LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME
UND DATA MINING

Master Thesis
in Computer Science

Developing New Deep Learning Methods to Break Strong Physical Unclonable Functions

Sebastian Gavra

Supervisor:	Prof. Dr. Peer Kröger
Advisors:	Nadine Sarah Schüler, Prof. Dr. Dr. Ulrich Rührmair
Submission date:	30.09.2021, with minor additions and modifications in December 2022 before publication

Abstract

Current literature has researched logistic regression and multilayer perceptrons to break XOR Arbiter PUFs, but more modern deep learning methods are missing. This thesis explores new deep learning methods to break XOR Arbiter PUFs. Experiments were conducted on various long short-term memory and convolutional architectures. The results show that the most interesting ones are a 3D convolutional neural network and an $n-$dimensional CNN. Optimization techniques are discussed, mainly avoiding a limitation of the current best method, a previously proposed multilayer perceptron architecture. In the results chapter, the 3D CNN is compared to the MLP and shows the optimal data amount and training times of attacks on $(5-8)-$XORs with $64-$bit, $125-$bit, and $216-$bit challenge length. The 3D CNN was optimized to require the least amount of data and needs up to 82% less data than the best method to reach 99% validation accuracy, but mostly takes longer to finish. The relative difference in data requirement gets bigger and the relative difference in training time gets smaller when the task gets harder. Three methods are evaluated on three different aspects. The future work chapter suggests how the MLP and 3D CNN can be improved and which designs and architectures are still left to attack.

Contents

Chapter 1

Introduction & Motivation

Today, the most commonly used practice to identify and authenticate devices is to store cryptographic keys on devices. These keys are stored in non-volatile memory and there are known attacks like the invasive attack or the side channel attack that threaten security. Cryptography requires many security mechanisms, which some devices can not provide. Modern devices, which are designed as small as possible, could benefit from a more lightweight identification and authentication process.

PUF is the abbreviation for physical unclonable function, first introduced by Pappu et al. [1] in 2002. PUFs can identify and authenticate devices. Weak PUFs use a cryptographic secret key, whereas Strong PUFs do not require any cryptography. Although they are physically unclonable because of manufacturing variations, they are not mathematically unclonable. Machine learning methods can model the PUF, learn the PUF's behaviour, and predict responses with high accuracy. [2] [3]

Logistic regression and multilayer perceptrons successfully broke XOR Arbiter PUF designs, but for many well-researched machine learning fields, such as computer vision or natural language processing, these methods are not popular choices. There are no mentions of popular deep learning networks such as long short-term memory or convolutional neural networks to be found in current literature on attacks on Strong PUFs, which is surprising, because these networks have achieved state-of-the-art results in many machine learning fields and this thesis will explore if these networks can learn XOR Arbiter PUFs better than existing methods.

Chapter 2

Strong PUFs

Strong physical unclonable functions (Strong PUFs) can be used for identification and authentication of devices. For each input challenge the PUF receives, it generates an output response. Since the challenge-response interface is unprotected, it is necessary to require a large challenge space, 2^{64} being most commonly used in current research, but 2^{128} or 2^{256} can have security advantages. The challenge space has to be large to prevent predictions of CRPs, even if an attacker has collected a large subset of CRPs. Unlike Weak PUFs, which require a cryptographic key, there are no cryptographic keys required when using a Strong PUF.

A Strong PUF is a small physical system and uses physical disorder to its advantage. Because of manufacturing variations, the PUFs behaviour is only known after challenges are applied to the PUF. A challenge and its corresponding response are called a challenge-response pair (CRP). [3]

2.1 Arbiter PUF

Gassend et al. [4] introduced the Arbiter PUF (APUF) in 2002. It comprises an input edge with two multiplexers and n stages. It receives an $n-$bit input challenge and generates a single-bit output response. When a challenge is applied, a race condition is established. When the top edge arrives at the latch first, the response bit is one, and when it arrives at the bottom latch, the response bit is zero.

Machine learning methods can learn APUFs easily because it is a linearly separable task. For 99.9% accuracy, only 18050 challenges are required, with a training time of only 0.6 seconds. [2]

2.2 XOR Arbiter PUF

An XOR Arbiter PUF (XPUF) comprises one or multiple APUFs which forward their response into an XOR gate, which then generates the final response. Because of the XOR gate, the response is either 1 or −1. In Figure 2.1, we can see a 64−bit 3−XPUF example, which comprises three APUFs.

XPUFs are harder to learn by machine learning methods than APUFs. The XOR gate transforms the linearly separable task into a linearly inseparable task. The higher the number of arbiters in the XPUF design and the higher the number of challenge bits, the harder the machine learning task. Overall, a high number of arbiters makes the attack harder than a high number of challenge bits. [5] Many other PUF architectures, including optical structures, can be found in the literature, to which we refer interested readers [6–79]. Easily accessible overviews can be found in some of the existing PUF surveys and tutorials [3, 5, 80–85].

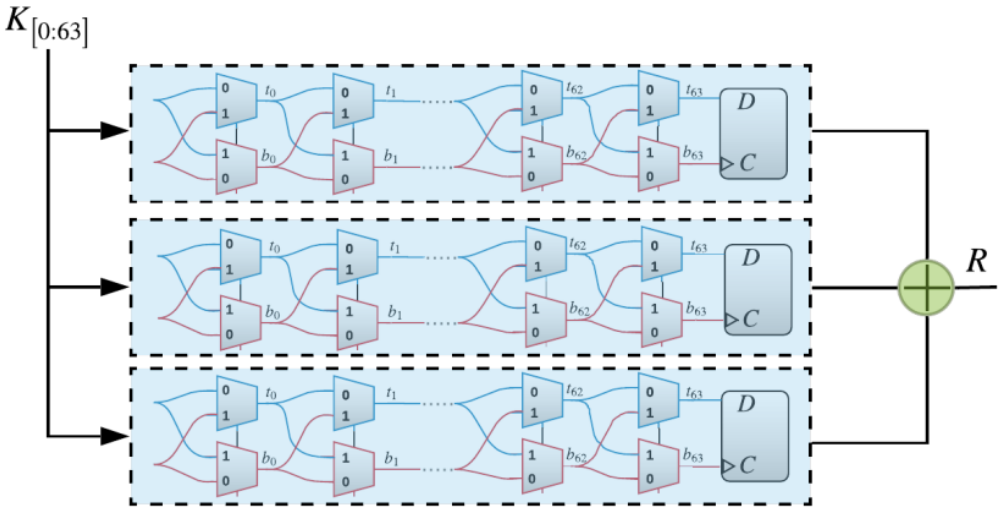

Figure 2.1: An XPUF with three arbiters and 64 stages. It is called a 64−bit 3−XOR. [86]

Chapter 3

Neural Network Concepts

This chapter focuses on the two neural network concepts which have achieved the best results in this thesis.

3.1 Multilayer perceptron

Frank Rosenblatt [87] introduced the single-layer perceptron in 1958 and it was based on the MP model, which was the first mathematical model of neurons, proposed by McCulloch and Pitts [88] in 1943. The single-layer Perceptron has a limitation - it cannot learn linearly inseparable problems.
The multilayer perceptron (MLP) however, proposed by Rumelhart et al. [89] in 1986, was the first neural network to learn a linearly inseparable problem.

The MLP is a feed-forward artificial neural network and comprises an input layer, at least one hidden layer, and an output layer. It uses a supervised learning technique called backpropagation for learning. Instead of updating each weight individually, backpropagation computes the gradient of the loss function regarding a single input-output example. [89]

3.2 Convolutional neural networks

Convolutional neural networks (CNNs) are some of the most represented networks in the deep learning field today. The term 'convolution' regarding neural networks was first used by LeCun et al. [90] in 1989 when they constructed a CNN for hand-written zip code recognition. Since then, CNNs have achieved the following state-of-the-art results: 1D CNNs are used for time series forecasting and signal identification, 2D CNNs are used for image classification, image object detection, image segmentation and face recognition and 3+D CNNs are used for human action recognition and video object de-

tection/recognition. [91]

Unlike traditional feature extraction methods like scale-invariant feature transform, histogram of oriented gradients, and local binary patterns, a CNN is a feed-forward neural network that can extract features from data automatically. A CNN kernel responds to various features in the data and outputs filters that are only connected to a few neurons. This reduces network parameters and therefore speeds up convergence. [91]

There are exclusive CNN operations that are not being used by other neural networks, although none of these operations are used in the neural network presented in this thesis (see section 7.2). To prevent information loss in the border when a small kernel size is set, the padding operation enlarges the data. To decrease data density, the stride operation can be employed, which skips adjacent features. To obviate redundancy in the data, the pooling (a.k.a. down-sampling) operation can remove trivial features and therefore increase learning efficiency. [91]

Chapter 4

Related Work

Rührmair et al. [2] conducted the first machine learning attacks on XPUFs in 2010. Their experiments included logistic regression, evolution strategies, and support vector machines. Logistic regression significantly outperformed the other two methods. They broke the 6−XOR 64−bit and 5−XOR 128−bit with a 99% prediction rate.

Tobisch and Becker [92] parallelized the logistic regression method in 2015. They showed it is possible to break 8−XOR & 9−XOR 64-bit, but not consistently, requiring much more data and training time than the deep learning methods or the optimized logistic regression [93] in the future.

