Differential Power Analysis of the Picnic Signature Scheme

Tim Gellersen, Okan Seker, and Thomas Eisenbarth

University of Lübeck
tim.gellersen@student.uni-luebeck.de
okan.seker@uni-luebeck.de
thomas.eisenbarth@uni-luebeck.de

Abstract. Post-quantum cryptography introduces cryptographic algorithms that are secure against adversaries who can employ a quantum computer and it is the inevitable next-step in the evolution of the cryptographic algorithms. In order to create a conventional foundation the National Institute of Standards and Technology (NIST) started a competition for Post-Quantum Cryptography in 2017.
This paper introduces the first differential side channel analysis of a candidate in the competition; the Picnic Signature Scheme. We present a successful side channel analysis of the underlying Multiparty LowMc implementation and show how leakages can be exploited to recover the entire secret key using two different parts of the algorithm. LowMc key recovery then allows to forge signatures for the calling Picnic post-quantum signature scheme. We target the NIST reference implementation executed on a FRDM-K66F development board. Key recovery succeeds with less than 1000 traces, which can be obtained from less than 30 observed Picnic signatures.

Keywords: Picnic Signature Scheme · LowMc · Multiparty Computation · Power Analysis · DPA.

1 Introduction

The recent trends in security show that public key cryptography is an indispensable component of secure communication. Because of our constant evolution in computer science, the existence of usable quantum computers might not be too far away from reality. Quantum computers which can perform much more powerful computations could break most of the widely used public key cryptographic schemes, due to Shor's algorithm [26]. Despite the possibility that there may never be quantum computers that are able to process enough bits to break the current public key schemes, we should still be prepared for the development of these types of quantum computers and expand the portfolio of deployable public key schemes. *NIST* started a competition for Post-Quantum Cryptography in 2017 and is currently in the second round of the submissions.

However, the security of cryptographic systems does not only require resistance to purely computational cryptanalysis. For practical implementations, resistance to other attacks such as physical side channel attacks is also relevant. A wide range of such attacks has been shown to reveal secret keys with little effort, especially if no countermeasures were taken during implementation. Two main classes of physical attacks are *fault injection attacks* and passive *side channel attacks*. Fault injection attacks manipulate the (secret) state of a cryptographic scheme, while it is performing an encryption or signing operation. Afterwards, the attacker tries to extract information about the scheme's secret key, from the faulted output of the scheme [7,6]. Similarly, the information derived from power [20], sound [15], or electromagnetic emanation [13] of a target that runs an unprotected cryptographic implementation can reveal information regarding the secret state. The impact of side channel analysis is shown in the literature ranges from high-speed CPU's [14] to virtual machines in cloud systems [17]. Furthermore system-on-chip embedded platforms [21] and even white-box implementations [8] are vulnerable to these kind of attacks.

Differential power analysis (DPA) is one of the most popular side channel attacks and was originally proposed by Kocher et al. [20]. The main idea is to measure the power consumption of a device, while it is performing a cryptographic operation and correlate it with key related information that is already known to the attacker. The impact of this attack is shown in the literature in works such as [23] and [25]

Prior work has already shown, that post-quantum cryptographic schemes on embedded systems are vulnerable to side channel attacks [27]. In 2018, Park et al. [24] presented that correlation power analysis can be applied to signature schemes based on multivariate quadratic equations namely; UOV [19] and Rainbow [11]. Side channel properties of hash based signatures have also been analyzed [12,9]. Recent work by Aranha et al. [5] analyses the security of Fiat-Shamir based signature schemes against fault attacks.

Our Contribution: This work analyses the side channel vulnerability of the *Picnic* scheme [10], with a focus on DPA attacks. Picnic is a candidate in the second round of the ongoing NIST PQC competition. We present the first successful side channel analysis of the Picnic signature scheme, by analysing its core component; MPC-LowMc. We explore the security features of MPC-LowMc and its direct effects on the security of the whole signature scheme. We used the reference implementation of MPC-LowMc on a FRDM-K66F development board and managed to collected electromagnetic emanation from it. Based on this, we recover the secret key from MPC-LowMc using two different attacks: an attack on the secret sharing process and an attack on the multi party computation of the Sbox layer. The first attack, is able to recover Picnic's secret key from less then 5000 power traces, which corresponds to around 30 observed Picnic signatures. The second attack can be used to leak 30 bits of secret key related information from each round of the MPC-LowMc cipher. However, this step can be repeated, to form a system of linear equations, whose solution is the 128 bit secret key of the Picnic scheme. Whether the information obtained in the first

step leads to linear independent equations or not depends on the randomised matrices that are used by MPC-LowMC. In practice, we were able to solve the system of linear equations by combining the information obtained from the first five rounds of the MPC-LowMc cipher. For longer key sizes, like 196 or 256 bits, we can just use more rounds, to receive 220 or 280 linear equations.

Outline of the Paper: In Section 2, we introduce the basic notations and definitions. In Section 3 we explain our target implementation with the practical setup and the implication of our attack on MPC-LowMc to the Picnic scheme. The attack on the secret sharing process and the attack on the Sbox layer are explained in Section 4 and Section 5 respectively. Moreover each attack section is supported by experimental results.

