# Attacking Single-Cycle Ciphers on Modern FPGAs

## featuring Explainable Deep Learning

Mustafa Khairallah[1,2] and Trevor Yap[1]

[1] Nanyang Technological University, Singapore, Singapore
[2] Dept. of Electrical and Information Technology, Lund University, Lund, Sweden
{m.khairallah,trevor.yap}@ntu.edu.sg

**Abstract.** In this paper, we revisit the question of key recovery using side-channel analysis for unrolled, single-cycle block ciphers. In particular, we study the Princev2 cipher. While it has been shown vulnerable in multiple previous studies, those studies were performed on side-channel friendly ASICs or older FPGAs (*e.g.,* Xilinx Virtex II on the SASEBO-G board), and using mostly expensive equipment. We start with the goal of exploiting a cheap modern FPGA and board using power traces from a cheap oscilloscope. Particularly, we use Xilinx Artix 7 on the Chipwhisperer CW305 board and PicoScope 5000A, respectively.
We split our study into three parts. First, we show that the new set-up still exhibits easily detectable leakage, using a non-specific $t$-test. Second, we replicate attacks from older FPGAs. Namely, we start with the attack by Yli-Mäyry *et al.*, which is a simple chosen plaintext correlation power analysis attack using divide and conquer. However, we demonstrate that even this simple, powerful attack does not work, demonstrating a peculiar behavior. We study this behavior using a stochastic attack that attempts to extract the leakage model, and we show that models over a small part of the state are inconsistent and depend on more key bits than what is expected. We also attempt classical template attacks and get similar results.
To further exploit the leakage, we employ deep learning techniques and succeed in key recovery, albeit using a large number of traces. We perform the explainability technique called Key Guessing Occlusion (KGO) to detect which points the neural networks exploit. When we use these points as features for the classical template attack, although it did not recover the secret key, its performance improves compared to other feature selection techniques.

**Keywords:** Deep Learning · Side-Channel Analysis · Princev2 · Low Latency · FPGA

## 1 Introduction

Block ciphers with unrolled implementations have gained popularity over the recent years with the emergence of ciphers such as Prince [7,8] (with both its

versions), Bipbip [4], SCARF [10], Speedy [14], Orthros [3], Gleeok [2], and several others. These ciphers are typically designed with smaller security margins and are popular in applications such as memory and/or cache encryption. In these applications, the top priority is to perform the encryption and decryption as fast as possible. Unlike older and more classical designs, these block ciphers are typically implemented such that they take only one clock cycle using dedicated hardware.

This strategy presents a challenge for side-channel adversaries. In fact, unrolling the implementation of a cipher (performing several rounds of the block cipher in one clock cycle using dedicated hardware) provides some basic resistance against side-channel attacks [6]. However, advanced attacks are still feasible [19,20,16,21].[3] Nonetheless, the demonstrated attacks have been significantly more complicated than attacks on classical unprotected ciphers.

While investigating these attacks, we have noticed that the majority of them were performed on FPGAs that are two decades old, with the exception of [16], where the attack was performed using a dedicated side-channel friendly ASIC. More importantly, the traces were acquired using expensive equipment that is typically found in well-established labs. More and more attacks on other ciphers are being performed using cheaper equipment that can even potentially be operated in the field. Thus, we set out to investigate the vulnerability of the Princev2 cipher to side-channel analysis using cheaper setups. We use the Chipwhisperer CW305 board with Xilinx Artix 7 FPGA and the PicoScope portable oscilloscope for acquiring power traces. Compared to the oscilloscopes used in previous experiments, the PicoScope has a significantly lower sampling frequency and cost. It is hard to tell apriori whether the sampling frequency is even sufficient for the task at hand. Typically, we would use the Nyquist criterion to determine if the sampling frequency is sufficient relative to the measured signal. However, in the case of unrolled implementations, the measured signal is not related to the clock frequency but rather to the delay of the individual gates, as we hope to be able to isolate different parts of the execution.

At the end of our study, we answer our research question in the affirmative: even with this cheap setup, the implementation is still vulnerable to side-channel attacks with reasonable resources. However, the road to this answer is more interesting than the destination. Quickly, we found out that the traces still exhibit observable leakage using Test Vector Leakage Assessment (TVLA), with very few traces. However, when applying known attacks, particularly the correlation power analysis in [19], we observe a peculiar behavior. While the attack converged, indicating a uniquely recovered key, it turned out the recovered key was wrong. This behavior was consistent with different parameters and with different leakage models: Hamming distance, Hamming weight, and Identity leakage models. To understand what is happening, we diverged our attention to a model extraction profiling attack, namely the stochastic attack [17]. The stochastic attack requires training on known keys and assumes only a polynomial leakage model. For instance, a Hamming weight leakage model is a polynomial leakage

[3] This is a comprehensive but non-exhaustive list.

model where all the bits of the targeted value contribute linearly and equally to the leakage. However, we again observed that this polynomial model does not hold. When we train with a fixed known key, we see that the model changes depending on the value of the key, and when we train with many random keys, we get inconsistent models.