Aseeri et al. [94] conducted the first deep learning attacks on XPUFs in 2018. They used an MLP architecture that comprises three hidden layers, each containing 2^k neurons for every k-XPUF. They successfully broke the 8−XOR 64−bit and 7−XOR 128−bit with 99% validation accuracy and used fewer CRPs than the logistic regression method.

In 2019, Santikellur et al. [95] proposed a Tensor Regression Network. It is a parallel structure containing separate models, similar to the logistic regression. They could not break the 7−XOR 64−bit and 6−XOR 128−bit.

Mursi et al. [86] released a second MLP approach in 2020. The network architecture comprises three hidden layers, the first and third contain 2^{k-1} and the middle layer contains 2^k neurons for k-XPUFs. This was the first method to break the 9−XOR 64−bit consistently and used less CRPs than all previous methods, although later Wisiol et al. [93] found a bug in their CRP generation process.

Wisiol et al. [93] compared the previous four methods on multiple threads in 2021. Since they generated their data using the pypuf library [96], they corrected the previous CRP amounts by Mursi et al. [86] and this method is considered to be the best to date. In addition, they broke 10−XOR & 11−XOR 64−bit consistently. They also optimized the logistic regression and

could break the 10−XOR 64-bit and although it required more CRPs, it took less training time than the best method.

Other attacks on PUFs, including side channels and protocol level attacks, can be found in [97–103] and elsewhere. Also works on PUF formalization, PUF-uses in advanced protocols, and PUF security proofs, which are all antagonists to these attacks, deserve mentioning in this context [104–110]. Very recently, Strong PUF security metrics have been developed that try to "anticipate" the security against ML-methods by various means [111].

Chapter 5

Data & Attack

This chapter contains technical information about the data and the attack.

5.1 Hardware

All experiments mentioned in this thesis have been executed on an Intel Core i7-8700 CPU at 3.2 GHz and 64 GB of RAM on up to 12 cores. All training times and processing times mentioned in this thesis refer to this CPU.

With this RAM capacity, there is a limit to how many CRPs can be loaded into memory in one chunk. The data limits for this thesis are 75 million 64−bit CRPs, 35 million 125−bit CRPs, and 20 million 216−bit CRPs.

Not having much RAM is not necessarily an issue, as Aseeri et al. [94] have shown in their paper, it is possible to load and pre-process multiple chunks of CRPs sequentially and train on the entire dataset in one run.

5.2 Data generation

Every dataset has been generated with the pypuf library [96]. The challenge and response data have been saved in two separate text files. One million 64−bit CRPs require 62 megabytes of disk space. The data is loaded into a memory map [112]. Mursi et al. [86] have used this concept before.

For this thesis, all CRPs have been loaded from 1 to n, where 1 is the first CRP in the text files and n is the size of the subset. All CRPs are loaded sequentially and shuffled during training before every epoch.

5.3 Pre-processing

Before the model can be trained, there are four pre-processing steps to perform:

Step 1: convert challenge bits from 1 and 0 to 1 and -1.
Step 2: reverse the order of challenge bits along axis 1.
Step 3: calculate the cumulative product of challenge bits along axis 1.
Step 4: convert response bits from 1 and -1 to 1 and 0.

These pre-processing steps are required because, in an APUF, the delays of the signals are additive and the transformed challenge space can be separated by a hyperplane. Since an APUF with an $n-$bit challenge represents an $n-$dimensional challenge space, a reverse accumulative product transform has to be applied to the challenge data. [2]

Pre-processing can take a considerably long time, 1 million 64-bit CRPs take almost 10 seconds. When a lot of data is being pre-processed for training, pre-processing can take longer than training. Saving pre-processed CRPs in a text file can therefore save time if multiple runs are planned to be executed.

5.4 Task

The task of the neural network is to create a model which predicts the response for any given challenge. There are only two possible predictions, either one or zero. This is known as a binary classification task.

There are four ways to evaluate a prediction: correctly predicted as one (true positive (TP)), incorrectly predicted as one (false positive (FP)), incorrectly predicted as zero (true negative (TN)), and correctly predicted as zero (false negative(FN)). The accuracy can be calculated as:

$$\frac{TP + FN}{TP + TN + FP + FN} \tag{5.1}$$

The difference between the training accuracy and the validation accuracy is that the training accuracy is evaluated on the data the model was trained on after every step, and the validation accuracy is evaluated on the data which was excluded from the training data at the end of every epoch.

In other binary classification tasks, there are other metrics, e.g. precision and recall, but these metrics are not relevant for this task because classifying as one or zero is equally as important.

Chapter 6

Experiments

This chapter contains details about experiments of all implemented methods in chronological order. All methods have been implemented using the Keras framework, unless otherwise specified.

All methods require an additional reshape step compared to the MLP [86]: the LSTM and 1D CNN require a 1D tensor, nD CNNs require an nD tensor. The reshape step takes a fraction of a second to complete for many millions of CRPs and can therefore be ignored when measuring runtime.

6.1 Long short-term memory

First, a long short-term memory (LSTM) network was developed. There was one approach with a single LSTM cell, one with two LSTM cells, and one with three LSTM cells.

Unfortunately, the LSTM needed over 30 minutes to reach 97% validation accuracy on the 1−XOR, whereas state-of-the-art methods can reach 99% validation accuracy in less than five seconds. There were sudden validation accuracy drops (see Figure 6.1) of up to 44% in only one epoch, this could have happened because of the LSTM's forget gate. Validation accuracy drops when training the MLP [86] were always less than 1% per epoch. After multiple failed runs on the 2−XOR, it was not developed any further.

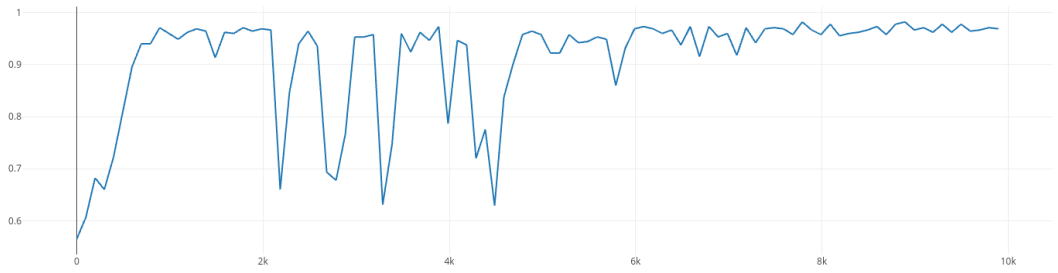

Figure 6.1: Multiple validation accuracy drops during the LSTM training, the x-axis shows the number of steps and the y-axis shows the validation accuracy.

6.2 CNN

6.2.1 1D

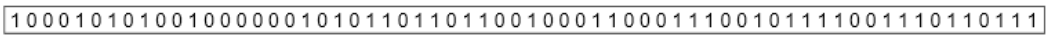

Figure 6.2: A 64−bit challenge represented as a 1D vector.

The 1D CNN achieved good results on the $(1-3)$-XOR but struggled to reach 95% validation accuracy on the 4−XOR. After adding up to four convolutional layers and not resolving the issue, the 2D CNN was implemented next.

Figure 6.2. visualizes how a 64−bit challenge can be represented as a one-dimensional vector.

6.2.2 2D

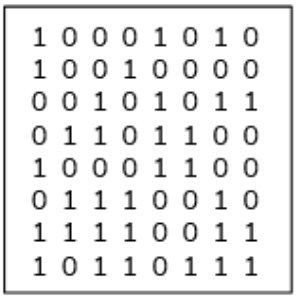

Figure 6.3: A 64−bit challenge represented as a 2D 8x8 square.

The 2D CNN also had problems with the 4−XOR but learned faster than the 1D. There were only a few runs executed with the 2D CNN because implementing the 3D CNN could have more interesting results.

Figure 6.3. visualizes how a 64−bit challenge can be represented as an 8x8 square.

6.2.3 3D

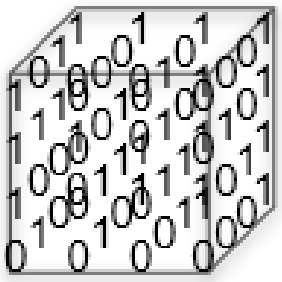

Figure 6.4: A 64−bit challenge represented as a 3D 4x4x4 cube.

The 3D CNN broke the 4−XOR on the first try and could even break the 6−XOR and 7−XOR sometimes before any optimization. This is the method that was developed and optimized the most (see Chapter 7).

Figure 6.4. visualizes how a 64−bit challenge can be represented as a 4x4x4 cube.

6.2.4 nD

After increasing the dimensionality of the CNN and seeing improvements every time, it was interesting to see if higher dimensionality could improve a CNN even more. Unfortunately, there was no machine learning framework found which supports 4+D convolutional layers.

An nD CNN was implemented by using the PyTorch framework based on an $n-$dimensional convolutional layer [113]. This layer is not part of the PyTorch framework and recursively reduces the dimensions until it reaches 3D, from that point on the PyTorch implementation is used.

The problem was that the 3D CNN implemented in PyTorch was not nearly as good as the Keras version because it struggled with the 2−XOR. Testing the nD on the 2−XOR did also not show any promising results. Because the difference between a method that struggled with the 2−XOR and a method that could break the 7−XOR seemed quite large, the nD CNN was not developed any further.