2 Preliminaries

In this section, we provide the definitions used in this paper. We start with the differential power analysis. DPA is first introduced by Kocher et al. [20] and analyses side channel leakage over many different traces, selecting the most likely secret values using a statistical test.

In order to implement DPA, an adversary first selects an intermediate variable which is a function (f) of a secret value and a known value (such as plaintext or ciphertext) . In the next step, the adversary chooses a function ϕ, that models how a register value influences the power consumption. Common models are the hamming weight or a single bit [22] of the register. We refer to this function as the *key hypothesis*. In the second step the adversary collects a large number of side channel traces to implement a statistical analysis.

In our analysis we use a simple correlation as our statistical tool. For each key candidate $k^* \in \mathcal{K}$ [1] we form the following set:

$$(\phi(f(p_1, k^*)), \ldots, \phi(f(p_N, k^*))),$$

where N is the number of traces and p_i for $1 \leq i \leq N$ is the known input. This set is then applied to the statistical tool together with the side channel measurements. The statistical analysis results in an observable peak for the correct key hypothesis, while wrong keys reside within a threshold.

In order to present our target implementation, we first need to introduce the underlying scheme. LowMc [4] is a flexible block cipher with low AND depth. Thus it is a suitable cipher for Secure Multi-Party Computation (MPC), Zero-Knowledge Proofs (ZK) and Fully Homomorphic Encryption (FHE). An overview of the structure of the scheme with the parameters (n, k, r) where n, k and r denotes block length, key size and number of rounds respectively, can be found in Algorithm 1. The details of the i^{th} round are as follows:

[1] $|\mathcal{K}|$ is small enough to process the analysis and generally it ranges from 2^4 to 2^8.

Algorithm 1 LowMc Encryption Scheme

Input: Key matrices $KM_{i\in[1,r]} \in \mathbb{F}_2^{n\times k}$, Linear matrices $LM_{i\in[1,r]} \in \mathbb{F}_2^{n\times n}$, Round constants $RC_{i\in[1,r]} \in \mathbb{F}_2^n$, a plaintext $p \in \mathbb{F}_2^n$ and a secret-key $k_s \in \mathbb{F}_2^n$
Output: Ciphertext $s \in \mathbb{F}_2^n$ such that $s = \text{LowMc}(p, k_s)$.

$s \leftarrow (KM_0 \cdot k_s) \oplus p$ ▷ Initial Key Addition
for $1 \leq i \leq r$ **do**
  $s \leftarrow Sbox(s)$ ▷ `SboxLayer`
  $s \leftarrow LM_i \cdot s$ ▷ `LinearLayer`
  $s \leftarrow RC_i \oplus s$ ▷ `ConstantAddition`
  $s \leftarrow (KM_i \cdot k_s) \oplus s$ ▷ `KeyAddition`
return s

1. `SboxLayer`: The round function consists of multiple parallel application of the same 3×3 Sbox as shown in Equation (1). However only a part of the state is processed by the Sbox layer and the remaining bits stay unchanged.

$$Sbox(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab). \tag{1}$$

2. `LinearLayer`: The state is multiplied with a random and invertible matrix LM_i, where $LM_i \in \mathbb{F}_2^{n\times n}$.
3. `ConstantAddition`: A constant vector RC_i is added to the state, where $RC_i \in \mathbb{F}_2^n$.
4. `KeyAddition`: The round key k_i is added to the state, where k_i is obtained by the multiplication of the key matrix KM_i where $KM_i \in \mathbb{F}_2^{n\times k}$ and k_s is the master-secret key .

ZKBOO decomposition and Picnic Signature Scheme: Zero knowledge for boolean circuits (ZKBOO) is a protocol introduced by Giacomelli et al. [16], that can prove the knowledge of a preimage of a one-way function f, such that $f(x) = y$, without leaking any information on x. The main idea of ZKBOO is to use MPC-in-the-head paradigm introduced in [18].

In the ZKBOO scheme, the verifier and the prover both know the output y. However, the prover is the only one, that knows x, such that $y = f(x)$. The prover then calculates $f(x)$ using a z-privacy circuit decomposition of f, i.e. at least z shares are required to reconstruct x. The scheme can be summarized as follows: The prover runs the MPC-Circuit multiple times and commits to each of the generated views, before sending the committed values to the verifier. The verifier then requests the prover to open the commitment of some (less than z) shares from each execution and checks the revealed values for consistency. As less than z shares of each run are opened, the verfier does not learn anything about x.

The *Picnic* scheme introduced by Chase et al. [10], uses ZKB++ (which is an improved version of ZKBOO) to create signatures. They chose the LowMc cipher as the one-way function f, because it only requires amount of AND gates, greatly reducing the complexity of the circuit decomposition.

