Iterated Truncated Differential for Internal Keyed Permutation of FLEXAEAD

Mostafizar Rahman[1], Dhiman Saha[2], Goutam Paul[1]

[1]Cryptology and Security Research Unit (CSRU), R. C. Bose Centre for Cryptology and Security,
Indian Statistical Institute, Kolkata 700108, India
mrahman454@gmail.com,goutam.paul@isical.ac.in
[2]Department of Electrical Engineering & Computer Science,
Indian Institute of Technology, Bhilai 492015, India
dhiman@iitbhilai.ac.in

Abstract. In this draft, the internal *keyed* permutation of FLEXAEAD has been analysed. In our analysis, we have first reported an iterated truncated differential for one round which holds with a probability of 2^{-7} and can penetrate same number of rounds as claimed by the designers with much less complexity which can be easily converted to a key-recovery attack. We have also reported a Super-Sbox construction in the internal permutation, which has been exploited using the Yoyo game to devise a 6-round deterministic distinguisher and a 7-round key recovery attack for 128-bit internal permutation. Similar attacks can be mounted for 64-bit and 256-bit internal permutation.

Keywords: FLEXAEAD , Distinguisher, Iterated Differential, Yoyo, NIST lightweight cryptography project

1 State Representation

We refer to internal *keyed* permutation (PF_K) of FLEXAEAD [EMdN19] as the FLEX-x, where x is the block size of the permutation in bits. **Fig 1** shows the byte representation in FLEX-128 state. Each state is divided into two equal halves- the bytes in the left half being numbered from B_0 to B_7, and the ones on the right half from B_8 to B_{15}. Each byte is divided into two parts representing the two nibbles with the upper half (upper nibble) being the most significant one. The other nibble is called as lower nibble. After the block shuffle operation, the 16 nibbles from B_0 to B_7 constitute the upper nibbles of each bytes whereas the nibbles from B_8 to B_{15} constitute the lower ones. The bytes at position B_i and $B_{(i+8)}$ are referred to as a "pair of symmetric bytes". Application of `BlockShuffle` operation on p in r-th round is denoted by $BS^r(p)$.

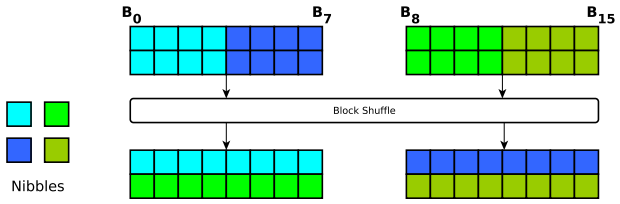

Figure 1: Byte Representation of FLEX-128 Block Cipher

2 Super-Sbox

Refer to **fig 2** for the Super-Sbox construction in FLEX-128 block cipher. Consider the bytes $\{B_0, B_2, \cdots B_{14}\}$ at X_1. Due to round function, only the symmetric bytes affect each other. So, in Y_1 every symmetric bytes depends on every symmetric bytes at X_1. Due to BS^2, B_{2i}, B_{2i+8} $(0 \leq i \leq 3)$ from Y_1 constitutes the B_{4i}, B_{4i+1} $(0 \geq i \leq 3)$ at X_2. Due to application of BS^3, $\{B_{2i}, B_{2i+1}, B_{2i+8}, B_{2i+9}\}$, $(0 \leq i \leq 1)$ at Y_2 affects $\{B_{8i}, B_{8i+1}, B_{8i+2}, B_{8i+3}\}$, $(0 \leq i \leq 1)$ at X_3. This constitutes a Super-Sbox which spans over 2.5 rounds (omitting the initial Byte Shuffle). There are two Super-Sboxes in the FLEX-128 state.
In similar way, FLEX-64 and FLEX-256 has Super-Sboxes which span over 1.5 and 3.5 rounds respectively.

