the new gadget. This leads to the technique of manually computing PDTs for small gadgets and then combining these to form the PDT of large gadgets as well as a security proof. This technique is limited in the by computational cost, (up to $d = 6$ for the ISW multiplication [Cas] Pg 127). They also present an alternative model called the local random probing model (LRPM) this is designed to model the amount of information an adversary can extract by using a type of attack called a Soft analytical side channel attack (SASCA) [VGS14]. SASCAs should extract the optimal amount of information from the target so by bounding the amount of information that can be extracted by a SASCA the security of the gadget can be shown.

Chapter 5

Critical Evaluation

The work presented covers several important topics to introduce side channel attacks and security however there is much more that I would have liked to have done. Below is a list of topics and ideas that, given time I would have liked to add to this project.

- Differential Power Analysis attacks
- Soft analytical side channel attacks
- Timing attack example & mitigation
- Attacks on side channels other than power (i.e. EM radiation)
- In depth examples of each attack including source code and results analysis
- Worked examples of masking well known cryptographic algorithms
- Computing the training information and perceivable information on an implementation
- Proof of security growing exponentially with the masking order
- Full detail on the Robust probing models
- Full detail on Probe distribution tables and Local random probing model
- Discussion of shuffled algorithms
- Detail and implement algorithms that tools like maskVerif and ironMask use
- Detail techniques for increasing the SNR. e.g. noise generation circuits
- Discussion of software masking

Unfortunately due to time constraints these did not get completed. I am especially disappointed with not having implemented any attacks or any masked algorithms.

Chapter 6

Conclusion

In this project I presented an introduction to the topic of side channel attacks and security. This proofs presented are edited for the target reader who isn't an expert in the field of side channel. Additionally several new proofs are presented, specifically for Proposition 14, Theorem 1 (To the best of my knowledge hasn't been presented but is so simple it is likely it has) and Proposition 5. Proofs in section 3.3 have been expanded upon to reduce the number of sources required to see the complete argument. Additionally a unified game based formulation of security models should ease the complexity of learning them from the literature where security models are often presented under different definitions. As it stands this project should serve as a gentle introduction to the field of side channel analysis by introducing multiple aspects of the field. This project is aimed so that whatever topics in side channel analysis the reader might discover later this project can serve as a backbone to support reading research papers.

Bibliography

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology — CRYPTO '96*. Ed. by Neal Koblitz. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 104–113. ISBN: 978-3-540-68697-2.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397. ISBN: 978-3-540-48405-9. DOI: https://doi.org/10.1007/3-540-48405-1_25.

[MD99] Thomas S. Messerges and Ezzy A. Dabbish. "Investigations of Power Analysis Attacks on Smartcards". In: *USENIX Workshop on Smartcard Technology (Smartcard 99)*. Chicago, IL: USENIX Association, May 1999. URL: https://www.usenix.org/conference/usenix-workshop-smartcard-technology/investigations-power-analysis-attacks-smartcards.

[AK01] András Antos and Ioannis Kontoyiannis. "Convergence properties of functional estimates for discrete distributions". In: *Random Structures & Algorithms* 19.3-4 (2001), pp. 163–193. DOI: https://doi.org/10.1002/rsa.10019. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rsa.10019. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.10019.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski, çetin K. Koç, and Christof Paar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–28. ISBN: 978-3-540-36400-9. DOI: https://doi.org/10.1007/3-540-36400-5_3.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 463–481. ISBN: 978-3-540-45146-4.

[Pan03] Liam Paninski. "Estimation of Entropy and Mutual Information". In: *Neural Computation* 15.6 (June 2003), pp. 1191–1253. ISSN: 0899-7667. DOI: 10.1162/089976603321780272. eprint: https://direct.mit.edu/neco/article-pdf/15/6/1191/815550/089976603321780272.pdf. URL: https://doi.org/10.1162/089976603321780272.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". In: *Cryptographic Hardware and Embedded Systems - CHES 2004* (2004), pp. 16–29. DOI: http://dx.doi.org/10.1007/978-3-540-28632-5_2.

[Ors+04] S.B. Ors et al. "Power-analysis attack on an ASIC AES implementation". In: *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.* Vol. 2. 2004, 546–552 Vol.2. DOI: 10.1109/ITCC.2004.1286711.