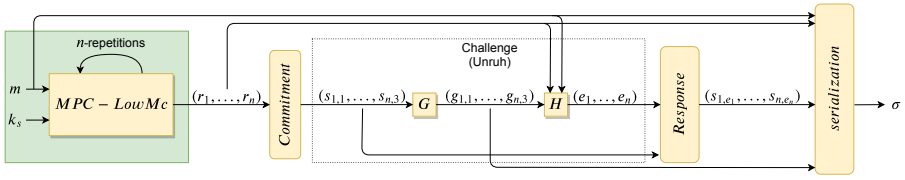

Fig. 1. Overview of the *Picnic* signature scheme where m is constant plaintext that is used in LowMc such that $\text{LowMc}(m, k_s) = y$, $sk = ((y, m), k_s)$ is the secret key and $pk = (y, m)$ is the public key. The figure is adapted from [5].

An overview of the Picnic signature scheme can be seen in Figure 1. As seen in the figure, the MPC-LowMc circuit is the foundation of the scheme. Next, we describe our attacks on the MPC-LowMc part of the scheme. The aim of the attacks is to recover the secret key k_s which is also the secret key used in Picnic.

3 DPA on LowMc and The Extensions to Picnic

In Figure 1 we can see the signing process of *Picnic*. Our goal is to apply a DPA attack on MPC-LowMC, to gain information about intermediate state values which then will be used to recover the secret key. As our target, we focus on the reference implementation given by the authors [3] that uses the Unruh transformation with security parameters `L1`. However, our attacks are independent of the actual transformation (Fiat-Shamir or Unruh Transformation), and can be adapted to the different security parameters.

Attacker Model: We model an adversary who has physical access to a target device, running the *Picnic* signature scheme. In particular, the attacker is able to measure the side channel information, such as power electromagnetic emanation, during the signing of any message and can obtain the valid signature in a *known-plaintext scenario*. Remark, that the signed message does not affect our model, since the message does not change the computation performed by LowMc, which is based on a fixed (and public) input m and a secret key k_s.

3.1 Experimental Setup

All experiments are performed on an embedded test platform. We ported the relevant parts of the Picnic reference implementation from the NIST round 2 submission of Picnic [3] to the FRDM-K66F development board.

Modifications to the Code: As the *Picnic* reference implementation needs a lot of RAM to save all the MPC-Rounds, we modified the code to be able to let the algorithm run on the device. Since the relevant part of the signature generation are the calls to LowMC, we only execute the shared *MPC-LowMc* computations on the board. As customary, we also placed a trigger before the start of the LowMc calls to make the attack easier in our setting and to save us some effort.

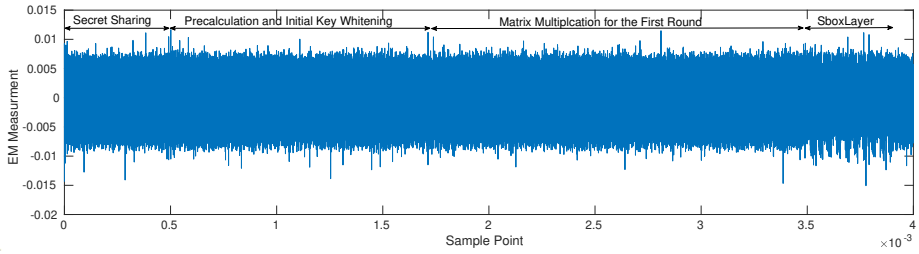

Fig. 2. An annotated example EM trace for one round of LowMc.

Measurement Setup: For our test purposes, we choose the FRDM-K66F development board, that features an NXP MK66FN2M0VMD18 Cortex-M4F MCU, which can be clocked at up to 180 MHz, with 2MB flash memory and 256 KB SRAM. Measurements were taken using a Tektronix MSO6 at 312.5 MHz sampling rate. We collected a total of 20000 traces using a Langer EM Probe [2] placed at the C37 0.1μF blocking capacitor of the FRDM-K66F development board [1]. The MCU was clocked at 120MHz.

An example trace starting with the secret sharing until the end of the first `SboxLayer` can be seen in Figure 2, which contains the relevant initial operations as well as operations from the first round of MPC-LowMC. The 10 shared Sbox computations are on the right hand side, while the secret sharing happens early, as indicated on the left hand side. We applied a minimal pre-processing step, of cutting the necessary part for each attack. Furthermore, we compressed 10 data points to 1, by applying the mean over these 10 points, effectively reducing the sampling rate and noise.

4 Attack on the Secret Sharing Process

In this section, we introduce our first attack which targets the secret sharing process of the MPC-LowMc reference implementation. As seen in Figure 2, the secret sharing process is located before the actual MPC-LowMc part starts. We start with a theoretical explanation of the attack which is followed by experimental results.