Subsequently, we opted to use more modern and advanced tools: Deep-Learning-based Side-Channel Analysis (DLSCA). It has been shown that DLSCA has obtained significant performance over classical side-channel attacks [15,5]. Specifically, in the presence of jitter/desynchronized and masking countermeasures, it has been demonstrated that DLSCA can recover the secret key without much preprocessing needed [9]. In fact, using a single-cycle cipher introduces natural desynchronization [6]. Therefore, DLSCA is the most suitable tool for exploiting the leakage within the traces. To the best of our knowledge, this is the first time DLSCA has been applied to traces of an unrolled, single-cycle block cipher. Furthermore, we utilize an explainability technique called Key Guessing Occlusion (KGO) [18] on the well-performing Deep Neural Network (DNN). This allows us to understand which timestamp/sample points the DNN is exploiting. We further use these relevant sample points for template attacks. The performance of the classical template attack improves compared to other feature selection tools or without any feature selection tools.

Part of the outcome of our study is the dataset that can be used to study advanced template attacks. To help further investigation, the dataset can be found at

`https://shorturl.at/yLIMR`

*Outline.* The paper is organized as follows: In Section 2, we give a brief description of the Princev2 cipher and the targeted operation. In Section 3, we show the vulnerability of the implementation to TVLA. In Section 4, we apply correlation power analysis and show the odd behavior that the traces exhibit. While in Section 5, we study this behavior in more detail using the stochastic attack. Next, the use of DLSCA is explored in Section 6. Finally, we conclude the paper in Section 7.

## 2 The Princev2 block cipher

Princev2 [8] is an update to the design of Prince [7]. The only difference between the two ciphers is in the key schedule. Thus, side-channel attacks, at least on the first round, are still applicable. Princev2 has a 64-bit block size and 128-bit key size. It is designed to have very small encryption critical path. The key is split into two 64-bit halves, and the round keys alternate between these two halves. It consists of 10 iterative rounds and one middle round. The first five rounds are of the form

$$\text{AddRoundKey} \rightarrow \text{Sbox} \rightarrow \text{LinearLayer}$$

while the last 5 rounds are of the form

$$\text{LinearLayer} \rightarrow \text{Sbox}^{-1} \rightarrow \text{AddRoundKey}.$$

The linear layer is an involution, so its inverse is the same as itself. Thus, the last 5 rounds are essentially the inverse of the first 5 rounds if the round keys were the same. The middle round is of the form

$$\text{Sbox} \rightarrow \text{LinearLayer} \rightarrow \text{Sbox}^{-1}.$$

The block cipher is designed with this structure so that the same circuit can be used for both encryption and decryption. In order to recover the full secret key, we need to recover two different round keys. In our attacks, we target the first two round keys. In particular, our attacks target the output of the operation:

$$Sbox(K \oplus P),$$

where $K$ and $P$ are 4-bit variables. In the classical attacks (Section 4 and 5), we target one nibble at a time. In other words, we need one set of traces for each targetted nibble. In the DLSCA (Section 6), we collect one set of traces which can be used to attack different nibbles.

## 3 Experimental set-up and detecting non-specific leakage

For our experiments, we use the Artix 7 xc7a35t FPGA, on the ChipWhisperer CW305 development board. We use PicoScope PS5000a as our oscilloscope. We use two oscilloscope channels, one for triggering, set at a range of 20V, and one for measurement, set at a range of 200mV. The FPGA runs at 10 MHz and the oscilloscope is set to time-base 2 (4ns sampling interval). This implies that each clock cycle is represented in the trace approximately 25 samples. In order to verify that the setup gives meaningful traces and that we can detect leakage at all, we perform Welsh's $t$-test using the TVLA framework. We used 1000 traces that randomly switch between random and fixed plaintexts, with 483 traces with a fixed plaintext and 517 random plaintexts. As expected, we can easily detect non-specific leakage with $t > 20$. The result is shown in Figure 1. Despite this positive and expected result, it is well known that TVLA does not give an indication of how the observable leakage can be exploited in a meaningful attack. In the remainder of the paper, we will attempt to exploit the leakage, first using known attacks and then using DLSCA.

Several side-channel attacks on single-cycle ciphers have been proposed over the years. [19,20,16,21] is a comprehensive (but non-exhaustive) list of examples. The main challenge in attacking single-cycle ciphers is their unrolled combinational circuit. This means that the measured power includes the switching activity from all the gates of the combinational circuit. On the other hand, side-channel attacks achieve the best results when we are able to perform attacks in a divide-and-conquer manner: the attacker targets a small part of the computation at a time, *e.g.* one substitution box (sbox) or a handful of sboxes. In this

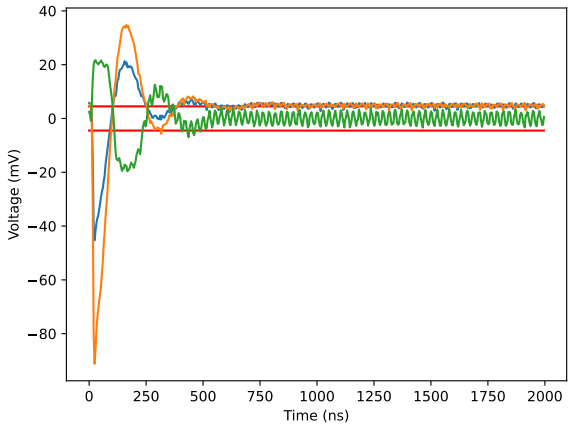



**Fig. 1.** TVLA result against Princev2. Green is the $t$ value, blue is the mean of all the fixed plaintext traces and orange is the mean of all the random plaintext traces. The red lines are set at $\pm 4.5$, the widely accepted standard for significant leakage.

framework, everything that is simultaneous to the targeted operation is seen as algorithmic noise. Thus, if we target one sbox in a single-cycle implementation, almost the whole cipher contributes to the noise. With a high-resolution, powerful oscilloscope, we can somehow reduce this effect by having more samples per cycle and somewhat having a better resolution in identifying the logic propagation patterns in the trace. Our setup is, however, geared towards studying the other end of this spectrum: a cheap low-resolution oscilloscope, a cheap board, and relatively high-frequency implementation. While the TVLA result shows significant observable leakage, it also shows that during the first clock cycle after the trigger, the power consumption is quite high with significant peaks.