Chapter 7

Optimization & 3D CNN Development

This chapter discusses optimization options for two methods and the 3D CNN development process.

7.1 MLP limitation

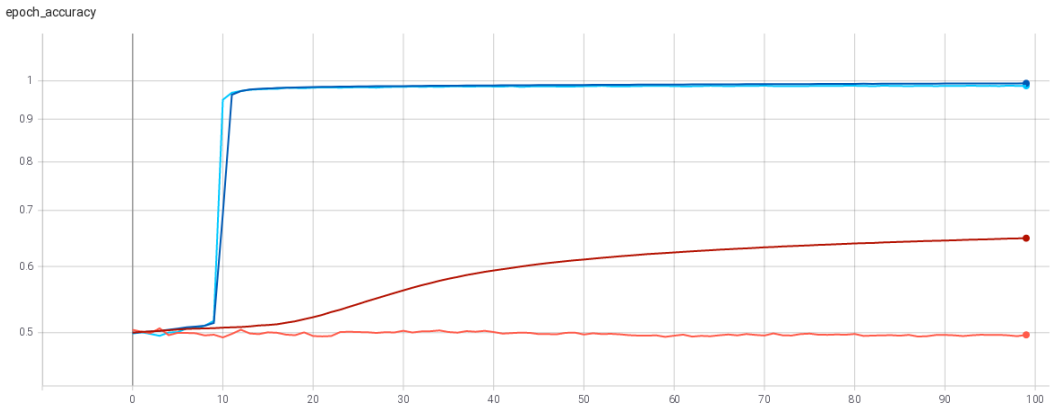

Figure 7.1: The training of a typical successful run (training accuracy: dark blue, validation accuracy: light blue) and a typical failed run (training accuracy: dark red, validation accuracy: light red) during one hundred epochs.

From now on, 'MLP' refers to the multilayer perceptron by Mursi et al. [86]. A drawback of the MLP is that it overfits early when training on less than the recommended CRP amount. Most commonly, the difference between a

successful ($> 99\%$ training accuracy & $> 99\%$ validation accuracy) and a failed run ($> 99\%$ training accuracy & $< 55\%$ validation accuracy) is that the overfitting occurs early on (see Figure 7.1). When the training accuracy is 5% higher than the validation accuracy at any time during the run, the model can never recover. Overfitting can also occur when the validation accuracy is between 95% and 99% although it is much less common and likely a sign of suboptimal network configuration.

When developing a method to successfully train on relatively low amounts of CRPs, it is crucial to avoid early overfitting, that is why the 3D CNN uses ridge activity regularization which has had the biggest impact to prevent this from occurring.

It is worth noting that ridge activity regularization was not tested on the MLP and could therefore yield even better results.

7.2 Network architecture

The 3D CNN comprises two hidden layers and one output layer.
All of the terminology used in this section in quotes can be found in the Keras documentation [114].

The method makes use of two regularizers, 'l1' (a.k.a. least absolute shrinkage and selection operator, 'lasso') and 'l2' (a.k.a ridge regression). The sum of square error estimate for least squares is

$$(Y - X\beta)^T(Y - X\beta) \tag{7.1}$$

. The regularizers add the 'l1' penalty

$$(Y - X\beta)^T(Y - X\beta) + \lambda|\beta|_1 \tag{7.2}$$

or 'l2' penalty

$$(Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta \tag{7.3}$$

to the equation [115].

Table 7.1 shows the architectural differences between both methods.

	Loss Function	**Optimizer**	**Hidden layers**	**Kernel initializers**	**Activation**
MLP	Probabilistic binary cross-entropy without reduction	Adam	3	'random_normal' & 3x 'glorot_uniform'	3x 'tanh' & 'sigmoid'
3D CNN	Logistic binary cross-entropy with sum reduction	Adam	2	'zeros' & 'he_uniform' & 'glorot_uniform'	2x 'tanh' & 'sigmoid'

Table 7.1: Architectural differences between the MLP [86] and 3D CNN.

7.2.1 3D convolutional layer

The 3D convolutional layer takes a 3D tensor as an input. The optimal amount of filters varies by the amount of data being used. As shown in Table 8.1., the amount of filters ranges from seven to twelve.

Unlike in other CNNs, where a kernel goes through the data bit by bit, this CNNs kernel size is equal to the cube size, so every challenge is learned as a whole in one step. The kernel is initialized as 'zeros' and is 'l1' regularized. The layer is activated by the 'tanh' function, which is 'l2' regularized.

As already mentioned in section 3.2, there are no exclusive CNN operations being used in the convolutional layer. Padding is not being used, because the kernel size equals the cube size and therefore, every bit in the challenge is learned exactly once, unlike with small kernels, where border features are learned more infrequently than centered features. Strides are not being used, because skipping bits in the challenge is detrimental to the learning process because every bit in the challenge could theoretically flip the response. And last, pooling is not being used, because it compresses multiple challenge bits to one feature, but every challenge bit should be learned individually.

7.2.2 Dense layer

While testing, the number of units in the dense layer did not affect the performance noticeably and the number of units should range from five to fifteen per filter. The formula for the number of units in the dense layer is the square of the number of filters in the 3D convolutional layer.

The kernel is initialized as 'he_uniform'. There is a 'zeros' bias, which is

'l1' regularized. The layer is activated by the 'tanh' function, which is 'l2' regularized.

7.2.3 Output layer

The output layer only has one unit, which is the predicted response bit. The kernel is initialized as 'glorot_uniform' and there is a 'ones' bias. The layer is activated by the 'sigmoid' function, which is 'l2' regularized.

7.2.4 Optimizer

As the MLP, the 3D CNN uses the 'Adam' optimizer, which is the RMSProp algorithm with momentum. All optimizers of the Keras framework have been tested, such as 'RMSProp', 'NAdam', 'Adamax' and 'Adam amsgrad'.

7.2.5 Loss function

Unlike the MLP, which uses a probabilistic loss function (standard 'BinaryCrossentropy'), the 3D CNN uses a logistic loss function (using 'from_logits') which also enables a reduction function. The 'sum' reduction works best on small datasets and the 'none' reduction learns faster on big datasets but doesn't succeed as often on small datasets.

7.2.6 Optimization

The method was optimized to require as few CRPs as possible. As already mentioned in section 7.1, a method that does not overfit early on could theoretically be superior to the MLP. The biggest contributing addition to avoiding overfitting was the 'l2' activity regularization in every layer.

7.2.7 Cubic & non-cubic challenge length

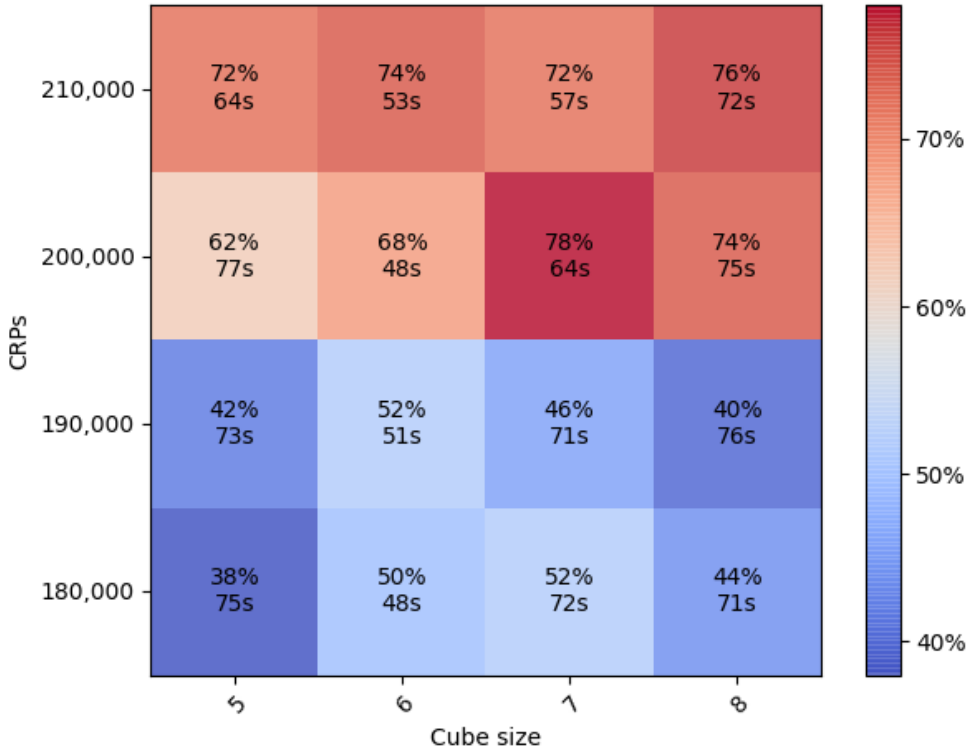

Figure 7.2: The success rate of attacks on the 96−bit 5−XOR. Success: validation accuracy $> 99\%$, 10 filters & 1000 batch size, 50 runs each. The color only pertains to the success rate. The average training time is displayed under the success rate.

Table 8.2 only contains results on XPUFs with challenge lengths 64, 125, and 216. These numbers have a whole number as their cube root. In case of a non-cubic challenge length, the cube root is rounded up, so the cube size is increased by one, e.g. 64−bit is learned on cube size 4, 65−bit is learned on cube size 5, and in this case, there would be $125 - 65$, equals 60 empty bits in the cube, which are filled with zeros.