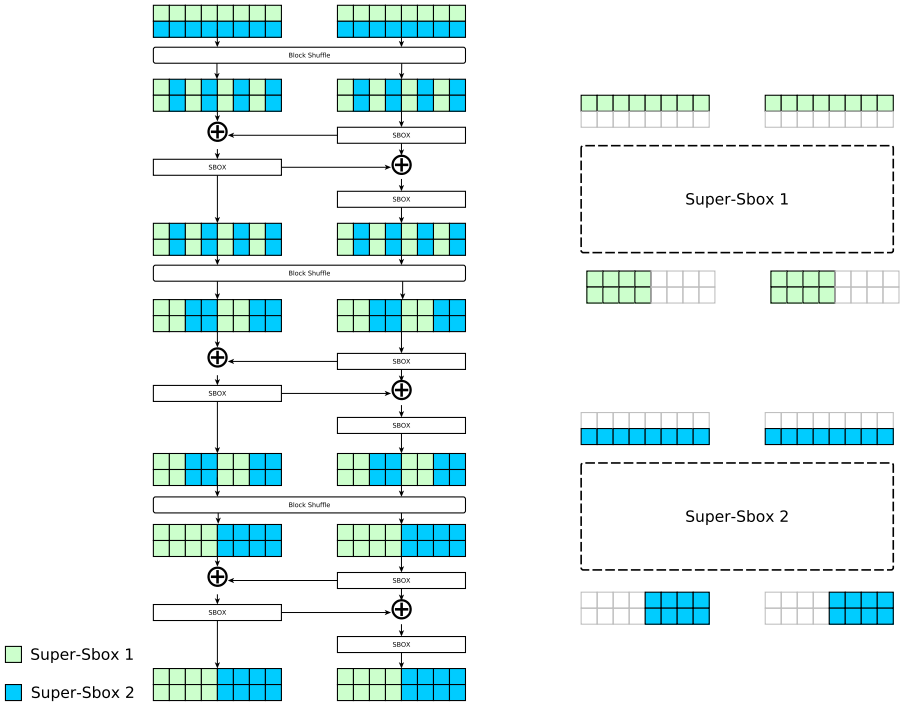

Figure 2: Super-Sbox of FLEX-128 Block Cipher

3 Iterated Differential

3.1 Property of AES DDT Table

From AES DDT table it has been observed that the number of randomly chosen input difference maps to an output difference, such that output difference bytes are confined to the upper nibble is 4096. Therefore, with probability 2^{-4} a random input difference transits to upper nibble in the output difference. With same probability random input difference transits to lower nibble.

3.2 One Round Probabilistic Iterated Truncated Differential

Refer to **Fig. 3** for the iterated differential. In X_1, keeping the difference in B_0 ensures that in Y_1 difference are in B_0 and B_8. With probability 2^{-7} both differences are confined

in either upper nibble or lower nibble in those bytes. Therefore, after `BlockShuffle` only one byte is active.

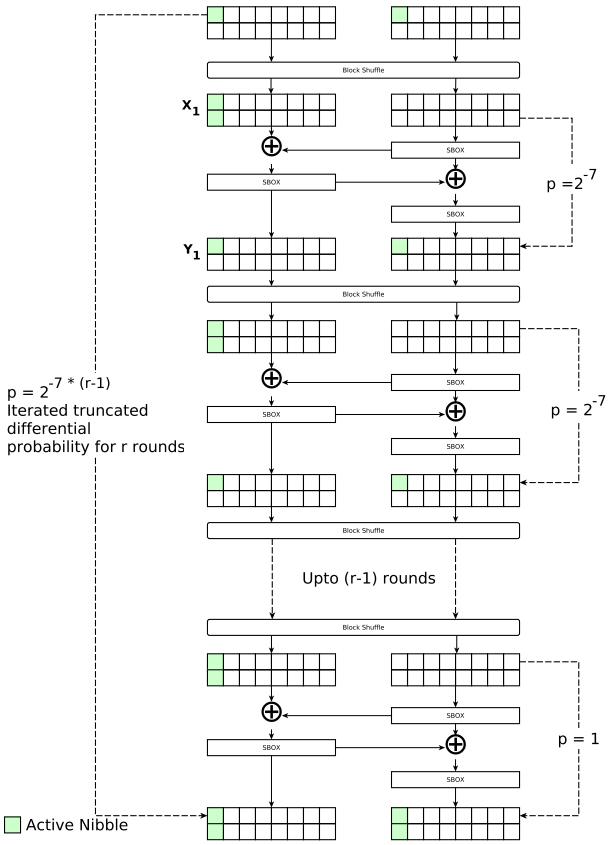

Figure 3: Iterated Truncated Differential with One-round probability of 2^{-7}. Note that the key-addition is not shown, since it has no effect on the trail.