## 4 Correlation Power Analysis

Yli-Mäyry *et al.* [19] proposed a CPA attack on prince using the Tektronix DPO7254 oscilloscope with $0.2ns$ sampling interval and Xilinx Virtex II FPGA. Information on the clock frequency does not seem to be reported. They were successful in recovering one round key with around $50,000$ traces. Yli-Mäyry *et al.* [20] proposed a new technique for reducing the algorithmic noise. Instead of collecting one set of traces with random plaintexts, they collected a separate set of traces for each targeted sbox, where the plaintext nibble corresponding to that sbox varies randomly, and the rest are set to zeros. This reduces the switching activity and algorithmic noise in the first sbox layer. They used the same oscilloscope as [19], but set to an even higher sampling frequency: $0.025ns$ sampling interval. They were able to find the first round key with $16 \times 500$ traces.

This is significantly better than [19]. They also attack the second round key but the number of traces increases by $10 - 40\times$. The authors seem to attribute the improved performance of [20] compared to [19] solely to the new strategy. While the strategy plays a major role, it should not be missed that the sampling frequency is increased by 8 folds. We also note that both papers use the Sasebo-G board, with Xilinx Virtex II FPGA[4]. The Sasebo-G board is heavily engineered to ease side-channel analysis. The Virtex II board is quite outdated in 2024.

Moos [16] studied the problem of attacking Princev2 on a custom $40nm$ side-channel-friendly ASIC. He performed two studies. The first is based on dynamic power analysis, using Teledyne LeCroy WaveRunner 8254M and $0.025ns$ sampling interval. He also used a high-frequency electromagnetic probe to get high-resolution power traces. In this case, he was able to recover the key with about $500,000$ traces. In the second study, he targeted static power. He used Teledyne LeCroy HRO 66Zi and $0.5ns$ sampling interval. He was able to recover most of the first round key using an estimate of $16 \times 50,000$ traces.

While these attacks are powerful, they are also quite costly. Each of the three oscilloscopes used in these experiments costs upwards of 10k Euros. Besides, these oscilloscopes are only useful for attacks that can be performed in a lab, as they lack portability. The oscilloscope we use (PicoScope 5000a) is portable, can be powered by a laptop and costs $<$ 2.5k Euros. We also needed an SMA-BNC cable to interface between the oscilloscope and the CW305 board, which cost about 30 Euros. We set to find out if the previous attacks are effective in this cheap setting. We performed the CPA of [20]. A random fixed key is generated at the beginning of the experiment and loaded to the FPGA. Next, we select which nibble we want to target and set all other plaintext nibbles to 0. When targeting the nibble at index $i$, the plaintext for trace $j$ is on the form:

$$0x000000000000000r_j << (4i)$$

where $r_j$ is a random nibble.

We collected $100,000$ traces per nibble and observed that the CPA attack converges towards one key guess. It seemed to converge with the maximum correlation coefficient $\approx 0.08$. However, upon further inspection, it seemed that the attack was always converging but not the correct value. For instance, the results for one nibble for each key guess are displayed in Figure 2, with 8 as the key guess with the highest correlation coefficient. The figure depicts the first two clock cycles and corresponds to the first two peaks in the power traces in Figure 1. There is also a clear ordering of the guess throughout the computation. Figure 3 shows the maximum correlation coefficient for each key nibble guess. Showing the correct guess should have been 0, but the attack guessed 8.

We tried this experiment numerous times and could not identify any pattern for why this behavior was happening. It seems that the wrongly guessed key value for the same nibble location and the same correct key nibble value also

[4] [20] lists the board as Sasebo-G II, but Sasebo-G II came with Virtex 5, not Virtex II.

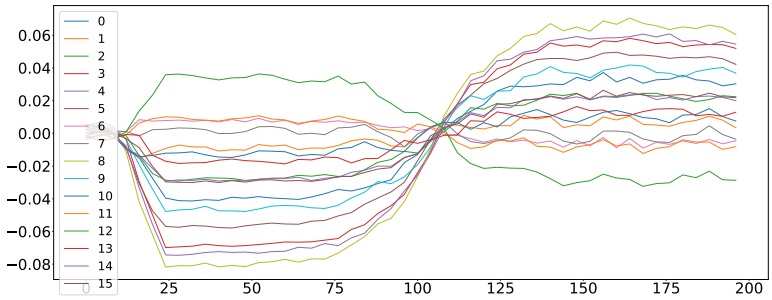



**Fig. 2.** Correlation coefficient for a key nibble in the first two clock cycles.

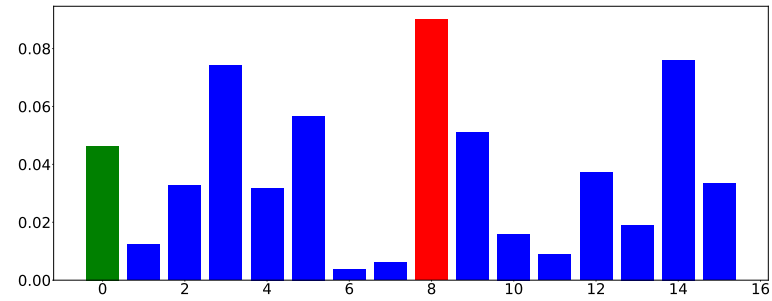



**Fig. 3.** Maximum correlation coefficient for a key nibble in the first two clock cycles. Red is the guessed key nibble, while green is the correct key nibble.

change by changing the full key, while it remains the same for the same key. In other words, consider the two round keys

$$0x0ae6568fd3cfa120$$

$$0xf27667ef4441a520$$

with plaintexts

$$0x0000000000000000r_j.$$

Both keys have a targeted key nibble of value 0, but during the attack, each key will lead to a different wrong guess. Figures 2 and 3 have been generated using the Hamming distance model at the output of the first sbox layer. However, the same behavior was observed using the Hamming weight model.

