Breaking Panther

Christina Boura, Rachelle Heim Boissier, Yann Rotella

Breaking PANTHER

Christina Boura[1], Rachelle Heim Boissier[1], Yann Rotella[1]

Université Paris-Saclay, UVSQ, CNRS,
Laboratoire de mathématiques de Versailles, 78000,Versailles, France
christina.boura@uvsq.fr, rachelle.heim@uvsq.fr, yann.rotella@uvsq.fr

Abstract. PANTHER is a sponge-based lightweight authenticated encryption scheme published at Indocrypt 2021. Its round function is based on four Nonlinear Feedback Shift Registers (NFSRs). We show here that it is possible to fully recover the secret key of the construction by using a single known plaintext-ciphertext pair and with minimal computational ressources. Furthermore, we show that in a known ciphertext setting an attacker is able with the knowledge of a single ciphertext to decrypt all plaintext blocks expect for the very first ones and can forge the tag with only one call and probability one. As we demonstrate, the problem of the design comes mainly from the low number of iterations of the round function during the absorption phase. All of our attacks have been implemented and validated.

1 Introduction

PANTHER is a sponge-based lightweight AEAD scheme designed by Bhargavi, Srinivasan and Lakshmy [3] and published at Indocrypt 2021. This construction works on a 328-bit state, divided as an outer part of rate $r = 64$ bits and an inner part of capacity $c = 264$ bits. The state is updated by iterating a function F that is composed of four interconnected NFSRs of sizes 19, 20, 21 and 22 nibbles respectively. This function is iterated 92 times for the initialization and the finalization part and only 4 times after the absorption of associated data (AD) or plaintext blocs or after the extraction of a block of the tag. The authors present their construction as lightweight, even if no concrete performance comparison is given with similar AEAD schemes.

A preliminary security analysis of PANTHER against various attacks is given in the document and the authors conclude that their construction is immune against all of the explored cryptanalysis techniques. They claim thus a security of $2^{c/2} = 2^{132}$. However, we show in this article that PANTHER has an important flaw that permits devastating attacks against it. More precisely, we demonstrate that due to the low number of iterations of the function F in all the middle computations, some public information

goes directly into the inner state. This fact has several consequences. In the known plaintext model, the attacker is for example able to inverse the state and to recover the secret key. In the known ciphertext (only) mode, it is possible to recover all plaintext blocks but the first six ones and also to forge the tag. A particularity of all our attacks is that they only need a single plaintext/ciphertext or a single ciphertext and the computation time is equivalent (or sometimes even smaller) to one encryption with PANTHER. As we will show, the main conclusion of this paper is that when using shift registers to build permutation-based constructions, one should at least take as many rounds as the size of the registers.

The rest of the paper is organized as follows. In Section 2 we provide the specifications of PANTHER and introduce some notations. Next, in Section 3 we describe our central observation on the diffusion of the cipher. Section 4 is dedicated to our attacks and finally we briefly describe in Section 5 their implementation.

2 Specification of PANTHER

The design of PANTHER is based on the sponge construction [1, 2]. Its central component is a function called F that applies to a 328-bit state. The state $\mathcal{S}$ is divided into a r-bit outer part $\overline{\mathcal{S}}$ and a c-bit inner part $\widehat{\mathcal{S}}$, where $r = 64$ is called the rate and $c = 264$ the capacity. The encryption works as follows.

Initialization phase. First, in the initialization phase, the 128-bit key and the 128-bit initial value (IV) are loaded to the state. More precisely, if we denote by k_i, $0 \leq i < 128$ the 128 bits of the secret key K, by iv_i, $0 \leq i < 128$ the 128 bits of the IV and by $\overline{x}$ the Boolean complement of the binary value x, the initial state is loaded with the following vector:

$$(k_0, \ldots, k_{127}, iv_0, \ldots, iv_{127}, \overline{k_0}, \ldots, \overline{k_{63}}, 1, 1, 1, 1, 1, 1, 1, 0).$$

The state is then updated 92 times by the function F.

Absorption phase. Both associated data and plaintexts are then processed in the absorption phase. First, associated data (AD) are incorporated to the state. This part processes data that only need to be authenticated and not necessarily encrypted. This is done by dividing the data into k blocks AD_i of 64 bits each, XORing each block to the outer part of the state and by applying next the permutation F four times.

This is repeated until all the AD blocks have been absorbed. This part can of course be omitted if there is no associated data to authenticate. Next, plaintext blocks are processed and ciphertext blocks are generated. For this, the plaintext is divided into n blocks of 64 bits and each block is absorbed by the outer part of the state. Once a plaintext block P_i is absorbed, a 64-bit ciphertext block C_i is immediately generated by outputting the outer part of the state. Four iterations of the permutation F are next applied for all blocks except for the last one.