3.3 Application to Variants of FlexAEAD Permutation

The one round iterated truncated differential can be applied to all the versions of internal *keyed* permutation. The iterated differential occurs with probability 2^{-7}. Also, the last round can be made free as no byte to nibble transition is needed for the last round. So, for r number of rounds, $2^{-7\times(r-1)}$ probability need to be paid. Table 1 compares the differential probabilities claim of the designers with our claim using the iterated differential. $\mathcal{P}_D$ denotes the designers claim whereas $\mathcal{Q}_D$ denotes our claim. We have experimentally verified the attacks upto 5 rounds.

3.4 Key Recovery Using Iterated Truncated Differential

At the end of each round, difference in a pair of symmetric bytes after Sbox transits to same nibble with probability 2^{-7}. This has been used as a filtering technique to eliminate wrong key bytes. Let, the first subkey, K_0 for Flex-128 is being recovered. Using iterated truncated differential for r rounds a right pair can be identified with probability $2^{-7*(r-1)}$. Suppose, in the right pair the initial difference is in B_i and $B_{(i+8)}$. So, we guess key byte $K_0[i]$ and $K_0[i+8]$. There are 2^{16} possible guesses and these are used to verify whether at the end of first round byte to nibble transition occur. Out of 2^{16}, 2^9 key-byte candidates

Table 1: Comparison of Differential Probabilities

BlockSize	Rounds (r-1)	Active Sboxes	$\mathcal{P}_D^{\dagger}$	$\mathcal{Q}_D^{*}$
64	14	28	2^{-168}	2^{-91}
128	17	34	2^{-204}	2^{-112}
256	20	40	2^{-240}	2^{-133}

† Probability of the classical differential trail claimed by the designers

∗ Probability of the iterated truncated differential trail

remain. The same filtering technique is used for second and third round to filter out more wrong key bytes. After using 3 rounds of filtering, it is expected only one candidate should remain for the byte pair $\left(2^{16} \times (2^{-7})^3 = 2^{-5} < 1\right)$. For the remaining symmetric key bytes, the procedure is repeated for 7 more times. At the end, it is expected that only one key candidate should pass the test. The other subkeys can be recovered in same way (After recovering the first subkey, the value of plaintexts are exactly known till second subkey whitening). Same key recovery attacks are applicable for Flex-64 and Flex-256.

4 Yoyo Game

A deterministic distinguisher for two generic Substitution-Permutation (SP) rounds have been reported by Rønjom et al [RBH17]. This has been used to devise a 6-round Flex-128 distinguisher and a 7-round Flex-128 key recovery attack. To apply their results, first **Zero Difference Pattern** and **Swapping of Words** needs to be defined here originally defined in [RBH17].

Let, $F : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be a permutation with $q = 2^k$ and

$$F(x) = S \circ L \circ S \circ L \circ S(x)$$

Here, S is the concatenation of several smaller SBoxes operating on elements from $\mathbb{F}_q$ in parallel and L is the linear layer over $\mathbb{F}_q^n$. A *state* is defined as the vector of words $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_{n-1}) \in \mathbb{F}_q^n$.

Definition 1. Zero Difference Pattern.[RBH17] Let, $\alpha \in \mathbb{F}_q^n$ for $q = 2^k$. The Zero Difference Pattern for α is

$$\nu(\alpha) = (z_0, z_1, ..., z_{n-1}),$$

where $\nu(\alpha)$ takes values in $\mathbb{F}_2^n$ and $z_i = 1$ if $\alpha_i = 0$ or $z_i = 0$ otherwise.

Definition 2. Swapping of Words.[RBH17] Let, $\alpha, \beta \in \mathbb{F}_q^n$ be two states and $v \in \mathbb{F}_2^n$ be a vector, then $\rho^v(\alpha, \beta)$ is a new state in $\mathbb{F}_q^n$ created from α, β by swapping components among them. The i^{th} component of $\rho^v(\alpha, \beta)$ is defined as

$$\rho^v(\alpha, \beta)_i = \begin{cases} \alpha_i, & \text{if } v_i = 1; \\ \beta_i, & \text{if } v_i = 0. \end{cases} \tag{1}$$

The following theorem describes the deterministic distinguisher for 2 generic SP-rounds.