## 5 Stochastic Attacks

To get one step closer to understanding the root cause of the behavior observed in the CPA attack, we perform a stochastic attack. During CPA, we assume a specific leakage model, such as Hamming weight or Hamming distance. During the stochastic attack, we assume a polynomial leakage model, typically linear. In particular, if the target variable is

$$X = S(K \oplus P)$$

and $X$ consists of 4 bits $x_3, x_2, x_1$ and $x_0$, then we assume the model

$$L(X) = \sum_{i=0}^{15} a_i x_0^{i_0} x_1^{i_1} x_2^{i_2} x_3^{i_3}$$

where $a_i \in \mathbb{R}$ and $i_3 i_2 i_1 i_0$ is the bit representation of the integer $i$ for all $0 \leq i \leq 15$. If $a_i = 0$ for every $i \notin \{0, 1, 2, 4, 8\}$, then the model is linear. The stochastic attack is a model extraction attack. In other words, its goal is not to recover a secret key but to recover the hardware leakage model, assuming it is a polynomial model. We collect many traces with known values of $X$. For the target nibble, the stochastic attack performs a form of regression to recover the coefficients $a_i$ for the values of $X$ (which are known) and the measured traces. We refer to [17] for full details on the computational part of the attack. If the Hamming weight model is correct, then we would expect $a_0 \neq 0, a_1 \approx a_2 \approx a_4 \approx a_8 \neq 0$ and $a_i \approx 0$ for the other values of $i$. Our stochastic analysis was performed with $10,000$ traces per experiment. It shows two peculiar observations:

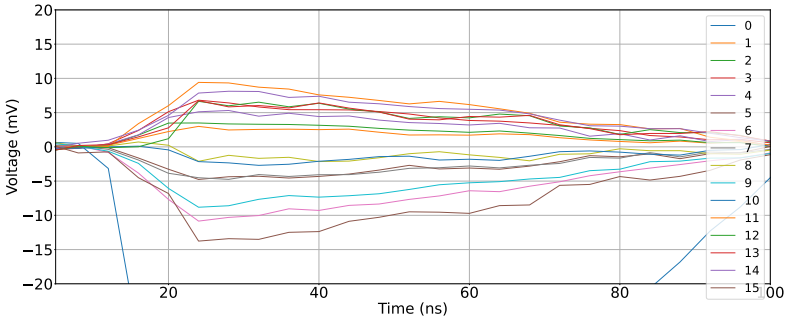



**Fig. 4.** Example of the model extracted by the stochastic attack

1. The model returned by the stochastic analysis is non-linear. For instance, when we use the key

$$0x398c46c68f4664cd6afc2ccca4b2eb9f$$

and vary the target plaintext nibble randomly, we get a uniform distribution for $X$. The model we get is shown in Figure 4. The blue cropped curve corresponds to $a_0$, which is the constant part of the polynomial. We expect 4 more curves to have significant non-zero values, but we can see that this is not the case.

2. By changing the key values during the training phase, we get different leakage models. This is demonstrated in Figures 5 and 6, by showing the linear part of the model for different keys used in the training, with both random and chosen examples. Note that while in these experiments, we use one key for each iteration, the plaintext varies uniformly at random for each trace. Thus, the distribution of $X$ is the same for all experiments.

Moreover, changing the key for each trace does not help achieve a reliable model either. In Figure 7, we give three identical experiments targeting the same nibble and varying the full key randomly for each trace, each with $10,000$ traces. We observe that the three experiments give different weights for different bits. The observations from the TVLA test, CPA attack and stochastic attack lead us to conclude that while the traces from our setup include enough information for observable non-specific leakage, they do not include enough information to be exploited by conventional attacks, both profiled and non-profiled. This begs the question: is the implementation secure against our attack setup? We answer this question negatively in the next section, using DLSCA.

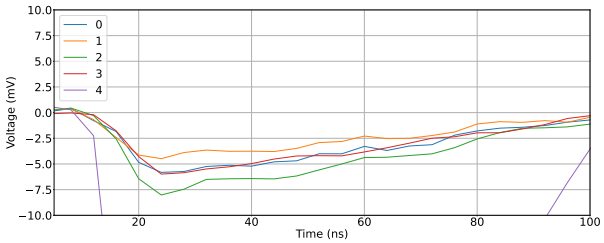
## 6 Deep Learning-based Side-Channel Attack

In recent years, DLSCA has garnered significant interest due to its impressive performance despite the presence of hiding and masking countermeasures. Therefore, in this section, we explore the capability of DLSCA on unrolled, single-cycle block ciphers. We will first recall the profiling attack.