Finalization phase. Once all plaintext blocks have been processed, the finalization mode is activated, during which the tag is generated. For this, 92 rounds of the permutation F are first applied to the state. Then, the outer part of the state is outputted as a first block of the tag and four rounds of the permutation F are next applied. The other blocks of the tag are generated in the same manner until a tag of the desired length is obtained. This procedure can be visualized in Figure 1.

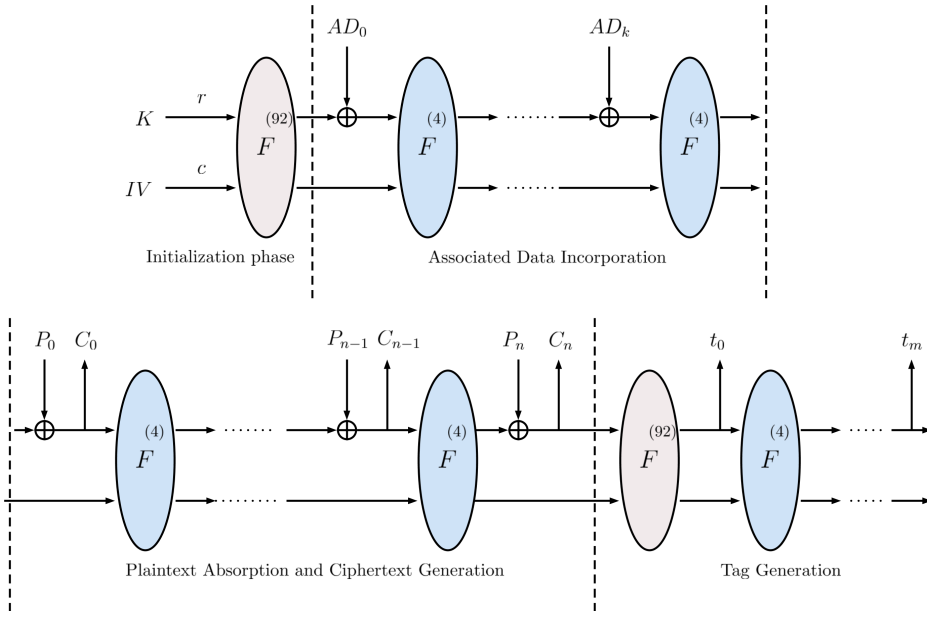

Fig. 1. PANTHER's global structure

2.1 State update function F

The cipher's function F applies to the 328-bit state. This state can be seen as 82 nibbles split into four unequally-sized registers P, Q, R and S. The register P contains the 19 nibbles $P_{18}, \ldots, P_0$, the register Q the 20 nibbles $Q_{19}, \ldots, Q_0$, the register R the 21 nibbles $R_{20}, \ldots, R_0$ and finally the register S the 22 nibbles $S_{21}, \ldots, S_0$. The outer part $\overline{\mathcal{S}}$ is composed by the last four nibbles of each register :

$$\begin{aligned}\overline{\mathcal{S}} = (&P_{15}, P_{16}, P_{17}, P_{18}, Q_{16}, Q_{17}, Q_{18}, Q_{19}, R_{17}, R_{18}, R_{19}, R_{20},\\ &S_{18}, S_{19}, S_{20}, S_{21}).\end{aligned}$$

We use the following notation to denote an arbitrary state of PANTHER, where everything left to the symbol $\|$ is the outer state.

$$\begin{aligned}P_{18}||P_{17}||P_{16}||P_{15} \,\|\, P_{14}||...||P_0\\ Q_{19}||Q_{18}||Q_{17}||Q_{16} \,\|\, Q_{15}||Q_{14}||...||Q_0\\ R_{20}||R_{19}||R_{18}||R_{17} \,\|\, R_{16}||R_{15}||R_{14}...||R_0\\ S_{21}||S_{20}||S_{19}||S_{18} \,\|\, S_{17}||S_{16}||S_{15}||S_{14}||...||S_0\end{aligned}$$

This state is then loaded into four interconnected NFSRs, each NFSR containing the values of the registers P, Q, R and S respectively, as can be seen in Figure 2. We provide here the full specification of the round function, even if most of these details are not needed to understand our attack nor have any effect on it.