To set up the model we focus on the description of the secret sharing of the secret key k_s in the scheme. First, two n bit keys for two players k_0 and k_1 are generated randomly. Using the generated keys (or key shares), the key share for the last player k_2 is calculated as follows:

$$k_2 = k_s \oplus k_0 \oplus k_1.$$

During the challenge/opening phase of the *Picnic* scheme, the key shares of two players are revealed. In the following analysis, we focus on Challenge 0 (C_0), which reveals k_1 and k_2. The power trace of a simple xoration does *usually* not reveal enough information to apply a side channel attack. However, in the case

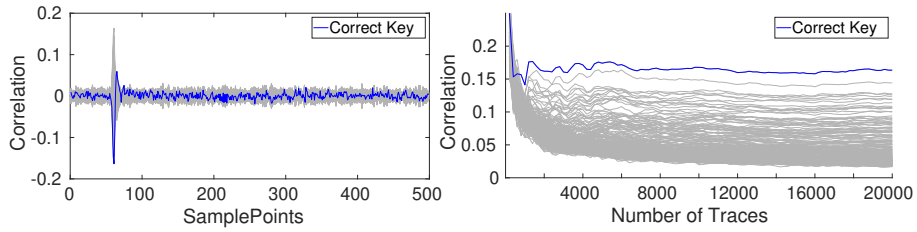

Fig. 3. Left side: DPA using 5000 sample points and 20.000 traces. Right side: The change of absolute values of DPA results with respect to number of traces. Although we can see a positive peak in the first figure, the second figure shows that the correct key is distinguishable with less that 2000 traces.

of Picnic, k_2 is the result of two chained xor operations. We can imagine that the output of $k_s \oplus k_0$ is saved in a register R and then continuously used to be xored with k_1. Thus, we build our hypothesis for the point where we xor the second key with the intermediate result $k_s \oplus k_0$. For each key candidate $k^* \in \mathcal{K}$ we produce the hypothesis value as follows:

$$H_{k^*} = \mathrm{HW}(k^* \oplus k_0 \oplus k_1), \tag{2}$$

where HW is the hamming weight function. That is the function ϕ (defined in Section 2) is selected as the hamming weight function. Using the hypothesis values H_{k^*} for all $k^* \in \mathcal{K}$ (where $\mathcal{K}$ denotes the key space), we can implement a statistical analysis and the best key candidate reveals k_s.

4.1 Experimental Results

The resulting correlation for the key hypothesis with 5.000 sample points and 20.000 traces can be seen on the left hand side of Figure 3 which shows the correlation values for the first byte of the secret key. Furthermore, on the right hand side of Figure 3, we present the change in the correlation results for each key guess with respect to the number of traces. We can see 9 clusters for the different key guesses based on the *hamming weight* values (which can take values between zero and eight). As seen in the figure we were able to recover the secret key with 2000 traces.

5 Attack on the Substitution Layer

In order to explain the attack, let us first denote the *MPC-LowMc* players as $\mathcal{P}_0$, $\mathcal{P}_1$, $\mathcal{P}_2$ and their states as p_0, p_1 and p_2 respectively. Also we denote a (theoretical) unshared version (player $\mathcal{P}_r$) of the state as p_r, i.e. $p_r = p_0 \oplus p_1 \oplus p_2$. Furthermore, the tapes which contain the random values that are produced by player $\mathcal{P}_i$ is denoted by t_i and the challenges are denoted by C_i.

We focus on the `SboxLayer` of the LowMc cipher in the MPC settings and compare them with the unmasked state values. That is, we analyze the state values of LowMc(m, k_s) with MPC-LowMc(m, k_s), where m is the constant (and predefined) message and k_s is the secret key. MPC-LowMc starts by initialising the state of $\mathcal{P}_i$ with k_i. As seen in Algorithm 1, the plaintext (or the message in our case) is xored with the key value. In the MPC setting, this step is only performed for the first player's state. In the unmasked LowMc version we set the $\mathcal{P}_r$ state to the value $KM_0 \cdot k_s \oplus m$. In summary, the initial states of the MPC-LowMc (p_0, p_1, p_2) variant and the (imaginary) LowMc variant (p_r) are as follows:

- $p_0 \leftarrow KM_0 \cdot k_0 \oplus m$,
- $p_1 \leftarrow KM_0 \cdot k_1$,
- $p_2 \leftarrow KM_0 \cdot k_2$,
- $p_r \leftarrow KM_0 \cdot k_s \oplus m$.

Keep in mind, that the state of $\mathcal{P}_r$ always equals to the xoration of all other players' corresponding states, i.e. p_r is equal to $p_0 \oplus p_1 \oplus p_2$ at any given time. We build our hypothesis on this fact. The initial state operations are followed by the `SboxLayer`. We remark Equation (1) where we introduce the Sbox operation;

$$S(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab).$$

In the *MPC-LowMc* setting, the Sbox is defined according to the linear decomposition of the binary multiplication gate. That means that the players need to communicate their bits according to the definition by ZKBOO [10], to securely calculate the values ab, bc and ac. Assume, that the values ab, bc and ac are calculated in the MPC setting and the state values for $\mathcal{P}_i$ are denote by a_i, b_i, c_i, $[ab]_i$, $[bc]_i$ and $[ac]_i$ respectively (for example $ab = [ab]_0 \oplus [ab]_1 \oplus [ab]_2$). Each player calculates the following equations in order to generate the shared representations of the state values:

$$\begin{aligned} [bc]_i &= b_i c_i \oplus b_j c_i \oplus b_i c_j \oplus r_i^{bc} \oplus r_j^{bc} \\ [ac]_i &= a_i c_i \oplus a_j c_i \oplus a_i c_j \oplus r_i^{ac} \oplus r_j^{ac} \\ [ab]_i &= a_i b_i \oplus a_j b_i \oplus a_i b_j \oplus r_i^{ab} \oplus r_j^{ab} \end{aligned} \tag{3}$$

where $j = i + 1 \bmod (3)$ and r_i^{ab} (resp. r_i^{ac} and r_i^{bc}) represents the random bit that is generated by the i^{th} player while calculating the output share of $[bc]_i$ (resp. $[ac]_i$ and $[bc]_i$). After the communication step, each player calculates its share of the state (i.e. output of the Sbox operation). The output state of $\mathcal{P}_i$ can be calculated as follows:

$$Sbox(a_i, b_i, c_i) = (a_i \oplus [bc]_i, a_i \oplus b_i \oplus [ac]_i, a_i \oplus b_i \oplus c_i \oplus [ab]_i) \tag{4}$$

In *Picnic*, n runs of the *MPC-LowMc* are pre-computed. Next, in the challenge response phase, two players' keys and random tapes are opened. For example, the values a_i, b_i, and r_i for $i = 1, 2$ are opened during the challenge C_0.

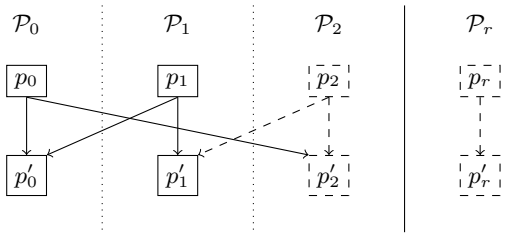

Fig. 4. `SboxLayer` calculation for *MPC-LowMc* (left hand side) and LowMc (right hand side). Dashed boxes and arrows represent the values that are *not* opened during the challenge C_0 and p_i (resp. p'_i) represents the state of $\mathcal{P}_i$ before (resp. after) the Sbox operation

A brief summary of challenge C_0 can be seen in Figure 4. In the representation we use the notation p_i (resp. p'_i) to represent the state of $\mathcal{P}_i$ before (resp. after) the Sbox operation.

Setup and Hypothesis: Similar to the previous attack, we focus on the challenge C_0. However, the attack can be adapted to other challenges C_1 and C_2. We know that in any instance of the cipher, the equation $p_r = p_0 \oplus p_1 \oplus p_2$ holds. Moreover, LowMC uses the same plaintext every run, which means that the state p_r, is constant before each Sbox operation in every run. Thus we can build our hypothesis on the state of $\mathcal{P}_r$ which can be seen in Figure 4.

We define a state-guess k^* of $\mathcal{P}_r$ of the Sbox and call the corresponding state p_{k^*}. The aim of the DPA attack is to get knowledge of the state of $\mathcal{P}_2$. As mentioned before, we cannot directly recover it with a first order DPA attack (due to the circuit decomposition), as we would need to guess the actual state as well as the random value/mask. However, we can exploit the structure of MPC-LowMC, to recover the random mask based on a key guess and the values opened in challenge C_0. This way, we can again use a first order DPA attack, to recover key related data. Here is the summary of the attack steps:

1. For each $k^* \in \mathcal{K}$ (in the LowMc case $\mathcal{K} := \mathbb{F}_2^3$) compute the state of player $\mathcal{P}_2$ as: $\tilde{p}_2 = p_0 \oplus p_1 \oplus p_{k^*}$.
2. Compare the opened output-share p'_1 with the $Sbox(p_1, \tilde{p}_2, r_1, \sim)$, where $\sim$ indicates the calculations are processed as in Equation (4) without using the randomness produced by $\mathcal{P}_2$ to have a guess value on r_2.

$$\tilde{r}_2 = p'_1 \oplus Sbox(p_1, \tilde{p}_2, r_1, \sim)$$

   where $r_i = (r_i^{ab}, r_i^{bc}, r_i^{ac})$ denotes the set of random bits generated by $\mathcal{P}_i$. The 3-bit state values p_1 (and corresponding random values r_1) are opened during the challenge C_0.
3. Calculate the Sbox output for guessed state of $\mathcal{P}_2$,

$$p'_{2H} = Sbox(p_1, \tilde{p}_2, r_1, \tilde{r}_2).$$

Algorithm 2 DPA Attack on the State of LowMc.

Input: A set of state values p_1, p_2, p'_1, p'_2 and random values r_0 and r_1. A side channel trace T.

Output: Best key candidate k^*.