This was tested on a 96−bit 5−XOR instance, which was trained on cube sizes from five to eight (see Figure 7.2). Surprisingly, cube size 5 performed the worst, so filling up the cube to the minimum required size is not ideal. Filling up the cube by one or two extra sizes can improve performance, but filling up by three sizes worsens performance.

Cube size 6 always has a higher success rate than cube size 5 and is always faster, cube size 7 mostly has a higher success rate than cube size 8 and is mostly faster. Cube size 7 has the highest overall success rate (62%), followed by cube size 6 (61%) but cube size 6 is faster overall, 50 seconds compared to 66 seconds on average. The best choice for the 96−bit 5−XOR is cube size 6.

In comparison, the MLP needed 250k and 260k CRPs for 58% success rate and 270k for 62% success rate with an average training time of 31 seconds. So even though the challenge has to be filled with at least 29 zeros for the 3D CNN only, the 3D CNN still needs 20% less data than the MLP which is higher than the 19% CRP difference for the 64−bit 5−XOR which does not require any filling.

This was discovered after the result chapter 8 and therefore, there was no filling to increase the cube size. Also, the method has only shaped the data into a cube and no other geometrical shapes were tested.

Chapter 8

Results

Challenge length	Design	Method	Streams or filters	Batch size
64−bit	5−XOR	MLP	5	1000
		3D CNN	7	100
	6−XOR	MLP	5	1000
		3D CNN	10	1000
	7−XOR	MLP	6	10000
		3D CNN	12	5000
	8−XOR	MLP	6	10000
		3D CNN	12	5000
125−bit	5−XOR	MLP	5	1000
		3D CNN	10	1000
	6−XOR	MLP	7	10000
		3D CNN	12	5000
216−bit	5−XOR	MLP	6	10000
		3D CNN	12	5000

Table 8.1: Network configurations for every attack. The MLP [86] uses a stream parameter, the 3D CNN uses a filter parameter.

8.1 Network configurations

As Wisiol et al. [93] have shown, it is important to configure the MLP correctly because it makes a successful attack much more likely. The batch sizes for 64−bit can be found in the original paper [86], for longer challenges, there is no information in the literature. Therefore, batch sizes for those attacks were chosen after testing. Regarding the stream parameter, which determines the length of the hidden layers in the MLP, there is no information in the literature, although, in the project's GitHub repository, there is a README file that states: "we suggest to choose stream = 5 if you are attacking the 2-XPUF, 3-XPUF, and 4-XPUF. If you are attacking the 9-XPUF, the stream should equal eight since (2^9) neurons will lead to overfitting." [116]

The configurations for all attacks of both methods can be found in Table 8.1.

8.2 MLP on pypuf data

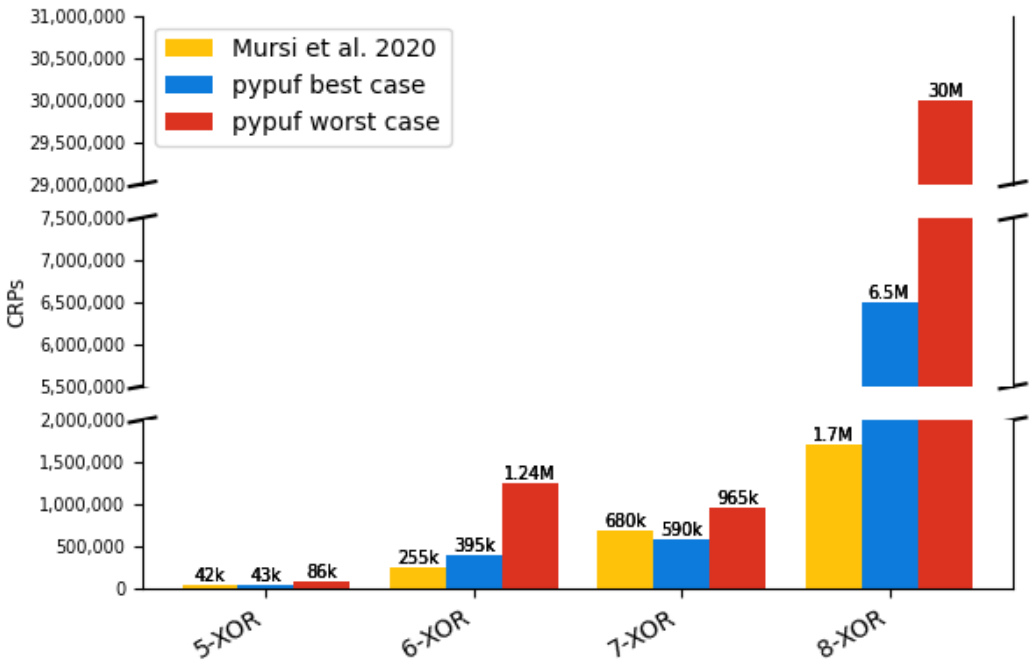

Figure 8.1: A comparison between the optimal CRP amounts published by Mursi et al. [86] on their data and pypuf data. The best case refers to the one instance out of five which requires the least amount of CRPs and the worst case refers to the one instance out of five which requires the most amount of CRPs.

As stated by Wisiol et al. [93], there was a bug in Mursi et al.'s [86] CRP generation process: about 20% of the randomly drawn delays were set to zero. As in Wisiol et al.'s work [93], the datasets of this thesis have also been generated by the pypuf library [96], but their results do not include a comparison where the goal was to use as few CRPs as possible.

Figure 8.1 displays the reference CRPs [86] and the best and worst case on pypuf data. It is also worth noting that out of the five generated 8−XOR datasets, two had to be skipped, because the MLP wasn't able to break those instances with the 75M CRP limit and two additional datasets were generated afterward.

Comparing Mursi et al.'s [86] CRP amounts to the pypuf averages (from Table 8.2), the MLP required 26% more CRPs for the 5−XOR, 173% more

CRPs for the 6−XOR, 20% more CRPs for the 7−XOR, and 1176% more CRPs for the 8−XOR.

8.3 Attacks on (5-8)-XPUFs

Challenge length	Design	Method	CRPs in 1000		Training time in s	
			min - max	avg	min - max	avg
64−bit	5−XOR	MLP	43 − 86	53	10 − 121	42
		3D CNN	26 − 60	43	66 − 324	137
	6−XOR	MLP	395 − 1240	696	26 − 164	78
		3D CNN	245 − 790	413	88 − 246	141
	7−XOR	MLP	590 − 965	813	24 − 36	29
		3D CNN	430 − 830	648	90 − 130	107
	8−XOR	MLP	6500−30000	21700	179 − 420	329
		3D CNN	2800 − 5000	3960	591 − 971	711
125−bit	5−XOR	MLP	220 − 860	443	56 − 158	103
		3D CNN	155 − 650	312	42 − 128	82
	6−XOR	MLP	3400−10000	6190	67 − 93	82
		3D CNN	840 − 2150	1474	117 − 281	167
216−bit	5−XOR	MLP	2000 − 2650	2390	32 − 47	38
		3D CNN	1280 − 1800	1480	88 − 111	104

Table 8.2: A comparison between the MLP by Mursi et al. [86] and the 3D CNN method. The table shows the results of five randomly generated instances per design. CRP amounts stand for the minimum amount needed to learn a particular instance with 99% validation accuracy in at least 5/6 successful runs. Training times are averages of all runs per instance until 99% validation accuracy was reached. The average training time of a design is calculated as the average of all average instance times.

8.3.1 Setup & methodology

For every design, five instances have been generated using the pypuf library. [96] Seven designs have been chosen, Table 8.2 therefore contains the results of 35 instances.

In related work, most result tables contain the actual accuracy values, except for Wisiol et al. [93], they stopped training at 95% validation accuracy. Excluding accuracy columns from a table can improve readability, especially when the values hold little significance (e.g. comparing results by fractional percentage).

For this thesis, the training has been stopped at 99% validation accuracy, so, given enough time, either the model has successfully reached the threshold after the training time displayed in the table, or the model has overfitted at some point during training. Although the threshold at 99% is set high, it has been observed during many runs that if the validation accuracy reaches 55% during a run, 99% can be achieved almost always if the network is configured optimally.