The function F can be described as follows. First the 4-bit values f_p, f_q, f_r and f_s, each one corresponding to the feedback polynomial of the corresponding NFSR are computed:

$$\begin{aligned}f_p &= P_0 \oplus P_7 \oplus P_{10} \oplus P_6 \otimes P_{18}\\ f_q &= Q_0 \oplus Q_4 \oplus Q_6 \oplus Q_7 \oplus Q_{15} \oplus Q_3 \otimes Q_7\\ f_r &= R_0 \oplus R_1 \oplus R_{15} \oplus R_{17} \oplus R_{19} \oplus R_{13} \otimes R_{15}\\ f_s &= S_0 \oplus S_1 \oplus S_4 \otimes S_{10} \oplus S_{11} \otimes S_{18}\end{aligned}$$

Here, the symbol $\otimes$ corresponds to the multiplication in the field $GF(2^4)$, where the field is constructed by using the polynomial x^4+x^3+1.

Next, four interconnection polynomials, g_p, g_q, g_r and g_s, mixing nibbles from different registers are computed:

$$\begin{aligned} g_p &= Q_9 \oplus R_{10} \oplus S_{12} \\ g_q &= P_4 \oplus R_2 \oplus S_5 \\ g_r &= P_{12} \oplus Q_{11} \oplus S_{16} \\ g_s &= P_{16} \oplus Q_{17} \oplus R_2 \end{aligned}$$

Next one computes the 4-bit values ℓ_1, ℓ_2, ℓ_3 and ℓ_4, each one corresponding to the XOR of the values f_* and g_* together with a constant rc_i:

$$\begin{aligned} \ell_1 &= f_p \oplus g_p \oplus rc_1 \\ \ell_2 &= f_q \oplus g_q \oplus rc_2 \\ \ell_3 &= f_r \oplus g_r \oplus rc_3 \\ \ell_4 &= f_s \oplus g_s \oplus rc_4, \end{aligned}$$

where the constant values are $rc_1 = \mathtt{7}, rc_2 = \mathtt{9}, rc_3 = \mathtt{b}, rc_4 = \mathtt{d}$ given in hexadecimal notation. After this, the vector $[\ell_1, \ell_2, \ell_3, \ell_4]^T$ is multiplied by a Toeplitz MDS matrix T_p to create the 16-bit vector $[d_1, d_2, d_3, d_4]^T = T_p \times [\ell_1, \ell_2, \ell_3, \ell_4]^T$. An 4-bit S-box Sb is then applied to each one of the nibbles d_1, d_2, d_3 and d_4 and the resulting 16-bit vector is then multiplied again by the matrix T_p:

$$[t_1, t_2, t_3, t_4]^T = T_p \times [Sb[d_1), Sb(d_2), Sb(d_3), Sb(d_4)]^T.$$

As the specification of the matrix T_p is not relevant to our attack, we omit its description here. Finally, the registers P, Q, R and S are shifted by one nibble to the right and the most-significant nibbles of each NFSR are updated by the values t_1, t_2, t_3 and t_4:

$$\begin{aligned} &P \gg 1, Q \gg 1, R \gg 1, S \gg 1 \\ &P_{18}, Q_{19}, R_{20}, S_{21} = t_1, t_2, t_3, t_4 \end{aligned}$$

F is applied successively a certain number of times n_r, where the value of n_r depends on the phase considered. In the initialization phase and before the first block of tag is outputted, n_r equals 92, while for all other applications of F, n_r equals 4.