1: **for all** $k^* \in \mathcal{K}$ **do**
2: $\tilde{p_2} = p_1 \oplus p_2 \oplus p_{k^*}$ ▷ state guess of $\mathcal{P}_2$
3: $\tilde{r_2} = p'_1 \oplus Sbox(p_1, \tilde{p_2}, r_1, \sim)$ ▷ calculate the mask depending on the guess
4: $p'_{2H} = Sbox(\tilde{p_2}, p_0, r_0, \tilde{r_2})$ ▷ output state based on key guess k_*
5: $R \leftarrow \text{corr}(\phi(p'_{2H}), T)$ ▷ proceed with the statistical analysis.
6: **return** best key candidate k^*

4. DPA: proceed a statistical analysis using $\phi(p'_{2H})$ with the power consumption of the device.

For the correct guess k^*, the output for the Step-2 will result in the mask which is used from $\mathcal{P}_2$ during the communication of the bits for the state of p'_1. With the knowledge of the mask for $\mathcal{P}_2$, we can build a hypothesis H on the output bits of $\mathcal{P}_2$ as in Step-3. Our hypothesis H will therefore be $p'_{2H} = Sbox(p_1, \tilde{p_2}, r_1, \tilde{r_2})$ which simulates the hypothetical intermediate output of the Sbox for $\mathcal{P}_2$. We can then correlate it with the power consumption of the device to apply to our DPA attack. A more compact description of the attack can be found in Algorithm 2. The state values p_1 and p_2 (resp. r_1 and r_2) represents the set of state values with N elements where N is the number of measurements such as p_i denotes the set of state values for $\mathcal{P}_i$ for N measurement i.e. $p_i = \{p_i^1, \ldots, p_i^N\}$.

5.1 Experimental Results

In order to experimentally verify our attack we focused on the first round of MPC-LowMc. Due to the `SboxLayer` structure of LowMc, only the first 30-bit state goes through Sbox operations therefore we can only recover 30 bit key-information of the first round key.

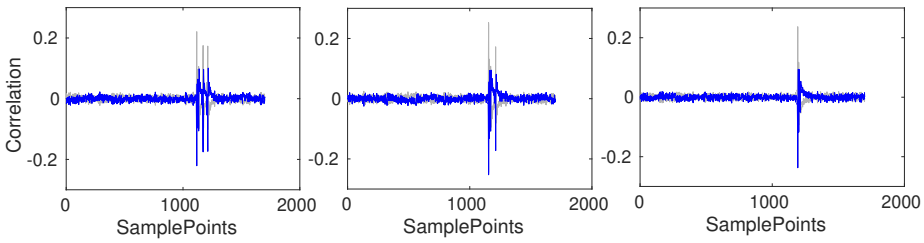

Fig. 5. DPA with 20.000 traces on the 10th Sbox using first, second and third bit respectively

In Figures 5, we present the DPA of the 3 output bits of the 10th Sbox for 20000 traces. We see that the two lines corresponds to the key guesses are the inverse of each other. As a result of our measurement setup, the negatives peaks correspond to the device's current and therefore give us the correct bits.

By looking at the correlation figures for bit 1 to 3, we can see a strong characteristics peaks. By overlaying the 3 figures we see that the peaks occurs at almost the same points in time. Remark that Sbox structure of LowMc has some characteristics(as seen in Equation (1)), such as first input bit a is xored with the first, the second and the third output bit. Therefore, when we build our hypothesis on the first bit of the Sbox output (i.e $a \oplus bc$), the guessed value is highly correlated with the second and the third output bit. Thus, we see three peaks in the first figure. Similarly in the second one (where we guess the $a \oplus b \oplus ac$) we see two peaks and in the last one we can see only one peak.

On the basis of the observed structure, we can see that our model fits well for the MPC structure. In Figure 6, we can see the distinguishability of the three bits of the 10th Sbox. We can see that no more than 1000 traces are needed to clearly identify the correct key.

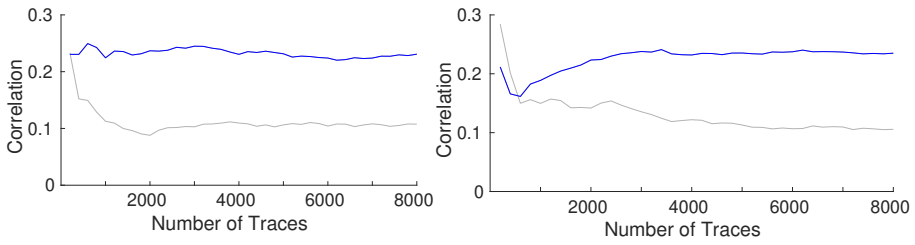

Fig. 6. Attack on 10th Sbox using first and third bit respectively. Correct key is distinguishable with less than 1000 traces.

The attack uses the opened state values of the two players, such as $\mathcal{P}_0$ and $\mathcal{P}_1$. Therefore the attack can be implemented independently to the deeper rounds to recover key-related information. As the scheme uses a constant plaintext for all challenges, the unshared state $\mathcal{P}_r$ is constant for every run. That means, our attack can be applied for every round to receive 30 bits key-related information. However, the interpretation of this information is not straightforward. Therefore, in the next section, we introduce an algebraic key recovery analysis in order to generate the connection between the recovered information from deeper rounds with the secret key k_s.

5.2 Algebraic Key Recovery

In Section 5, we described how to gain key-related information using a side channel analysis. In this section, we combine this information to recover the secret key.