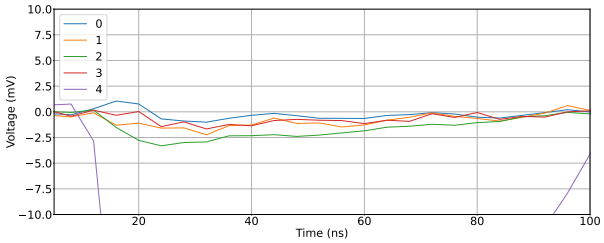
### 6.1 Profiling Attack

Suppose $Z$ and $\boldsymbol{T}$ to be random variables of the sensitive variable and traces, respectively. There are two phases in a profiling attack: the profiling phase and the attack phase. In the profiling phase, a clone device is used to build a distinguisher $\mathcal{F}$. The adversary either has the knowledge of the key or can manipulate the key of the clone device. Profiling traces of a known set of random public variables (plaintexts or ciphertexts) are then collected from the clone device to build $\mathcal{F}$. Next, in the attack phase, several attack traces (using another set of known public variables) are collected using the target device. Then, the attack traces are given to $\mathcal{F}$ to output a probability score for each hypothetical sensitive value *i.e.,* $\boldsymbol{y}_i = \mathcal{F}(\boldsymbol{t}_i)$ for each attack traces $\boldsymbol{t}_i$ acquired from the target device. For key recovery, the log-likelihood score for each key $k \in \mathcal{K}$ is calculated:

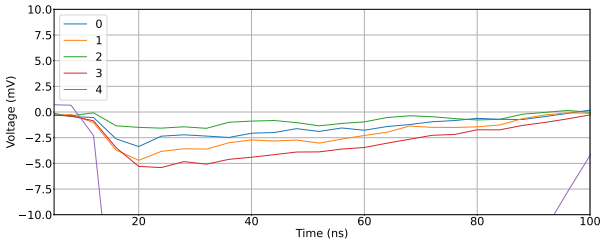
$$score(k) = \sum_{i=1}^{N_a} \log(\boldsymbol{y}_i[z_{i,k}])$$



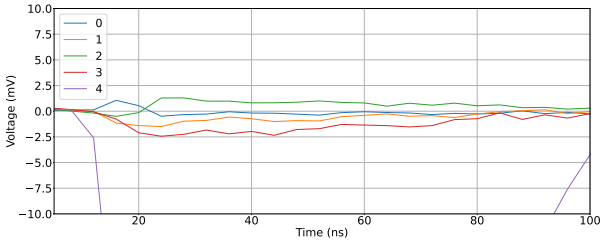

(a) $0x0000000000000000000f000000000000$




(b) $0x0000000000000000000f0f0000000000$

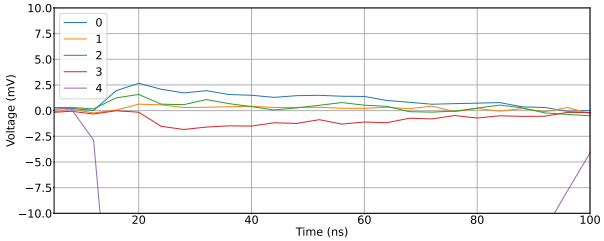



(c) $0x0000000000000000000fff0f00000000$

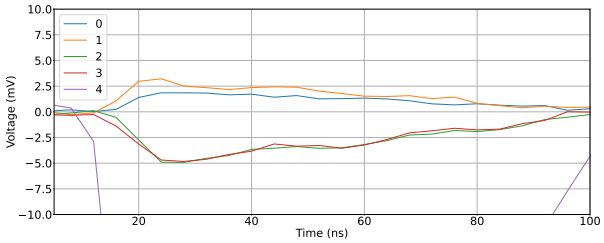



(d) $0x0000000000000000000fff0f000000ff$

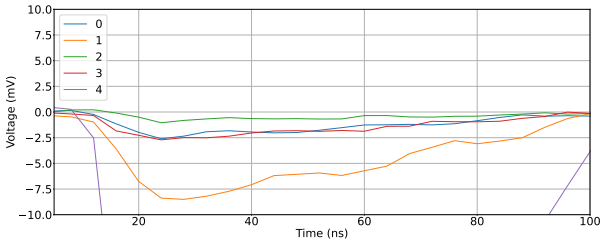
**Fig. 5.** Different linear leakage models obtained for different chosen training keys.