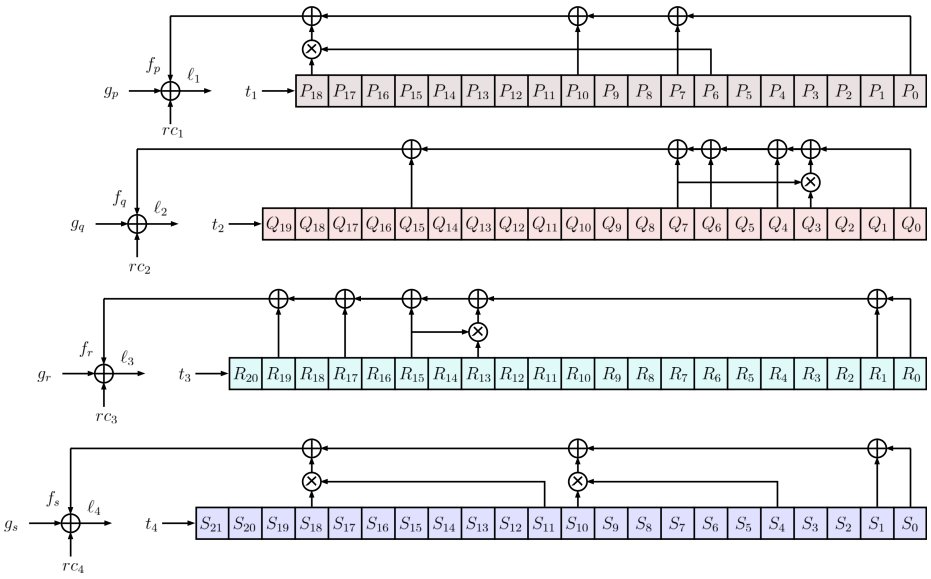

Fig. 2. Function F

3 Main observation on PANTHER

In this section, we make an observation on PANTHER that is at the core of all the attacks provided next. The property that we exhibit is, as we will show, caused by the low number of applications of the state update function F in the plaintext absorption and ciphertext generation phase and has as a consequence to greatly alter the security of the cipher.

3.1 An observation on F^4

As stated above, our main observation stems from the very low number $n_r = 4$ of times F is applied between each ciphertext output. We thus exhibit a very simple observation on F^4.

We begin by studying one application of F. Let us denote the input nibbles to F by

$$\begin{aligned} P_{18}||P_{17}||P_{16}||P_{15} \,\|\, P_{14}||\cdots||P_0 \\ Q_{19}||Q_{18}||Q_{17}||Q_{16} \,\|\, Q_{15}||Q_{14}||\cdots||Q_0 \\ R_{20}||R_{19}||R_{18}||R_{17} \,\|\, R_{16}||R_{15}||R_{14}\cdots||R_0 \\ S_{21}||S_{20}||S_{19}||S_{18} \,\|\, S_{17}||S_{16}||S_{15}||S_{14}||\cdots||S_0 \end{aligned} \tag{1}$$

As can be observed in Figure 2, because of the action of the four NFSRs to the state, the output of F is of the form

$$\begin{aligned} X_0||P_{18}||P_{17}||P_{16} \,\|\, P_{15}||\cdots||P_1 \\ Y_0||Q_{19}||Q_{18}||Q_{17} \,\|\, Q_{16}||Q_{15}||\cdots||Q_1 \\ Z_0||R_{20}||R_{19}||R_{18} \,\|\, R_{17}||R_{16}||R_{15}||\cdots||R_1 \\ T_0||S_{21}||S_{20}||S_{19} \,\|\, S_{18}||S_{17}||S_{16}||S_{15}||\cdots||S_1 \end{aligned} \tag{2}$$

where X_0, Y_0, Z_0, T_0 are 4-bit values depending on the input nibbles. As the exact expression of these values has no impact on our attacks we do not provide their details here, but those can be found in Section 2.1.

Thus, note that each time the state is updated, only one nibble per register is modified. The values of the other nibbles remain unchanged, they are simply shifted. As a consequence, after four state updates, only four nibbles per register have been properly modified whilst all the others remain unchanged and are simply shifted to the right. By repeating the same analysis for the following rounds, we can see that the output of F^4 is of the form

$$\begin{aligned} X_3||X_2||X_1||X_0 \,\|\, P_{18}||\cdots||P_4 \\ Y_3||Y_2||Y_1||Y_0 \,\|\, Q_{19}||Q_{18}||\cdots||Q_4 \\ Z_3||Z_2||Z_1||Z_0 \,\|\, R_{20}||R_{19}||R_{18}||\cdots||R_4 \\ T_3||T_2||T_1||T_0 \,\|\, S_{21}||S_{20}||S_{19}||S_{18}||\cdots||S_4 \end{aligned} \tag{3}$$

where the X_i, Y_i, Z_i, T_i for $0 \leq i \leq 3$ depend on the nibbles of the input state (again, their actual expression is not of interest, for more details see the specification of F in Section 2.1).

In particular, note that the outer part nibbles of the initial state are among those nibbles that have not been modified, but simply shifted into the inner part. For a more visual representation, we colour in red the nibbles of the outer part of the input that have been moved to the inner part of the output of F^4 :

$$\begin{aligned} X_3||X_2||X_1||X_0 \,\|\, P_{18}||P_{17}||P_{16}||P_{15}||\cdots||P_4 \\ Y_3||Y_2||Y_1||Y_0 \,\|\, Q_{19}||Q_{18}||Q_{17}||Q_{16}||Q_{15}||\cdots||Q_4 \\ Z_3||Z_2||Z_1||Z_0 \,\|\, R_{20}||R_{19}||R_{18}||R_{17}||R_{16}||R_{15}||\cdots||R_4 \\ T_3||T_2||T_1||T_0 \,\|\, S_{21}||S_{18}||S_{17}||S_{16}||S_{17}||S_{16}||S_{15}||\cdots||S_4 \end{aligned} \tag{4}$$

3.2 Consequences in a known ciphertext only setting

At the end of the initialisation phase, the state is *a priori* unknown since the key has been mixed in with the IV by the application of F^{92}. The absorption of the associated data which follows does not reveal anything about the state at the beginning of the plaintext absorption/ciphertext generation phase either. However, as soon as ciphertext blocks start to be outputted, an attacker has knowledge of the outer part of the input state to each application of F^4.