[05] "Entropy, Relative Entropy, and Mutual Information". In: *Elements of Information Theory*. John Wiley and Sons, Ltd, 2005. Chap. 2, pp. 13–55. ISBN: 9780471748823. DOI: https://doi.org/10.1002/047174882X.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047174882X.ch2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047174882X.ch2.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511581274.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. "Soft Analytical Side-Channel Attacks". In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 282–296. ISBN: 978-3-662-45611-8.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. *Making Masking Security Proofs Concrete or How to Evaluate the Security of any Leaking Device (Extended Version)*. Cryptology ePrint Archive, Paper 2015/119. https://eprint.iacr.org/2015/119. 2015. URL: https://eprint.iacr.org/2015/119.

[Bar+16] Gilles Barthe et al. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 116–129. ISBN: 9781450341394. DOI: 10.1145/2976749.2978427. URL: https://doi.org/10.1145/2976749.2978427.

[Bat+16] Alberto Battistello et al. *Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme*. Cryptology ePrint Archive, Paper 2016/540. https://eprint.iacr.org/2016/540. 2016. URL: https://eprint.iacr.org/2016/540.

[Bel+16] Sonia Belaïd et al. *Randomness Complexity of Private Circuits for Multiplication*. Cryptology ePrint Archive, Paper 2016/211. https://eprint.iacr.org/2016/211. 2016. URL: https://eprint.iacr.org/2016/211.

[BGR18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. "Tight Private Circuits: Achieving Probing Security with the Least Refreshing". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven Galbraith. Cham: Springer International Publishing, 2018, pp. 343–372. ISBN: 978-3-030-03329-3.

[CS18] Gaëtan Cassiers and François-Xavier Standaert. *Trivially and Efficiently Composing Masked Gadgets with Probe Isolating Non-Interference*. Cryptology ePrint Archive, Paper 2018/438. https://eprint.iacr.org/2018/438. 2018. DOI: 10.1109/TIFS.2020.2971153. URL: https://eprint.iacr.org/2018/438.

[Bro+19] Olivier Bronchain et al. *Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations*. Cryptology ePrint Archive, Paper 2019/132. https://eprint.iacr.org/2019/132. 2019. URL: https://eprint.iacr.org/2019/132.

[CS19] Gaëtan Cassiers and François-Xavier Standaert. "Towards Globally Optimized Masking: From Low Randomness to Low Noise Rate or Probe Isolating Multiplications with Reduced Randomness and Security against Horizontal Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019 (2019), pp. 162–198.

[Che+19] Eloi de Cherisey et al. *Best Information is Most Successful*. Cryptology ePrint Archive, Paper 2019/491. https://eprint.iacr.org/2019/491. 2019. URL: https://eprint.iacr.org/2019/491.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. "Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.2 (Feb. 2021), pp. 136–158. DOI: 10.46586/tches.v2021.i2.136-158. URL: https://tches.iacr.org/index.php/TCHES/article/view/8790.

[Cas+21] Gaëtan Cassiers et al. "Towards Tight Random Probing Security". In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Cham: Springer International Publishing, 2021, pp. 185–214. ISBN: 978-3-030-84252-9.

[Mas+22] Loïc Masure et al. *Information Bounds and Convergence Rates for Side-Channel Security Evaluators*. Cryptology ePrint Archive, Paper 2022/490. https://eprint.iacr.org/2022/490. 2022. URL: https://eprint.iacr.org/2022/490.

[Cas] Gaëtan Cassiers. *Composable and efficient masking schemes for side-channel secure implementations*. URL: https://dial.uclouvain.be/pr/boreal/object/boreal:264203.

[Crya] CryptoExperts. *IronMask: Versatile Verification of Masking Security*. URL: https://github.com/CryptoExperts/IronMask.

[Cryb] CryptoExperts. *MaskVerif, Automatic tool for the verification of side-channel countermeasures*. URL: https://cryptoexperts.com/maskverif/.

[DDF] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *Journal of Cryptology* 32 (), pp. 151–177. DOI: https://doi.org/10.1007/s00145-018-9284-1.

[Pic] PicoTech. *Picoscope, USB oscilloscopes*. URL: https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview.

[ST] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. URL: https://csrc.nist.gov/publications/detail/fips/197/final. (accessed: 27.04.2023).