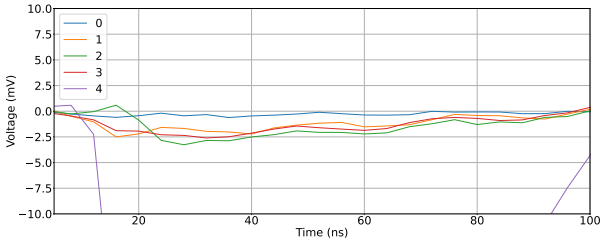

(a) $0xd126cb4b31a9956863561bdcc0832d62$




(b) $0x8e2541eac0e9d3e0ed868f23e166783c$

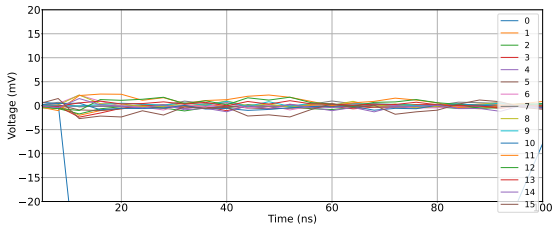



(c) $0xc272b2ae66d3a152e34e9bd3392f8f26$

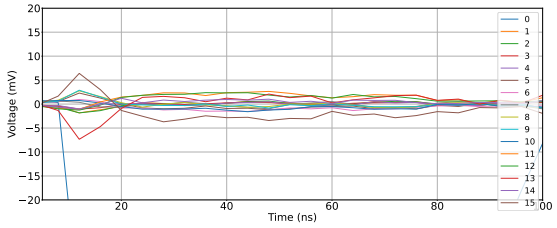



(d) $0x0dbc6fda00c47de857c71be3629c4e12$

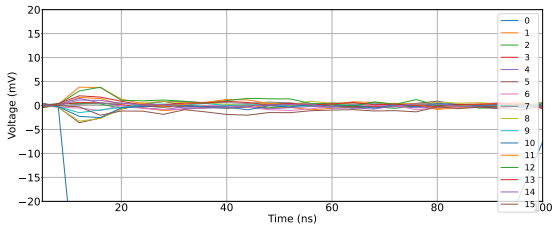
**Fig. 6.** Different linear leakage models obtained for different random training keys.




(a)




(b)




(c)

**Fig. 7.** Different linear leakage models obtained from three experiments where the key varies for each trace.

with $N_a$ as the number of attack traces used and $z_{i,k} = Crypt(pt_i, k)$ are the hypothetical sensitive values based on the key $k$ with $pt_i$ being the corresponding public variable to the trace $\boldsymbol{t}_i$. In this work, we set $Crypt$ to be the sbox of Princev2. The scores are then sorted in a guess vector, $\boldsymbol{G} = [G_0, G_1, \ldots, G_{|\mathcal{K}|-1}]$ where $G_0$ is the most likely key candidate while $G_{|\mathcal{K}|-1}$ is the least likely key candidate. The rank of the key is defined as the index of $\boldsymbol{G}$. We define $GE$ as the average rank correct secret key. Therefore, we recover the key successfully when $GE = 0$. Furthermore, we denote $NTGE$ as the least number of attack traces required to attain $GE = 0$.

For the classical template attack, the conditional probability $Pr(\boldsymbol{T}|Z = z)$ is assumed to be a multivariate Gaussian distribution and uses Bayes' Theorem to build the distinguisher. On the other hand, for a typical profiled DLSCA, a

**Table 1.** Hyperparameter search space.

| Hyperparameter | Options |
|---|---|
| **MLP** | |
| Number of Dense Layers | 1 to 8 (step 1) |
| Neurons per layer | $10, 20, 50, 100, 200$ |
| **CNN** | |
| Convolution layers | 1 to 4 (step 1) |
| Convolution filters | 4 to 16 (step 4) |
| Kernel size | 26 to 52 (step 2) |
| Padding | 0 to 16 (step 2) |
| Pooling type | Average or Max |
| Pooling size | 2 to 10 (step 2) |
| Number of Dense Layers | 1 to 8 (step 1) |
| Neurons per layer | $10, 20, 50, 100, 200$ |
| **Others** | |
| Batch size | 100 to $1,000$ in a step of 100 |
| Activation function | $ReLU, SeLU, ELU$ or $tanh$ |
| Optimizer | Adam or RMSprop |
| Learning Rate | $1e^{-3}, 1e^{-4}, 5e^{-4}, 1e^{-5}, 5e^{-5}$ |
| Weight Initializer | Random Uniform or Glorot Uniform or He Uniform |

DNN is trained as a distinguisher with traces as the input and sensitive values as the labels [22,11].

### 6.2 Experimental Setting

***Hyperparameter Search Space:*** Among the many different DNN architectures, the Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) are the most commonly used architecture within the side-channel domain [22,11,1]. We utilize random search, a popular hyperparameter tuning strategy, sampling 100 DNN configurations from the defined hyperparameter search space stated in Table 1. As with many works, we use the softmax activation function [13] in the last layer for every DNN used and train each DNN with 100 epochs using the categorical-cross entropy loss function.

***Dataset:*** The dataset consists of $50,000$ profiling traces collected under random key settings and $50,000$ attack traces obtained using a fixed key setting. We truncate the traces such that it only includes the encryption portion ranging from sample point 5 to 31.[5] Each of the traces consists of 26 sample points. We use the output sbox of the first round of Princev2 as the label for our attack.[6]

[5] We also manage to recover the secret key with full traces using DLSCA successfully.

[6] We have also applied its Hamming weight as the label. However, we see superior results with just the output of the sbox and will only present these results here.

### 6.3 Experimental Results

We target all 16 nibbles of the first round key and present our results in Table 2. Using DLSCA, we successfully recovered the first round keys for all nibbles using fewer than $20,000$ attack traces. Nibbles 3 and 8 were the easiest to attack, with a NTGE of less than $17,000$. In contrast, nibble 14 proved the most challenging to recover, as the MLP failed to retrieve the key, while the CNN required an NTGE of $19,997$.

Next, we apply DLSCA to all 16 nibbles of the second-round key and display the results in Table 3. Similar to the first-round keys, we successfully recover all nibbles using fewer than $20,000$ attack traces. Notably, we can easily recover nibbles 2 and 7 with less than $18,000$ attack traces. Using both MLP and CNN, we are able to recover the entire second-round key. Overall, we successfully recover the full Princev2 key using DLSCA.