If we recall the observations on F^4 made above, the outer part of the input state to F^4 is not modified but simply shifted into the inner state. Let $C = C_0||\cdots||C_{n-1}$ be the known ciphertext of an unknown padded plaintext $M = M_0||\cdots||M_{n-1}$ where $|C_i| = |M_i| = 64$ for $0 \leq i < n$. An output of one ciphertext block C_{i-1} thus not only leaks information on the outer part of the state at the entry of F^4, but also on the inner part of the output of F^4. As the next message block M_i is then XORed only to the outer part of the output state, when the next ciphertext block C_i is outputted, the attacker knows not only the outer part of the state but also 64 bits of the inner state. As more ciphertext blocks are outputted, more information on the inner state is given to the attacker. Once 6 consecutive ciphtertext blocks $C_{i-1}, \ldots, C_{i+4}$ have been outputted, the attacker knows the whole inner state and the whole outer state. The property is illustrated in Figure 3 with $C_0, \ldots, C_5$.

We show this property in a more formal way for the first ciphertext outputs $C_0, \ldots, C_5$. We consider the state at the beginning of the plaintext absorption and ciphertext generation. In the following, we use the color blue to put forward what the attacker knows (which corresponds to the ciphertext blocks). Once the first ciphertext is outputted, the entry to F^4 is as follows :

$$
\begin{aligned}
C_3^0||C_2^0||C_1^0||C_0^0 \,\|\, P_{14}||\cdots||P_0 \\
C_7^0||C_6^0||C_5^0||C_4^0 \,\|\, Q_{15}||Q_{14}||\cdots||Q_0 \\
C_{11}^0||C_{10}^0||C_9^0||C_8^0 \,\|\, R_{16}||R_{15}||R_{14}||\cdots||R_0 \\
C_{15}^0||C_{14}^0||C_{13}^0||C_{12}^0 \,\|\, S_{17}||S_{16}||S_{15}||S_{14}||\cdots||S_0
\end{aligned}
$$

After the application of F^4, the state is of the following form :

$$\begin{aligned}
X_3||X_2||X_1||X_0 \,\|\, C_3^0||C_2^0||C_1^0||C_0^0||P_{14}||\cdots||P_4\\
Y_3||Y_2||Y_1||Y_0 \,\|\, C_7^0||C_6^0||C_5^0||C_4^0||Q_{15}||Q_{14}||\cdots||Q_4\\
Z_3||Z_2||Z_1||Z_0 \,\|\, C_{11}^0||C_{10}^0||C_9^0||C_8^0||R_{16}||R_{15}||R_{14}||\cdots||R_4\\
T_3||T_2||T_1||T_0 \,\|\, C_{15}^0||C_{14}^0||C_{13}^0||C_{12}^0||S_{17}||S_{16}||S_{15}||S_{14}||\cdots||S_4
\end{aligned}$$

The message is then XORed to the nibbles in blue, and the outer part thus takes the value of the outputted C^1. Therefore, the state has the following form just before the next application of F^4:

$$\begin{aligned}
C_3^1||C_2^1||C_1^1||C_0^1 \,\|\, C_3^0||C_2^0||C_1^0||C_0^0||P_{14}||\cdots||P_4\\
C_7^1||C_6^1||C_5^1||C_4^1 \,\|\, C_7^0||C_6^0||C_5^0||C_4^0||Q_{15}||Q_{14}||\cdots||Q_4\\
C_{11}^1||C_{10}^1||C_9^1||C_8^1 \,\|\, C_{11}^0||C_{10}^0||C_9^0||C_8^0||R_{16}||R_{15}||R_{14}||\cdots||R_4\\
C_{15}^1||C_{14}^1||C_{13}^1||C_{12}^1 \,\|\, C_{15}^0||C_{14}^0||C_{13}^0||C_{12}^0||S_{17}||S_{16}||S_{15}||S_{14}||\cdots||S_4
\end{aligned}$$

As C_2 is outputted, the attacker knows 128 bits of the inner state as well as the whole outer state. This phenomenon goes on iteratively: as more consecutive ciphertexts get known, more information is given to the attacker. Once the attacker knows the 6 first ciphertext blocks $C_0, \ldots, C_5$, the attacker knows the whole inner state and the whole outer state. For a visual representation of the property, see Figure 3. In general, leaks on the value of the inner state of a sponge-based cipher have a devastating effect on the security of this cipher. In the case of PANTHER, the attacker recovers the value of the whole inner state and, depending on the attack settings, controls or knows the outer state. Unsurprisingly, this weakness will allow an attacker to mount extremely powerful key-recovery, plaintext-recovery and forging attacks as described in the next section.