8.3.1.1 CRP amounts

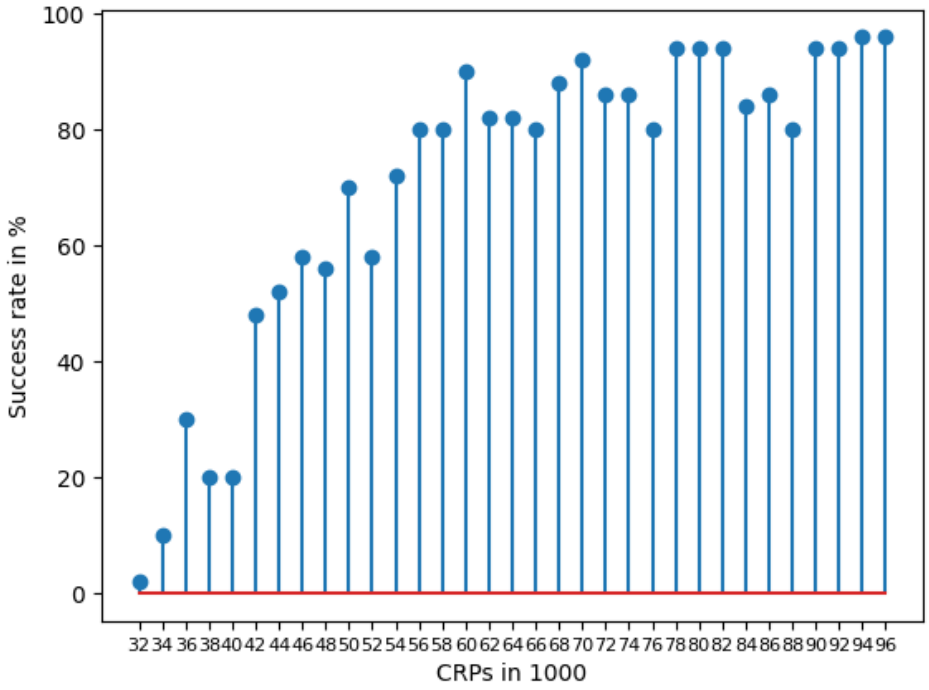

Figure 8.2: Fifty runs with every CRP amount on a 64−bit 5−XOR instance. A successful run reached 99% validation accuracy, the success rate is calculated by dividing the number of successful runs by the total number of runs.

In Figure 8.2, we can see that, for this instance, the first successful run was achieved with 32000 CRPs, so training on fewer CRPs is unlikely to be successful. When doubling the CRPs, it works 80% of the time and when tripling the CRPs, it works 96% of the time. The CRP values in the results are usually taken from the second quarter, so in this example, the CRP value would be between 48000 and 64000.

The idea behind an 'optimal' CRP amount is that only a subset of the entire dataset is required to achieve good results. The fewer CRPs are trained on, the less computing power will be required. Removing e.g. 20% of CRPs from the dataset would decrease the success rate significantly, but adding the equivalent CRP amount to the dataset would increase the success rate slightly at best.

Finding the optimal CRP amounts for a particular instance can take many

runs, which is why the following process was used:
- start one run each on three CPUs with a low CRP amount
    - if all three runs were successful, start another three runs
        - if at least 5/6 runs were successful, then the CRP amount is considered to be optimal
    - if 2/3, 1/3 or 0/3 runs were successful, increase CRPs by $\sim 2\%$, $\sim 5\%$ or $\sim 10\%$ and start over

8.3.1.2 Time

All papers mentioned in the related work chapter only measured training time. When attacking an instance, the dataset has to be pre-processed (see Section 5.3.). When comparing two methods on the same dataset and one method requires millions more CRPs than the other but trains faster, it may actually be slower in total runtime. For small datasets ($<$ 1M CRPs), pre-processing can be disregarded, but for big datasets, pre-processing time can exceed training time. Table 8.2 contains only training time results. For a closer look regarding runtime on big datasets, see Figures 8.2 and 8.3.

In Wisiol et al.'s work [93], when attacking the $64-$bit $10-$XOR, the logistic regression was trained on one billion CRPs and needed 41 minutes of training time and the MLP was trained on 119 million CRPs and needed 291 minutes of training time. It is possible to compare the runtime by adding the pre-processing time to the training time. Because the LR required 881 million CRPs more, at a duration of ten seconds per million CRPs, the pre-processing time for the LR attack would take around 146 minutes longer than the pre-processing for the MLP attack. When comparing runtime, the LR attack is still faster, but not by as much as it seems when comparing training time.

8.3.2 64-bit

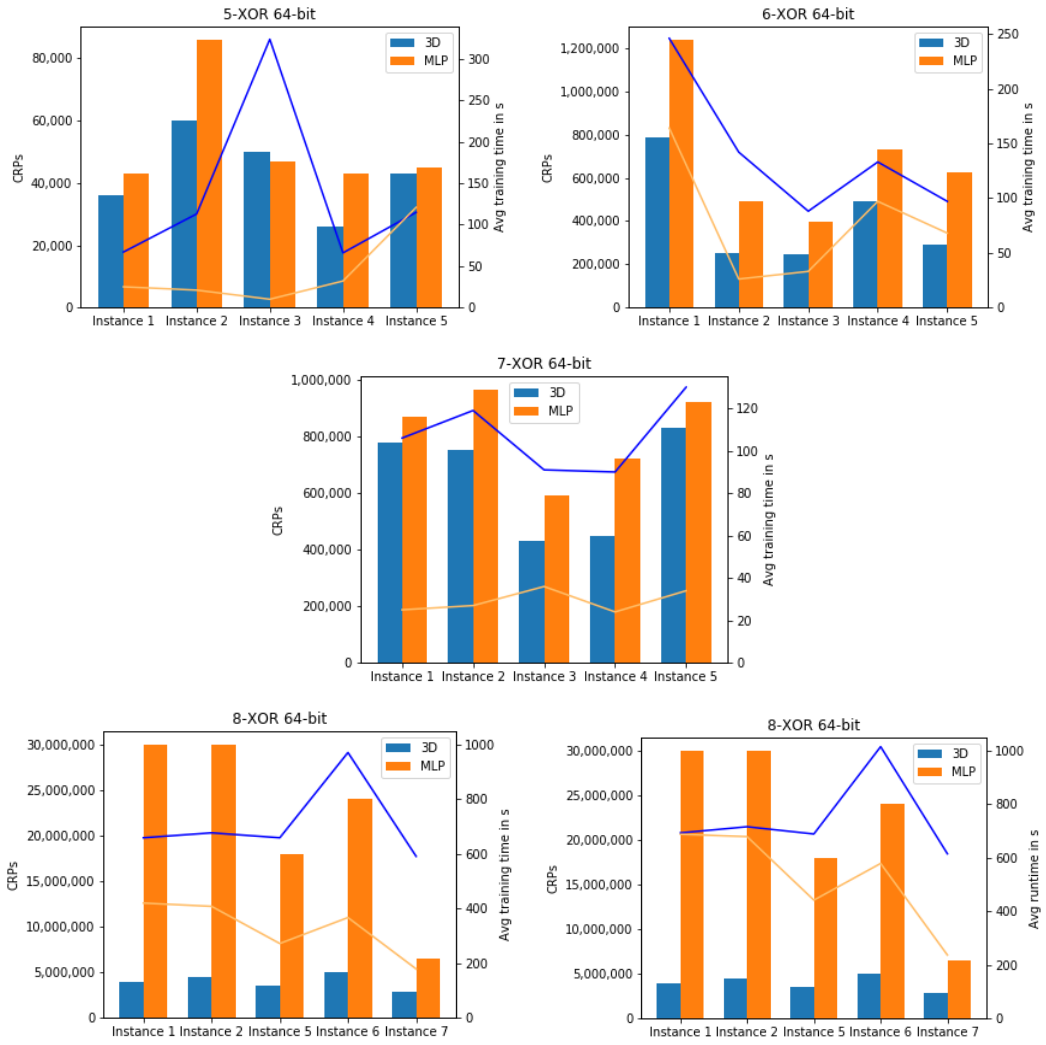

Figure 8.3: Results of every 64−bit instance. The bars and the left y-axis stand for the optimal CRP amount, the plot and the right y-axis stand for time.

The 3D CNN required less data for 19/20 instances and more data for one 5−XOR instance (see Figure 8.3). It required 19% fewer CRPs for the 5−XOR, 41% for the 6−XOR, 20% for the 7−XOR, and 82% for the 8−XOR on average.

The MLP learned 19/20 instances faster and one 5−XOR instance slower. On average, the MLP took 69% less training time for the 5−XOR, 55% less

for the 6-XOR, 73% less for the 7−XOR and 54% less for the 8−XOR.

As the related work has shown before, the number of arbiters in the design has a large impact on the amount of CRPs required and the training time. The differences in training time are interesting, surprisingly, the 7−XOR was learned the fastest on average by both methods. Also interesting is that the MLP required more data for one 6−XOR instance than all 7−XOR instances, but this is not the case for the 3D CNN.