### 6.4 Explanability of Deep Neural Network

In order to understand which sample points the DNNs are using for key recovery, we employ the explainability techniques called KGO proposed by [18]. KGO utilizes a technique called occlusion, which replaces each sample point with a default baseline value. The KGO algorithm iteratively occludes each sample point to obtain the minimum set of sample points that is necessary for a trained DNN to recover the secret key. These minimum set of relevant sample points are called OccPoIs. [18] also proposed an algorithm called 1-KGO to state how much contribution an OccPoI is to the DNN. This is done by occluding each OccPoI one by one and obtaining the GE values for each OccPoI occluded. If the GE is high, this means that this OccPoI has a high contribution to the key recovery, while if the GE is low, it means that it contributes less for the DNN to recover the key. We focus only on the best-performing MLP for nibble 0 as the technique can extend to other well-performing networks. Using KGO, we acquire the sample points 7 and 15 as OccPoIs. Then we apply the 1-KGO algorithm and obtain the GE values 11 and 2 for sample points 7 and 15, respectively. This means that the sample point 7 has the most contribution compared to 15. This aligns with the TVLA result that sample point 7 leaks the most. We plot the OccPoIs and their contribution in Figure 8 (blue plot).

To evaluate OccPoIs' relevance in classical SCA, we conducted a template attack utilizing OccPoIs. Notably, the GE converged to 2, outperforming the $GE = 11$ achieved without feature selection on truncated traces. We also compare its performance with two other feature selection tools, namely SOSD and SOST [12], by selecting the same number of sample points as OccPoIs. SOSD selected the sample points at 6 and 12 while SOST selected the sample points at 2 and 13 (illustrated in Figure 8). We apply template attack with these sample points only and observe that both SOSD an SOST have high GE, suggesting that SOSD and SOST fail to recover the secret key. This demonstrates that OccPoIs, selected via DNNs with KGO, effectively exploit leakages and enhance classical attack effectiveness.

**Table 2.** Performance using DLSCA. Random key setting with truncated traces for first round.

| **Nibble** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **MLP** | 19847 | 19599 | 19971 | 15065 | 19353 | 18850 | 19959 | 18523 |
| **Nibble** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **MLP** | 16771 | 19677 | 19997 | 19345 | 19881 | 17300 | $(GE = 3)$ | 18372 |
| **Nibble** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **CNN** | 19771 | 18048 | 19745 | 16416 | 19118 | 18843 | 19453 | 18025 |
| **Nibble** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **CNN** | 15246 | 19386 | 19866 | 19445 | 18924 | 17811 | 19997 | 18348 |

**Table 3.** Performance using DLSCA. Random key setting with truncated traces for second round.

| **Nibble** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **MLP** | 18778 | 19846 | 17144 | 19901 | 19144 | 17510 | 19996 | 15436 |
| **Nibble** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **MLP** | 19995 | 19985 | 19188 | 19519 | 18381 | $(GE = 2)$ | 19273 | 19829 |
| **Nibble** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **CNN** | 19199 | 19863 | 17036 | 18036 | 17110 | 19047 | 18620 | 15883 |
| **Nibble** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **CNN** | $(GE = 1)$ | 19932 | 17299 | 19503 | $(GE = 1)$ | 19994 | 19972 | 19260 |

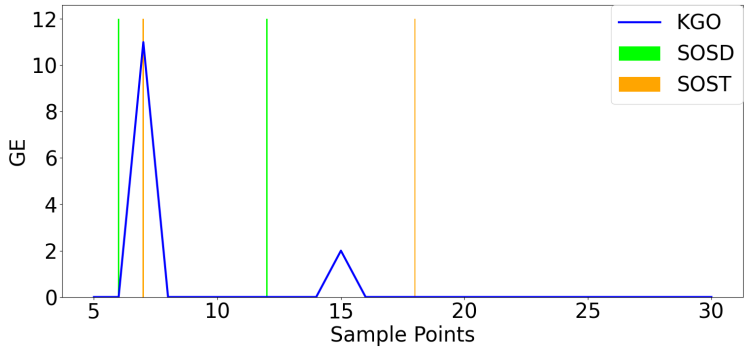



**Fig. 8.** Sample Points Selected by Various Feature Selection Tools.

**Table 4.** Performance using Template Attack with Feature Selection Used.

| Technique Used | Template Attack Performance |
|---|---|
| No Feature Selection | $GE = 11$ |
| KGO | $GE = 2$ |
| SOSD | $GE = 13$ |
| SOST | $GE = 11$ |

## 7 Conclusion

In this paper, we investigated the vulnerability of the hardware implementation of Princev2 on Artix 7 to power analysis using cheaper and portable oscilloscopes. We demonstrated that the setup can still easily observe leakage and can perform key recovery using DLSCA. We also presented the challenges of extending these attacks to classical attacks that are not based on deep learning. Furthermore, we investigated which sample points are useful (i.e., OccPoIs) to the neural network for key recovery through the KGO explainability technique. We improve the classical template attack by using these relevant sample points. As part of future work, we plan to investigate the leakage model and the reasons behind the observed inconsistencies in classical attacks in more detail.

## Acknowledgment

Mustafa Khairallah is supported by the Wallenberg-NTU Presidential Postdoctoral Fellowship and worked on the experiments in the paper while at Lund University Sweden. We would like to thank Shivam Bhasin for suggestions and support during these experiments.