The main idea of the analysis, is to gain information by analysing the connection between recovered states and the secret key in use. We consider the following system of linear equations $\mathcal{U}k_s = \mathcal{V}$ where k_s is an n-bit secret key, $\mathcal{U}$ is a $m \times n$ matrix (where m depends on the number of analyzed rounds) and $\mathcal{V}$ is a $n \times 1$ matrix. In the following, we explain how to produce such a system to recover the secret key k_s. To achieve this, we propose the following iterative method and we form the equations $\mathcal{U}_i k_s = \mathcal{V}_i$ using the attack on the i^{th} round to characterise the connection between secret-key and recovered-state values.

In the analysis, the notation $\mathcal{U}[i:j]$ is used to denote the rows of a matrix between i^{th} and j^{th} index. The i^{th} round state before and after `SboxLayer` is denoted by p_i and p'_i respectively. Remark that, the LowMc Sbox layer does not cover the whole state, and only 30-bit state processed by the Sbox layer i.e. $p'_i = Sbox(p_i)[1:30]||p'_i[31:128]$. We start with the attack on the initial step or *zero round attack*.

Zero Round Attack: We can recover the 30 bits of the initial key, as given in the previous section. Remark, that the recovered bits correspond to line 1 of Algorithm 1 and the recovered information is $k_0[1:30]$. Therefore, the equation can be formed as:

$$\mathcal{U}_0 = KM_0 \text{ and } \mathcal{V} \leftarrow k_0$$

The solution parameters for the initial stage are determined as follows:

$$\mathcal{U}[1:30] \leftarrow KM_0[1:30] \text{ and } \mathcal{V}[1:30] \leftarrow k_0[1:30].$$

i^{th}-Round Attack: Using the above analysis we can repeat the same procedure for the deeper rounds. First of all we reformulate the state values as follows:

$$\begin{aligned} p^b_{i+1} &= LM_i(SBox(p_i)) \oplus RC_i \oplus KM_i k_s \\ &= LM_i(p'_i[1:30] \,||\, p_i[31:128]) \oplus RC_i \oplus KM_i k_s \\ &= LM_i(p'_i[1:30] \,||\, (\mathcal{U}_{i-1}k_s \oplus t_{i-1})[31:128]) \oplus RC_i \oplus KM_i k_s \\ &= LM_i(p'_i[1:30] \,||\, t_{i-1}[31:128]) \oplus LM_i(\mathcal{Z} \,||\, (\mathcal{M}_{i-1}k_s))[31:128]) \oplus RC_i \oplus KM_i k_s \end{aligned}$$

where $\mathcal{Z}$ is a 30×128 zero matrix. Using this state representation we can define the system of linear equations $\mathcal{U}_i k_s = \mathcal{V}_i$ as:

- $\mathcal{U}_i = LM_i(\mathcal{Z}||(\mathcal{U}_{i-1}))[31:128]) \oplus KM_i$ and
- $\mathcal{V}_i = LM_i(p'_i[1:30]||t_{i-1}[31:128]) \oplus p_{i+1}[1:30] \oplus RC_i[1:30]$.

and update the state value as:

- $t_i = LM_i(p'_i[1:30]||t_{i-1}[31:128]) \oplus RC_i$

After forming the equation as above, we can collect the indexes correspond to the 30-bit state information. Hence, we update the solution matrices $\mathcal{U}$ and $\mathcal{V}$ as follows:

- $\mathcal{U}[30i+1:30(i+1)] \leftarrow \mathcal{U}_i[1:30]$

Algorithm 3 Algebraic Key Recovery

Input: The state values recovered using DPA, $p_i[1:30]$ for $i = 0, \ldots, n$.
Output: k_s

1: $\mathcal{U}_0 = KM_0$ and $t_0 = m$
2: $\mathcal{U}[1:30] \leftarrow \mathcal{U}_0[1:30]$ and $\mathcal{V}[1:30] \leftarrow k_0[1:30]$
3: **for** Round $i = 1$ to n **do**
4: $\quad \mathcal{U}_i = LM_i(\mathcal{Z}||(\mathcal{U}_{i-1})[31:128]) \oplus KM_i$
5: $\quad \mathcal{V}_i = LM_i(p'_i[1:30]||t_{i-1}[31:128]) \oplus p_{i+1}[1:30] \oplus RC_i[1:30]$
6: $\quad t_i = LM_i(p'_i[1:30]||t_{i-1}[31:128]) \oplus RC_i$
7: $\quad \mathcal{U}[30i+1:30(i+1)] = \mathcal{U}_i[1:30]$ ▷ Update the coefficient Matrix
8: $\quad \mathcal{V}[30i+1:30(i+1)] = \mathcal{V}_i[1:30]$ ▷ Update the solution Matrix
9: **Return** Solution of $\mathcal{U} \cdot k_s = \mathcal{V}$

– $\mathcal{V}[30i+1:30(i+1)] \leftarrow \mathcal{V}_i[1:30]$

This calculation holds for any round, thus we can generate equations for 30 rows of $\mathcal{U}_i$ with the solution in $\mathcal{V}_i$. After collecting enough equations, we can solve the system $\mathcal{U}k_s = \mathcal{V}$ with a simple Gaussian elimination and recover the secret key. As we gain 30 equations per observed round, 5 rounds suffice to make the search space small enough to recover the key quickly.