4 Cryptanalysis of PANTHER

In this section, we show how the observation of Section 3 allows us to mount three attacks including a known plaintext key recovery attack, a known ciphertext-only attack and a chosen ciphertext-only forge. Note that each of these three attacks is extremely powerful, as they simply require the knowledge of either one plaintext/ciphertext pair or of a single ciphertext.

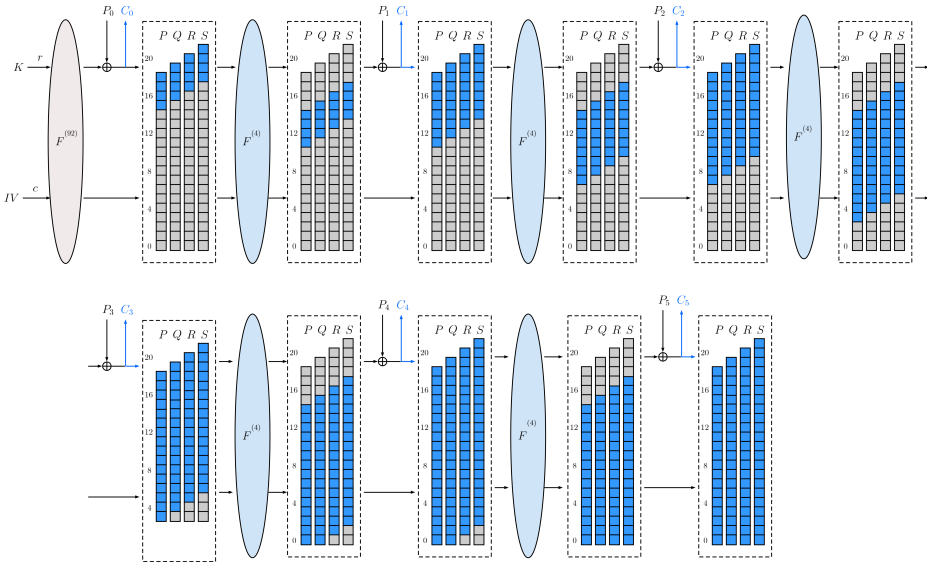

Fig. 3. Attack on PANTHER. The blue nibbles correspond to the nibbles known to the attacker, while the grey nibbles are values that are a priori unknown.

4.1 Key-recovery attack with one plaintext/ciphertext pair

We start by describing the most powerful of our attacks, namely a known plaintext attack which recovers the full key with a single plaintext/ciphertext pair. This attack, as also all the following ones, is memoryless and its time complexity is equivalent to a single encryption or decryption with PANTHER. This attack is a direct consequence of our observation from Section 3.

The only constraint on the pair is that the padded message M must contain at least six 64-bit blocks. As shown in Section 3, the attacker recovers the full state as soon as she knows six consecutive ciphertext blocks. Once the full state is known, one can recover the full key as F is a permutation and its inverse can be very easily computed. As the attacker can invert F and knows all the message blocks M_i (and the optional associated data blocks), she can recover the initial state and thus the key. The fact that F is a permutation is not explicitly mentioned by the authors. Thus, we provide a short proof at the end of this section. From this proof one can easily deduce how to invert F.

We've shown that with only one plaintext/ciphertext pair, an attacker can recover the key with a very easy and straightforward procedure. The

attack is memoryless and, as for time, it is equivalent to a single encryption or decryption with PANTHER.

Proof that F is a permutation Let

$$I = P_{18}||\cdots||P_0||Q_{19}||\cdots||Q_0||S_{20}||\cdots||S_0||R_{21}||\cdots||R_0 \in \mathbb{F}_2^{328}$$

be an input to F, and let

$$O = P'_{18}||\cdots||P'_0||Q'_{19}||\cdots||Q'_0||S'_{20}||\cdots||S'_0||R'_{21}||\cdots||R'_0$$

be its image by F. We show that I is uniquely determined by O.