8.3.3 125-bit & 216-bit

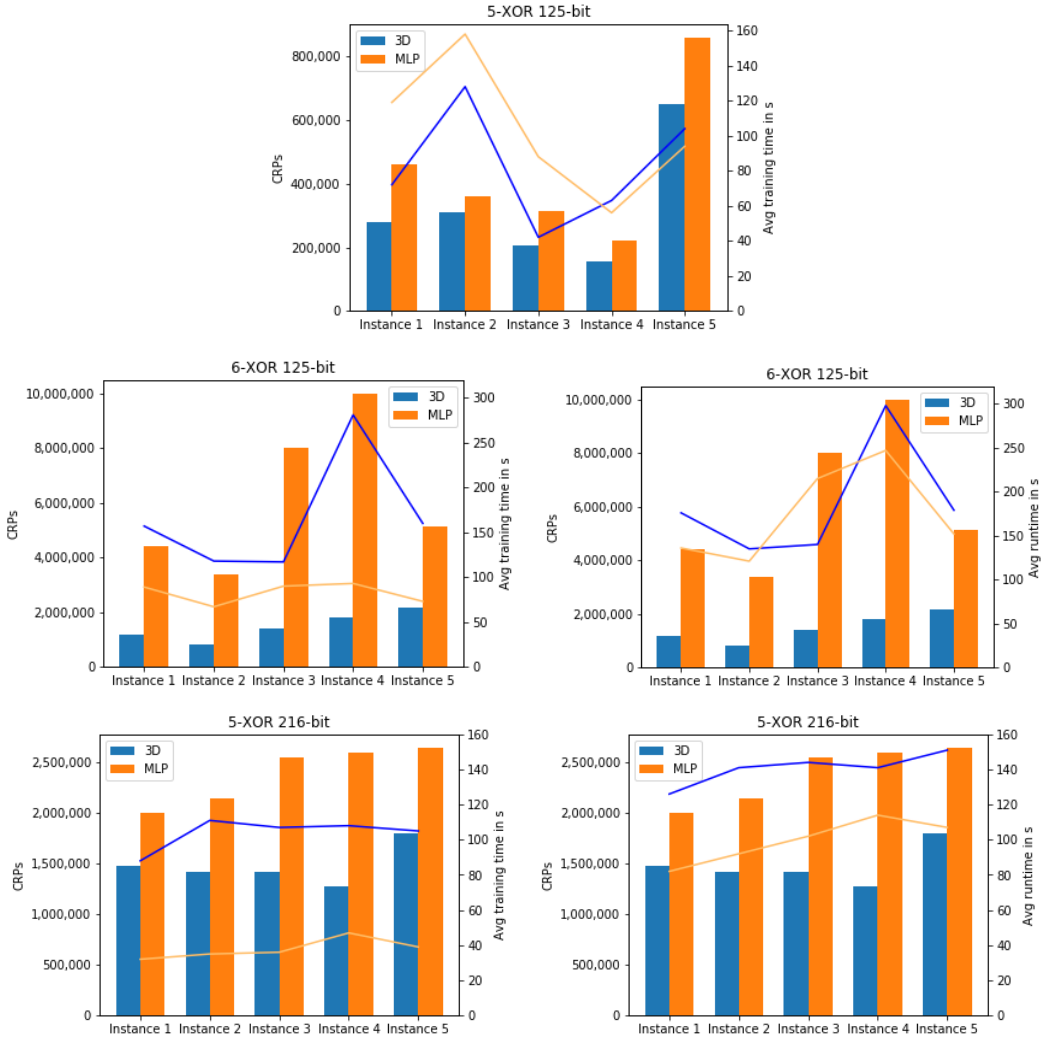

Figure 8.4: Results of every 125−bit & 216−bit instance. The bars and the left y-axis stand for the optimal CRP amount, the plot and the right y-axis stand for time.

The 3D CNN required less data for all 15 instances (see Figure 8.4). It required 30% fewer CRPs for the 5−XOR 125−bit, 76% for the 6−XOR 125−bit, and 32% for the 5−XOR 216−bit on average.

The MLP learned 12/15 instances faster and 3/15 instances slower and learned the 5−XOR 125−bit slower on average by 20%. On average, it learned the 6−XOR 125−bit faster by 51% and the 5−XOR 216−bit faster by 73%.

Interestingly, the 6−XOR 125−bit was much harder to learn than the 5−XOR 216−bit with the MLP, but it seems like these two designs were similarly as hard for the 3D CNN.

8.4 Increasing arbiters & stages

When trying to prevent machine learning attacks on XPUFs, there are two options: the first is increasing the number of arbiters that forward their response into the XOR-gate and the second is increasing the number of stages where the signal races through.

As we can see in the results, increasing the number of arbiters to eight had the biggest impact on making the attack harder. The 64−bit 8−XOR was harder to learn than the 216−bit 5−XOR. Increasing the number of stages did also have an impact. The 5−XOR 216−bit on average required at least four times as many CRPs as the 5−XOR 125−bit which required at least seven times as many CRPs as the 5−XOR 64−bit.

Regarding CRP requirement, the 3D CNN is performing better when the task is harder because the smallest relative difference is 19% for the easiest task (5−XOR 64−bit) and the biggest relative difference is 82% for the hardest task (8−XOR 64−bit).

When comparing training time, the results also show that the relative difference is smaller on the hardest tasks. The relative time difference for the 64−bit 5−XOR is 226%, and for the 64−bit 8−XOR it is 116%. Also worth noting, when comparing total runtime on hard tasks (see Figures 8.3 and 8.4) we can see that the gap becomes even smaller, which is another argument that the 3D CNN is better suited for harder tasks.

Chapter 9

Evaluation

The deep learning methods were already evaluated in subsections 8.3.2 and 8.3.3 and section 8.4, while only focusing on the hardware and limited CRP aspect with sufficient training time. In this chapter, all three methods are evaluated from a theoretical and practical perspective.

When evaluating different methods, there are three aspects to consider:
1) CRP amount: how many CRPs are available to the attacker?
2) Computing power: how many threads can the attacker run in parallel?
3) Time: how important is it to break the PUF as soon as possible?

Depending on if one of these resources is insufficient, the attacker can choose between three different methods. In practice, 3) can only be the most important aspect if both 1) and 2) are sufficient, because not having enough CRPs can fail the attack, and not having enough computing power can slow down pre-processing and training. Therefore, when 1) or 2) are insufficient, the attacker should choose a method according to these limitations.

9.1 3D CNN vs. MLP vs. logistic regression

This section determines the best attack choice based on the three aspects mentioned above.

9.1.1 Theoretically optimal

When an attacker has collected many billions of CRPs and has access to a very powerful computer, e.g. a supercomputer, the best choice could theoretically be the logistic regression, because it is the fastest. As Wisiol et al. [93] have shown, LR broke the 10−XOR in 41 minutes, whereas the

MLP needed 291 minutes. Even though LR's success rate was 10% less, in case of failure, the attacker could start the attack over much more quickly and therefore save time.

9.1.2 Modern household computers

When the attacker is using a modern household computer, LR is only the best choice for small XPUF designs because it requires much more CRPs which cannot be held in RAM at once. Also, it would be hard to train on many billions of CRPs, one billion 64− bit CRPs saved in a text file require 62 gigabytes of disk space.

When the attacker has collected sufficiently enough CRPs, the MLP can be the best choice for designs up to 7−XOR, but there were problems with the 8−XOR as reported in section 8.2.

The 3D CNN method seems to be the best choice when the collected data is not sufficient for the MLP and LR and theoretically for the (8+)−XOR designs on normal computers but (9+)-XOR still need empirical results.

Chapter 10

Future Work

This chapter proposes possible improvements to both methods and describes which attacks on which designs and architectures could be interesting in the future.

10.1 MLP optimization and side channels

The MLP could be optimized similarly to the 3D CNN (see Chapter 7) and could require even fewer CRPs. The first steps could be to test the ridge activity regularization and the logistic loss function with sum reduction.

In 2013, Mahmoud et al. [117] proposed a 'Side Channel Attack'. The number of response bits of every arbiter in a design that is one or zero can be known by the attacker and therefore this information can additionally be used in an attack.

10.2 3D CNN cube size and nD CNN

As discovered in subsection 7.2.7, increasing the cube size can improve the success rate and speed. The method was optimized to work best without filling empty cube values. There could still be room for optimizing further. The method may respond better to a change in filters, dense layer units, batch size, kernel initialization, and so forth.

Because 2D works better than 1D and 3D works better than 2D, nD could theoretically work even better than 3D. Unfortunately, the $n-$dimensional CNN could not be optimized in time for this thesis.

10.3 3D CNN attacks on (9+)-XORs

Due to a lack of computing power and time constraints, attacks on the (9+)−XORs have not been conducted with the 3D CNN. During the writing of this thesis, Wisiol et al. [93] released results of the MLP on the 64−bit 10 & 11−XOR. These designs have not been broken by any deep learning method before. It would be interesting to see how well the 3D CNN performs on these designs, especially because the relative CRP difference of these two methods increases on harder tasks (as shown in section 8.4).

10.4 3D CNN attacks on other Strong PUFs

Neural network attacks also have had success on other Strong PUFs than the XPUF. The 'Splitting Attack' by Wisiol et al. [118] is the best attack on Interpose PUFs [119]. Alkatheiri and Zhuang [120] have attacked the Feed-Forward Arbiter PUF [121] successfully. Rührmair et al. [2] broke the Lightweight Secure PUF [71]. The Bistable Ring PUF [49] has not been broken by deep learning methods yet and there are new Strong PUF architectures released every year. It would be interesting to see the performance of the 3D CNN on all of these Strong PUFs.

10.5 Training on GPUs

Qualitative GPUs have become more affordable and from own experience, methods can be trained about ten times faster than on CPUs. But modern GPUs mostly only have four or six gigabytes of RAM. This works well on small designs but makes chunking on big designs inevitable.

Chapter 11

Conclusion

In this thesis, new deep learning methods were developed as an alternative to the multilayer perceptron. LSTMs have not performed well. The nD CNN was not optimized enough to be a viable option but could work better than the 3D CNN in theory. This thesis introduced the first CNN to break an XPUF, the best being a 3D CNN method. It was compared to an MLP, the best method to date [86], requiring $19\% - 82\%$ fewer CRPs. Considering training time it took 20% less for one design but more for the other designs by up to 269%. The method could be optimized by increasing the cube size and there are also possible improvements to be made to the MLP. The MLP was trained on pypuf data and the results were compared to the reported CRP amounts on another data source [86], and the data requirement was higher for every design. The logistic regression [93], MLP [86], and 3D CNN are all viable choices, depending on how many CRPs, how much computing power and how much training time is available to the attacker. The 3D CNN misses important results for the (9+)−XOR. Judging by the presented results, CNNs have shown potential to break Strong PUFs and should be considered for deep learning attacks in the future.