## References

1. Acharya, R.Y., Ganji, F., Forte, D.: Information Theory-based Evolution of Neural Networks for Side-channel Analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 401–437 (2023). https://doi.org/10.46586/TCHES.V2023.I1.401-437, https://doi.org/10.46586/tches.v2023.i1.401-437
2. Anand, R., Banik, S., Caforio, A., Ishikawa, T., Isobe, T., Liu, F., Minematsu, K., Rahman, M., Sakamoto, K.: Gleeok: A family of low-latency prfs and its applications to authenticated encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 545–587 (2024). https://doi.org/10.46586/TCHES.V2024.I2.545-587, https://doi.org/10.46586/tches.v2024.i2.545-587
3. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency PRF. IACR Trans. Symmetric Cryptol. **2021**(1), 37–77 (2021). https://doi.org/10.46586/TOSC.V2021.I1.37-77, https://doi.org/10.46586/tosc.v2021.i1.37-77

4. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: Bipbip: A low-latency tweakable block cipher with small dimensions. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 326–368 (2023). https://doi.org/10.46586/TCHES.V2023.I1.326-368, https://doi.org/10.46586/tches.v2023.i1.326-368
5. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptogr. Eng. **10**(2), 163–188 (2020). https://doi.org/10.1007/S13389-019-00220-8, https://doi.org/10.1007/s13389-019-00220-8
6. Bhasin, S., Guilley, S., Sauvage, L., Danger, J.: Unrolling cryptographic circuits: A simple countermeasure against side-channel attacks. In: Pieprzyk, J. (ed.) Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings. Lecture Notes in Computer Science, vol. 5985, pp. 195–207. Springer (2010). https://doi.org/10.1007/978-3-642-11925-5_14, https://doi.org/10.1007/978-3-642-11925-5_14
7. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_14, https://doi.org/10.1007/978-3-642-34961-4_14
8. Bozilov, D., Eichlseder, M., Knezevic, M., Lambin, B., Leander, G., Moos, T., Nikov, V., Rasoolzadeh, S., Todo, Y., Wiemer, F.: Princev2 - more security for (almost) no overhead. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12804, pp. 483–511. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_19, https://doi.org/10.1007/978-3-030-81652-0_19
9. Cagli, E., Dumas, C., Prouff, E.: Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Preprocessing. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 45–68. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_3, https://doi.org/10.1007/978-3-319-66787-4_3
10. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A low-latency block cipher for secure cache-randomization. In: Calandrino, J.A., Troncoso, C. (eds.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023. pp. 1937–1954. USENIX Association (2023), https://www.usenix.org/conference/usenixsecurity23/presentation/canale
11. Eng, T.Y.H., Bhasin, S., Weissbart, L.: Train Wisely: Multifidelity Bayesian Optimization Hyperparameter Tuning in Side-Channel Analysis. IACR Cryptol. ePrint Arch. p. 170 (2024), https://eprint.iacr.org/2024/170

12. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 15–29. Springer (2006). https://doi.org/10.1007/11894063_2, https://doi.org/10.1007/11894063_2
13. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive computation and machine learning, MIT Press (2016), http://www.deeplearningbook.org/
14. Leander, G., Moos, T., Moradi, A., Rasoolzadeh, S.: The SPEEDY family of block ciphers engineering an ultra low-latency cipher from gate level for secure processor architectures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 510–545 (2021). https://doi.org/10.46586/TCHES.V2021.I4.510-545, https://doi.org/10.46586/tches.v2021.i4.510-545
15. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking Cryptographic Implementations Using Deep Learning Techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10076, pp. 3–26. Springer (2016). https://doi.org/10.1007/978-3-319-49445-6_1, https://doi.org/10.1007/978-3-319-49445-6_1
16. Moos, T.: Unrolled cryptography on silicon A physical security analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(4), 416–442 (2020). https://doi.org/10.13154/TCHES.V2020.I4.416-442, https://doi.org/10.13154/tches.v2020.i4.416-442
17. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer (2005). https://doi.org/10.1007/11545262_3, https://doi.org/10.1007/11545262_3
18. Yap, T., Bhasin, S., Picek, S.: OccPoIs: Points of Interest based on Neural Network's Key Recovery in Side-Channel Analysis through Occlusion. IACR Cryptol. ePrint Arch. p. 1055 (2023), https://eprint.iacr.org/2023/1055
19. Yli-Mäyry, V., Homma, N., Aoki, T.: Improved Power Analysis on Unrolled Architecture and Its Application to PRINCE Block Cipher. In: Güneysu, T., Leander, G., Moradi, A. (eds.) Lightweight Cryptography for Security and Privacy - 4th International Workshop, LightSec 2015, Bochum, Germany, September 10-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9542, pp. 148–163. Springer (2015). https://doi.org/10.1007/978-3-319-29078-2_9, https://doi.org/10.1007/978-3-319-29078-2_9
20. Yli-Mäyry, V., Homma, N., Aoki, T.: Chosen-Input Side-Channel Analysis on Unrolled Light-Weight Cryptographic Hardware. In: 18th International Symposium on Quality Electronic Design, ISQED 2017, Santa Clara, CA, USA, March 14-15, 2017. pp. 301–306. IEEE (2017). https://doi.org/10.1109/ISQED.2017.7918332, https://doi.org/10.1109/ISQED.2017.7918332
21. Yli-Mäyry, V., Ueno, R., Miura, N., Nagata, M., Bhasin, S., Mathieu, Y., Graba, T., Danger, J., Homma, N.: Diffusional side-channel leakage from unrolled lightweight block ciphers: A case study of power analysis on PRINCE. IEEE Trans. Inf. Forensics Secur. **16**, 1351–1364

(2021). https://doi.org/10.1109/TIFS.2020.3033441, https://doi.org/10.1109/TIFS.2020.3033441

22. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 1–36 (2020). https://doi.org/10.13154/TCHES.V2020.I1.1-36, https://doi.org/10.13154/tches.v2020.i1.1-36