First, note that all $P_i, 0 < i \leq 18$, $Q_j, 0 < j \leq 19$, $R_k, 0 < k \leq 20$ and $S_\ell, 0 < \ell \leq 21$ are uniquely determined by O since

$$\begin{aligned} P_i &= P'_{i-1} \quad \text{for} \quad 0 < i \leq 18 \\ Q_j &= Q'_{j-1} \quad \text{for} \quad 0 < j \leq 19 \\ R_k &= R'_{k-1} \quad \text{for} \quad 0 < k \leq 20 \\ S_\ell &= S'_{\ell-1} \quad \text{for} \quad 0 < \ell \leq 21 \end{aligned}$$

Thus, we now only need to show that P_0, Q_0, R_0 and S_0 are uniquely determined by O. First, note that $(P'_{18}, Q'_{19}, R'_{20}, S'_{21})$ uniquely determines the value of

$$\begin{aligned} \ell_1 &= P_0 \oplus P_7 \oplus P_{10} \oplus P_6 \otimes P_{18} \oplus Q_9 \oplus R_{10} \oplus S_{12} \oplus rc_1 \\ \ell_2 &= Q_0 \oplus Q_4 \oplus Q_6 \oplus Q_7 \oplus Q_{15} \oplus Q_3 \otimes Q_7 \oplus P_4 \oplus R_2 \oplus S_5 \oplus rc_2 \\ \ell_3 &= R_0 \oplus R_1 \oplus R_{15} \oplus R_{17} \oplus R_{19} \oplus R_{13} \otimes R_{15} \oplus P_{12} \oplus Q_{11} \oplus S_{16} \oplus rc_3 \\ \ell_4 &= S_0 \oplus S_1 \oplus S_4 \otimes S_{10} \oplus S_{11} \otimes S_{18} \oplus P_{16} \oplus Q_{17} \oplus R_2 \oplus rc_4 \end{aligned}$$

as both the matrix T_p and the S-box Sb are invertible (T_p being MDS). Since the $P_i, 0 < i \leq 18$, $Q_j, 0 < j \leq 19$ $R_k, 0 < k \leq 20$ and $S_\ell, 0 < \ell \leq 21$ are also uniquely determined by O as shown just above, it comes that

$$\begin{aligned} P_0 &= \ell_1 \oplus P_7 \oplus P_{10} \oplus P_6 \otimes P_{18} \oplus Q_9 \oplus R_{10} \oplus S_{12} \oplus rc_1 \\ Q_0 &= \ell_2 \oplus Q_4 \oplus Q_6 \oplus Q_7 \oplus Q_{15} \oplus Q_3 \otimes Q_7 \oplus P_4 \oplus R_2 \oplus S_5 \oplus rc_2 \\ R_0 &= \ell_3 \oplus R_1 \oplus R_{15} \oplus R_{17} \oplus R_{19} \oplus R_{13} \otimes R_{15} \oplus P_{12} \oplus Q_{11} \oplus S_{16} \oplus rc_3 \\ S_0 &= \ell_4 \oplus S_1 \oplus S_4 \otimes S_{10} \oplus S_{11} \otimes S_{18} \oplus P_{16} \oplus Q_{17} \oplus R_2 \oplus rc_4 \end{aligned}$$

are also uniquely determined by O. We've shown that F is injective which is sufficient to prove that it is a permutation. Further, it is easy to see from this proof how to invert F.

4.2 Plaintext-recovery attack with one known ciphertext

In this section, we show how our observation on F^4 also allows one to mount attacks in a known ciphertext only setting. Even if this attack does not recover the secret key, it is nevertheless devastating as it allows the attacker to recover full plaintext blocks. More precisely, for any padded message $M = M_0||\cdots||M_{n-1}$ where $|M_i| = 64$ for all i and such that $n \geq 6$, one can fully recover all plaintext blocks from the seventh on.

As shown in the previous sections, knowing the six first ciphertext blocks allows one to recover the full state. Thus, the attacker also knows the full state after another application of F^4, that is when the rest of the plaintext blocks $M_i, i \geq 6$ are absorbed. To recover these blocks, the attacker only needs to XOR the known ciphertext block $C_i, i \geq 6$ to the outer part of the state after each application of F^4. With only known ciphertext of sufficient length, an attacker can recover the whole plaintext except for the first 384 bits. This attack is also memoryless and requires only one ciphertext. Concerning the time complexity, it is striking that this attack is more efficient than a decryption since one does not need to go through the initialisation phase and the absorption of the associated data.

4.3 Forging attacks

Last but not least, our observation on F^4 can also allow one to launch forging attacks both in a known plaintext and in a known ciphertext only setting.

To begin, an attacker with access to plaintext/ciphertext pairs can recover the key with the method described in Section 4.1. Thus, she can generate a valid tag for any chosen plaintext and any chosen ciphertext.