I would like to thank my advisors Nadine Sarah Schüler and Prof. Dr. Dr. Ulrich Rührmair and my professor Prof. Dr. Peer Kröger for teaching me and for enabling me to write my master thesis at the DBS chair of the Institute of Informatics at the LMU Munich.

Bibliography

[1] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[2] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 237–249, 2010.

[3] U. Rührmair and D. E. Holcomb, "Pufs at a glance," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[4] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 148–160, 2002.

[5] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *International workshop on cryptographic hardware and embedded systems*, pp. 63–80, Springer, 2007.

[7] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly puf protecting ip on every fpga," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 67–70, IEEE, 2008.

[8] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant puf design based on non-linear voltage transfer characteristics," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 653–658, IEEE, 2015.

[9] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong puf: Possible or a pipe dream?," in *2016 IEEE international symposium on hardware oriented security and trust (HOST)*, pp. 19–24, IEEE, 2016.

[10] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive puf analysis," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 30–38, IEEE, 2013.

[11] P. Koeberl, Ü. Kocabaş, and A.-R. Sadeghi, "Memristor pufs: a new generation of memory-based physically unclonable functions," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 428–431, IEEE, 2013.

[12] S. Katzenbeisser, Ü. Kocabaş, V. Van Der Leest, A.-R. Sadeghi, G.-J. Schrijen, and C. Wachsmann, "Recyclable pufs: Logically reconfigurable pufs," *Journal of Cryptographic Engineering*, vol. 1, no. 3, pp. 177–186, 2011.

[13] J. Delvaux, "Machine-learning attacks on polypufs, ob-pufs, rpufs, lhs-pufs, and puf–fsms," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2043–2058, 2019.

[14] B. Chatterjee, D. Das, S. Maity, and S. Sen, "Rf-puf: Enhancing iot security through authentication of wireless nodes using in-situ machine learning," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 388–398, 2018.

[15] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, "Memristor puf—a security primitive: Theory and experiment," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 222–229, 2015.

[16] J.-L. Zhang, G. Qu, Y.-Q. Lv, and Q. Zhou, "A survey on silicon pufs and recent advances in ring oscillator pufs," *Journal of computer science and technology*, vol. 29, no. 4, pp. 664–678, 2014.

[17] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong pufs," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 39, no. 10, pp. 2138–2151, 2019.

[18] R. Kumar and W. Burleson, "On design of a highly secure puf based on non-linear current mirrors," in *2014 IEEE international symposium on hardware-oriented security and trust (HOST)*, pp. 38–43, IEEE, 2014.

[19] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 37–49, 2014.

[20] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for sram pufs," in *2009 IEEE international symposium on information theory*, pp. 2101–2105, IEEE, 2009.

[21] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic pufs from flip-flops on reconfigurable devices," in *3rd Benelux workshop on information and system security (WISSec 2008)*, vol. 17, p. 2008, 2008.

[22] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on pufs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, 2016.

[23] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 137–142, IEEE, 2013.

[24] B. C. Grubel, B. T. Bosworth, M. R. Kossey, H. Sun, A. B. Cooper, M. A. Foster, and A. C. Foster, "Silicon photonic physical unclonable function," *Optics Express*, vol. 25, no. 11, pp. 12710–12721, 2017.

[25] R. Horstmeyer, B. Judkewitz, I. M. Vellekoop, S. Assawaworrarit, and C. Yang, "Physical key-protected one-time pad," *Scientific reports*, vol. 3, no. 1, pp. 1–6, 2013.

[26] J. D. Buchanan, R. P. Cowburn, A.-V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan, "'fingerprinting'documents and packaging," *Nature*, vol. 436, no. 7050, pp. 475–475, 2005.

[27] G. DeJean and D. Kirovski, "Rf-dna: Radio-frequency certificates of authenticity," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 346–363, Springer, 2007.

[28] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2008.

[29] P. Simons, E. van der Sluis, and V. van der Leest, "Buskeeper pufs, a promising alternative to d flip-flop pufs," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 7–12, IEEE, 2012.

[30] Y. Eliezer, U. Ruhrmair, N. Wisiol, S. Bittner, and H. Cao, "Exploiting structural nonlinearity of a reconfigurable multiple-scattering system," *arXiv preprint arXiv:2208.08906*, 2022.

[31] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "Reliable and modeling attack resistant authentication of arbiter puf in fpga implementation with trinary quadruple response," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1109–1123, 2018.

[32] C. Q. Liu, Y. Cao, and C. H. Chang, "Acro-puf: A low-power, reliable and aging-resilient current starved inverter-based ring oscillator physical unclonable function," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 12, pp. 3138–3149, 2017.

[33] R. A. John, N. Shah, S. K. Vishwanath, S. E. Ng, B. Febriansyah, M. Jagadeeswararao, C.-H. Chang, A. Basu, and N. Mathews, "Halide perovskite memristors as flexible and reconfigurable physical unclonable functions," *Nature Communications*, vol. 12, no. 1, pp. 1–11, 2021.

[34] K. Rosenfeld, E. Gavas, and R. Karri, "Sensor physical unclonable functions," in *2010 IEEE international symposium on hardware-oriented security and trust (HOST)*, pp. 112–117, IEEE, 2010.

[35] G. S. Rose, J. Rajendran, N. McDonald, R. Karri, M. Potkonjak, and B. Wysocki, "Hardware security strategies exploiting nanoelectronic circuits," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 368–372, IEEE, 2013.

[36] J. Tang, R. Karri, and J. Rajendran, "Securing pressure measurements using sensorpufs," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1330–1333, IEEE, 2016.

[37] Q. Chen, G. Csaba, X. Ju, S. B. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, and U. Rührmair, "Analog circuits for physical cryptography," in *Proceedings of the 2009 12th International symposium on integrated circuits*, pp. 121–124, IEEE, 2009.

[38] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, and U. Ruehrmair, "Circuit-based approaches to simpl systems," *Journal of Circuits, Systems, and Computers*, vol. 20, no. 01, pp. 107–123, 2011.

[39] Y. Gao, C. Jin, J. Kim, H. Nili, X. Xu, W. Burleson, O. Kavehei, M. van Dijk, D. C. Ranasinghe, and U. Rührmair, "Efficient erasable pufs from programmable logic and memristors," *Cryptology ePrint Archive*, 2018.

[40] R. Horstmeyer, S. Assawaworrarit, U. Ruhrmair, and C. Yang, "Physically secure and fully reconfigurable data storage using optical scattering," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 157–162, IEEE, 2015.

[41] L. Orosa, U. Rührmair, A. G. Yaglikci, H. Luo, A. Olgun, P. Jattke, M. Patel, J. Kim, K. Razavi, and O. Mutlu, "Spyhammer: Using rowhammer to remotely spy on temperature," *arXiv preprint arXiv:2210.04084*, 2022.

[42] U. Rührmair, "Simpl systems as a keyless cryptographic and security primitive," in *Cryptography and Security: From Theory to Applications*, pp. 329–354, Springer, 2012.

[43] M. Sauer, P. Raiola, L. Feiten, B. Becker, U. Rührmair, and I. Polian, "Sensitized path puf: A lightweight embedded physical unclonable function," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 680–685, IEEE, 2017.

[44] U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, "Towards electrical, integrated implementations of simpl systems," in *IFIP International Workshop on Information Security Theory and Practices*, pp. 277–292, Springer, 2010.

[45] C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, and M. Stutzmann, "Random pn-junctions for physical cryptography," *Applied Physics Letters*, vol. 96, no. 17, p. 172103, 2010.

[46] P. Lugli, A. Mahmoud, G. Csaba, M. Algasinger, M. Stutzmann, and U. Rührmair, "Physical unclonable functions based on crossbar arrays for cryptographic applications," *International journal of circuit theory and applications*, vol. 41, no. 6, pp. 619–633, 2013.

[47] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Rührmair, "Application of mismatched cellular nonlinear networks for physical cryptography," in *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pp. 1–6, IEEE, 2010.

[48] U. Ruehrmair, M. Stutzmann, J. Finley, C. Jirauschek, G. Csaba, P. Lugli, E. Biebl, R. Dietmueller, K. Mueller, and H. Langhuth, "Method for security purposes," July 5 2012. US Patent App. 13/250,534.

[49] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 134–141, IEEE, 2011.

[50] G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Rührmair, "On-chip electric waves: An analog circuit approach to physical uncloneable functions," *Cryptology ePrint Archive*, 2009.

[51] H. Langhuth, S. Frédérick, M. Kaniber, J. J. Finley, and U. Rührmair, "Strong photoluminescence enhancement from colloidal quantum dot near silver nano-island films," *Journal of fluorescence*, vol. 21, no. 2, pp. 539–543, 2011.

[52] U. Rührmair, "Simpl systems: On a public key variant of physical unclonable functions," *Cryptology ePrint Archive*, 2009.

[53] U. Rührmair, "Simpl systems, or: can we design cryptographic hardware without secret key information?," in *International Conference on Current Trends in Theory and Practice of Computer Science*, pp. 26–45, Springer, 2011.

[54] C. Jin, W. Burleson, M. van Dijk, and U. Rührmair, "Programmable access-controlled and generic erasable puf design and its applications," *Journal of Cryptographic Engineering*, pp. 1–20, 2022.