6 Conclusion

In this work, we provided the first side channel analysis of the Picnic signature scheme, a family of digital signature schemes secure against attacks by quantum computers. We showed that the core part of the scheme, LowMc is vulnerable to side channel attacks. By exploiting the features of the known shares for side channel analysis, we were able to recover 30-bits of each round state of LowMc. We showed how to use this information, to derive a straightforward algebraic key recovery by solving a system of linear equations. The simple algebraic attack gave us a solution space for key candidates which can be searched easily. Our analysis showed, that we can recover the secret key with less than 2000 traces for a specific challenge, depending on the attack.

We further showed, how the attack which only targets the secret sharing part can be extended to completely recover the secret key of *Picnic*. As one Picnic signature contains at least 219 calls to LowMc, as little as 30 signatures suffices to recover the secret key even if we only use traces for one particular unknown party. These results highlight the need for side channel protection for Picnic. When compared to classic block cipher protection, the MPC-in-the head rather makes the attack easier, as known inputs are available for all rounds of computation.

References

1. Frdm-k66f: Freedom development platform for kinetis. `https://www.nxp.com/downloads/en/schematics/FRDM-K66F-SCH.pdf`

2. Lf-u 2.5, h-field probe 100 khz-50 mhz. https://www.langer-emv.de/en/product/lf-passive-100-khz-50-mhz/36/lf-u-2-5-h-field-probe-100-khz-up-to-50-mhz/5
3. Picnic: Post quantum signatures. https://github.com/microsoft/Picnic
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for mpc and fhe. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 430–454. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
5. Aranha, D.F., Orlandi, C., Takahashi, A., Zaverucha, G.: Security of hedged fiat-shamir signatures under fault attacks. Cryptology ePrint Archive, Report 2019/956 (2019), https://eprint.iacr.org/2019/956
6. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Proceedings of the IEEE **94**(2), 370–382 (Feb 2006). https://doi.org/10.1109/JPROC.2005.862424
7. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults. In: Fumy, W. (ed.) Advances in Cryptology EUROCRYPT'97, Lecture Notes in Computer Science, vol. 1233, pp. 37–51. Springer Berlin Heidelberg (1997)
8. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: Hiding your white-box designs is not enough. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 215–236. Springer (2016)
9. Castelnovi, L., Martinelli, A., Prest, T.: Grafting trees: a fault attack against the sphincs framework. In: International Conference on Post-Quantum Cryptography. pp. 165–184. Springer (2018)
10. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Proceedings of the 2017 acm sigsac conference on computer and communications security. pp. 1825–1842 (2017)
11. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: International Conference on Applied Cryptography and Network Security. pp. 164–175. Springer (2005)
12. Eisenbarth, T., Von Maurich, I., Ye, X.: Faster hash-based signatures with bounded leakage. In: International Conference on Selected Areas in Cryptography. pp. 223–243. Springer (2013)
13. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems CHES 2001. pp. 251–261. Springer (2001)
14. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E.: Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In: International workshop on cryptographic hardware and embedded systems. pp. 207–228. Springer (2015)
15. Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low-bandwidth acoustic cryptanalysis. In: Advances in Cryptology–CRYPTO 2014, pp. 444–461. Springer Berlin Heidelberg (2014)
16. Giacomelli, I., Madsen, J., Orlandi, C.: Zkboo: Faster zero-knowledge for boolean circuits. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 1069–1083. USENIX Association, Austin, TX (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli

17. Inci, M.S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., Sunar, B.: Cache attacks enable bulk key recovery on the cloud. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 368–388. Springer (2016)
18. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing. pp. 21–30. STOC '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1250790.1250794, http://doi.acm.org/10.1145/1250790.1250794
19. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 206–222. Springer (1999)
20. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. pp. 388–397. CRYPTO '99, Springer-Verlag, Berlin, Heidelberg (1999), http://dl.acm.org/citation.cfm?id=646764.703989
21. Longo, J., De Mulder, E., Page, D., Tunstall, M.: Soc it to em: electromagnetic side-channel attacks on a complex system-on-chip. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 620–640. Springer (2015)
22. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag, Berlin, Heidelberg (2007)
23. Mangard, S., Oswald, E., Standaert, F.X.: One for all–all for one: unifying standard differential power analysis attacks. IET Information Security **5**(2), 100–110 (2011)
24. Park, A., Shim, K.A., Koo, N., Han, D.G.: Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations **2018**, 500–523 (Aug 2018). https://doi.org/10.13154/tches.v2018.i3.500-523, https://tches.iacr.org/index.php/TCHES/article/view/7284
25. Rohatgi, P.: Improved techniques for side-channel analysis. In: Cryptographic Engineering, pp. 381–406. Springer (2009)
26. Shor, P.W.: Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Sci. Statist. Comput. **26**, 1484 (1997). https://doi.org/10.1137/S0097539795293172
27. Taha, M., Eisenbarth, T.: Implementation attacks on post-quantum cryptographic schemes. IACR Cryptology ePrint Archive **2015**, 1083 (2015)