In the known ciphertext only setting, let us consider a ciphertext C composed of n blocks of 64 bits with $n \geq 6$ and let T be the valid tag for C. As explained in Section 3.2, when six consecutive ciphertext blocks are outputted, the full state is known by the attacker. In particular, the value of the last six ciphertext blocks fully determines the state at the end of the absorption phase. Thus, for any ciphertext C' composed of m blocks such that $m \geq 6$, if the six last blocks of C' are equal to those of C, the two states will fully collide at the end of the ciphertext generation phase, and thus at the beginning of the tag generation phase. It stems that T is also a valid tag for C'.

As the other attacks presented above, this forging attack is very powerful as it is memoryless and requires only one valid ciphertext/tag pair.

The time complexity is also negligible, as the attacker does not even need to apply F once. The forged ciphertexts can have any length as long as they have at least 6 blocks, and only the last 6 blocks are constrained. As a consequence, one can build as many valid ciphertext/tag pairs as they wish. Forging is thus not only possible but also very easy and with a large degree of freedom for the attacker.

5 Implementation

All of the described attacks need negligible memory and computational ressources and can thus easily be implemented. Therefore, we implemented all of them in `C` in order to confirm their validity. Our code is accessible online[1].

Our program works in the following way. First, a random 128-bit key, a random 128-bit initial value (IV) and a random 512-bit plaintext are generated. Since PANTHER has a rate of 64 bits, the plaintext is processed in 8 blocks. The code does not generate associated data as our attacks work regardless. The plaintext is then encrypted with the key and IV and the corresponding ciphertext and tag are returned. The three attacks are then launched.

Key recovery. First, we implemented a function that takes as input the plaintext/ciphertext pair and returns the secret key. The program verifies that the key returned matches with the random secret key that was generated.

Plaintext recovery. Second, we implemented a function that takes as input the ciphertext and returns all plaintext blocks but the first six. Once the plaintext blocks are recovered, the program verifies that they match the actual plaintext encrypted.

Forge. Lastly, we implemented a function that takes as input the ciphertext and returns a forged ciphertext which has the same tag. We then implemented a function that checks whether the forged ciphertext is valid. This function takes as input the key and the IV. It works as a decryption function on the forged ciphertext and returns the valid 128-bit tag for the forged ciphertext. The program then verifies that the forged ciphertext tag matches the initial ciphertext tag.

[1] https://github.com/panthercryptanalyst/Panther-cryptanalysis

5.1 Repairing PANTHER

The main problem in the cipher's design comes from the fact that the number of rounds that the function F needs to be iterated in the middle computation was wrongly estimated. While determining the least number of rounds for the cipher to resist all known attacks is not an easy task, a minimum requirement is that the function F^r provides full diffusion, in the sense that at the end of the computation every output bit depends on all input bits. Computing the minimal round r ensuring a full diffusion for F^r is an easy procedure that we implemented. The code can be found together with the attacks code. This simple computation permitted us to affirm that the minimal number of rounds for reaching full diffusion is 46.

We can therefore conclude that at least 46 rounds are needed in the middle part of the cipher in order to resist the presented attacks. Of course, this minimal number of rounds does not necessarily guarantee the resistance of the cipher against other attacks, for example those exploiting a low algebraic degree. To determine this, a more in-depth analysis of the structure of F is required but such an analysis is out-of-scope of the current article.

.

6 Conclusion

In this paper we showed several devastating attacks in different scenarios against the AEAD scheme PANTHER. All of our attacks are extremely powerful as they are memoryless, require a single plaintext/ciphertext or a single ciphertext and have negligible execution time. This work shows that this design cannot be used in its current form to securely transmit data. We also demonstrate that special care is required when combining the sponge construction with an NFSR-based update function. More precisely, the inner part should always remain secret in sponge-like constructions, hence, when using shift registers, the number of rounds should at least be the size of the register, so that all bits in the inner part cannot be deduced from the ciphertext.

We believe that modifying PANTHER in order for it to resist our attacks requires to greatly increase the number of rounds of the update function (from 4 to at least 46) in order to get full diffusion. However, in this scenario, the lightweight character of the cipher will very probably not be ensured any more, limiting thus the interest of someone to use it.

References

1. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge Functions. Ecrypt Hash Workshop 2007 (May 2007), available at https://keccak.team/files/SpongeFunctions.pdf
2. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Cryptographic sponge functions (2011), https://keccak.team/files/CSF-0.1.pdf
3. Bhargavi, K.V.L., Srinivasan, C., Lakshmy, K.V.: Panther: A sponge based lightweight authenticated encryption scheme. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) INDOCRYPT 2021. Lecture Notes in Computer Science, vol. 13143, pp. 49–70. Springer (2021)