[55] U. Rührmair, C. Hilgers, S. Urban, A. Weiershäuser, E. Dinter, B. Forster, and C. Jirauschek, "Optical pufs reloaded," *Cryptology ePrint Archive*, 2013.

[56] C. Jin, X. Xu, W. Burleson, U. Rührmair, and M. van Dijk, "Playpuf: programmable logically erasable pufs for forward and backward secure key management," *Cryptology ePrint Archive*, 2015.

[57] C. Jin, W. Burleson, M. van Dijk, and U. Rührmair, "Erasable pufs: formal treatment and generic design," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, pp. 21–33, 2020.

[58] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba, "Applications of high-capacity crossbar memories in cryptography," *IEEE Transactions on Nanotechnology*, vol. 10, no. 3, pp. 489–498, 2010.

[59] U. Rührmair, J. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J. J. Finley, and W. P. Burleson, "Virtual proofs of reality and their physical implementation," in *2015 IEEE Symposium on Security and Privacy*, pp. 70–85, IEEE, 2015.

[60] F. Pavanello, I. O'Connor, U. Rührmair, A. C. Foster, and D. Syvridis, "Recent advances in photonic physical unclonable functions," in *2021 IEEE European Test Symposium (ETS)*, pp. 1–10, IEEE, 2021.

[61] M. Majzoobi, F. Koushanfar, and S. Devadas, "Fpga puf using programmable delay lines," in *2010 IEEE international workshop on information forensics and security*, pp. 1–6, IEEE, 2010.

[62] Y. Cao, L. Zhang, C.-H. Chang, and S. Chen, "A low-power hybrid ro puf with improved thermal stability for lightweight applications," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 7, pp. 1143–1147, 2015.

[63] J. Rajendran, G. S. Rose, R. Karri, and M. Potkonjak, "Nano-ppuf: A memristor-based security primitive," in *2012 IEEE Computer Society Annual Symposium on VLSI*, pp. 84–87, IEEE, 2012.

[64] M. T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "Aro-puf: An aging-resistant ring oscillator puf design," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[65] K. Xiao, M. T. Rahman, D. Forte, Y. Huang, M. Su, and M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of sram-puf," in *2014 IEEE international symposium on hardware-oriented security and trust (HOST)*, pp. 101–106, IEEE, 2014.

[66] K. Lofstrom, W. R. Daasch, and D. Taylor, "Ic identification circuit using device mismatch," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*, pp. 372–373, IEEE, 2000.

[67] B. Škorić, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," in *International Conference on Applied Cryptography and Network Security*, pp. 407–422, Springer, 2005.

[68] C. Herder, L. Ren, M. Van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2016.

[69] M. Arapinis, M. Delavar, M. Doosti, and E. Kashefi, "Quantum physical unclonable functions: Possibilities and impossibilities," *Quantum*, vol. 5, p. 475, 2021.

[70] R. Maes, A. V. Herrewege, and I. Verbauwhede, "Pufky: A fully functional puf-based cryptographic key generator," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 302–319, Springer, 2012.

[71] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure pufs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–673, IEEE, 2008.

[72] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable pufs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 2, no. 1, pp. 1–33, 2009.

[73] P. Tuyls, G.-J. Schrijen, B. Škorić, J. v. Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 369–383, Springer, 2006.

[74] P. Tuyls and B. Škorić, "Strong authentication with physical unclonable functions," in *Security, Privacy, and Trust in Modern Data Management*, pp. 133–148, Springer, 2007.

[75] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu, "The dram latency puf: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity dram devices," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 194–207, IEEE, 2018.

[76] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "Dram-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1085–1097, 2016.

[77] X. Xi, H. Zhuang, N. Sun, and M. Orshansky, "Strong subthreshold current array puf with 2 65 challenge-response pairs resilient to machine learning attacks in 130nm cmos," in *2017 Symposium on VLSI Circuits*, pp. C268–C269, IEEE, 2017.

[78] X. Xi, G. Li, Y. Wang, and M. Orshansky, "A provably secure strong puf based on lwe: Construction and implementation," *IEEE Transactions on Computers*, 2022.

[79] C. Jin, C. Herder, L. Ren, P. H. Nguyen, B. Fuller, S. Devadas, and M. Van Dijk, "Fpga implementation of a cryptographically-secure puf based on learning parity with noise," *Cryptography*, vol. 1, no. 3, p. 23, 2017.

[80] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," in *Introduction to Hardware Security and Trust*, pp. 65–102, Springer, 2012.

[81] U. Rührmair, J. Sölter, and F. Sehnke, "On the foundations of physical unclonable functions," *Cryptology ePrint Archive*, 2009.

[82] U. Rührmair, "Towards secret-free security," *Cryptology ePrint Archive*, 2019.

[83] U. Ruhrmair, "Sok: Towards secret-free security," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, pp. 5–19, 2020.

[84] U. Rührmair, "Secret-free security: A survey and tutorial," *Journal of Cryptographic Engineering*, pp. 1–26, 2022.

[85] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," *Towards Hardware-Intrinsic Security*, pp. 3–37, 2010.

[86] K. T. Mursi, B. Thapaliya, Y. Zhuang, A. O. Aseeri, and M. S. Alkatheiri, "A fast deep learning method for security vulnerability study of xor pufs," *Electronics*, vol. 9, no. 10, p. 1715, 2020.

[87] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[88] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[89] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[90] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[91] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[92] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pp. 17–31, Springer, 2015.

[93] N. Wisiol, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural-network-based modeling attacks on xor arbiter pufs revisited.," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 555, 2021.

[94] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on xor pufs for resource-constraint internet of things," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, pp. 49–56, IEEE, 2018.

[95] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty, "Deep learning based model building attacks on arbiter puf compositions.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 566, 2019.

[96] nils wisiol, "https://github.com/nils-wisiol/pypuf," 2021.

[97] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE transactions on information forensics and security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[98] U. Rührmair and M. van Dijk, "Pufs in security protocols: Attack models and security evaluations," in *2013 IEEE symposium on security and privacy*, pp. 286–300, IEEE, 2013.

[99] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 476–492, Springer, 2014.

[100] U. Rührmair, U. Schlichtmann, and W. Burleson, "Special session: How secure are pufs really? on the reach and limits of recent puf attacks," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–4, IEEE, 2014.

[101] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, and W. Burleson, "Power and timing side channels for pufs and their efficient exploitation," *Cryptology ePrint Archive*, 2013.

[102] M. van Dijk and U. Rührmair, "Protocol attacks on advanced puf protocols and countermeasures," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[103] U. Rührmair, "On the security of puf protocols under bad pufs and pufs-inside-pufs attacks," *Cryptology ePrint Archive*, 2016.

[104] U. Rührmair, H. Busch, and S. Katzenbeisser, "Strong pufs: models, constructions, and security proofs," in *Towards hardware-intrinsic security*, pp. 79–96, Springer, 2010.

[105] U. Rührmair, "Oblivious transfer based on physical unclonable functions," in *International Conference on Trust and Trustworthy Computing*, pp. 430–440, Springer, 2010.

[106] R. Ostrovsky, A. Scafuro, I. Visconti, and A. Wadia, "Universally composable secure computation with (malicious) physically uncloneable functions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 702–718, Springer, 2013.

[107] D. Dachman-Soled, N. Fleischhacker, J. Katz, A. Lysyanskaya, and D. Schröder, "Feasibility and infeasibility of secure computation with malicious pufs," in *Annual Cryptology Conference*, pp. 405–420, Springer, 2014.

[108] U. Rührmair and M. v. Dijk, "Practical security analysis of puf-based two-player protocols," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 251–267, Springer, 2012.

[109] U. Rührmair and M. van Dijk, "On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols," *Journal of Cryptographic Engineering*, vol. 3, no. 1, pp. 17–28, 2013.

[110] U. Rührmair, "Physical turing machines and the formalization of physical cryptography," *Cryptology ePrint Archive*, 2011.

[111] F. Kappelhoff, R. Rasche, D. Mukhopadhyay, and U. Rührmair, "Strong puf security metrics: Response sensitivity to small challenge perturbations," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pp. 1–10, IEEE, 2022.

[112] memmap, "https://numpy.org/doc/stable/reference/generated/numpy.memmap.html," 2021.

[113] pvjosue, "https://github.com/pvjosue/pytorch_convnd," 2021.

[114] keras, "https://keras.io/api/," 2021.

[115] C. Van Dusen, "Methods to prevent overwriting and solve ill-posed problems in statistics: Ridge regression and lasso," *Preprint submitted to Colorado College Department of Mathematics September*, vol. 16, 2016.

[116] kmursi, "https://github.com/kmursi/ml_attack_xor_puf," 2021.

[117] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong pufs.," *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 632, 2013.

[118] N. Wisiol, C. Mühl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Rührmair, "Splitting the interpose puf: A novel modeling attack strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–120, 2020.

[119] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose puf: Secure puf design against state-of-the-art machine learning attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 243–290, 2019.

[120] M. S. Alkatheiri and Y. Zhuang, "Towards fast and accurate machine learning attacks of feed-forward arbiter pufs," in *2017 IEEE Conference on Dependable and Secure Computing*, pp. 181–187, IEEE, 2017.

[121] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.