Theorem 1. *[RBH17] Let, $p^0, p^1 \in \mathbb{F}_q^n$, $c^0 = G_2(p^0)$ and $c^1 = G_2(p^1)$. For any vector $v \in \mathbb{F}_2^n$, $c'^0 = \rho^v(c^0, c^1)$ and $c'^1 = \rho^v(c^1, c^0)$. Then*

$$\nu(G_2^{-1}(c'^0) \oplus G_2^{-1}(c'^1)) = \nu(p'^0 \oplus p'^1) = \nu(p^0 \oplus p^1).$$

5 Attacks on FLEX-128

The results of [RBH17] have been used to devise a 6-round deterministic distinguisher. Then the 6-round deterministic distinguisher is extended using classical differential to perform key recovery attack on 7-round FLEX-128. Similar kinds of attacks are applicable for FLEX-64 and FLEX-256 for different number of rounds.

5.1 Deterministic Distinguisher for 6-round FLEX-128

In devising this distinguisher, theorem 1 have been used directly. For this purpose, the $S \circ L \circ S$ layers need to be identified in this construction. The S here corresponds to Super-Sbox described in 2 whereas the L corresponds to the `BlockShuffle` layer. A pair of plaintexs is chosen such that only one of the Super-Sbox is active at X_1. Yoyo game is played using these two plaintexts to obtain new pair of texts. The same Super-Sbox should be active in the new pair of texts and the other should be inactive. For a uniform random discrete distribution, this occurs with probability $\frac{1}{2^{64}}$.

Attack Procedure

1. Choose two 128-bit plaintexts p_1, p_2 such that, $wt(\nu(p_1 \oplus p_2)) = 1$. Inverse `BlockShuffle` is applied to p_1, p_2 and then they are queried to encryption oracle to obtain c_1, c_2.

2. As there is two Super-Sboxes, so only one swapping is possible. One of the Super-Sbox is swapped between c_1 and c_2 to form $c_1^{'}, c_2^{'}$, which are queried to encryption oracle and $p_1^{'}, p_2^{'}$ is obtained.

3. Check whether $wt(\nu(BS(p_1^{'}) \oplus BS(p_2^{'}))) = 1$ or not. If it is 1, then distinguish it as FLEX-128; otherwise it is a random permutation.

Complexity Evaluation The attack needs 2 encryption queries and 2 decryption queries; its time complexity is 2 `BlockShuffle` , 2 inverse `BlockShuffle` operation and 2 FLEX-128 state XOR, and the memory complexity is negligible.

5.2 Key Recovery for 7-round FLEX-128

For attacking 7-round FLEX-128, yoyo distinguishing attack on 6-round FLEX-128 is composed with the one round trail of iterated truncated differential 3.2. The attack is shown in fig 4. With probabilty 2^{-7} only one Super-Sbox is active at X_2. By virtue of yoyo game, only one Super-Sbox should be active in W_2. Due to inverse block shuffle, the differences should be confined to the upper nibbles only in Z_1; the lower nibbles should be free. With probability 2^{-8}, two symmetric bytes becomes free at Z_1. There are 8 choices for symmetric byte positions which increases the probability to 2^{-5}. Therefore, at the cost of 2^{-12}, two symmetric bytes become free for the 7-round FLEX-128.

Attack Procedure

1. Choose 2^6 plaintexts such that they differ only in B_0 and B_8. Apply inverse `BlockShuffle` on them and query them to encryption oracle to obtain corresponding ciphertexts. Consider all ciphertext pairs, swap bytes between them according to the Super-Sbox output and query them to the decryption oracle to obtain new pairs of plaintexts. Check whether the pair has a pair of free symmetric bytes. At least one such pair is expected.

2. Repeat step 1 two more times to obtain two right pairs. Let, c_1, c_2 and c_3, c_4 be such pairs and their corresponding plaintexts are p_1, p_2, p_3, p_4. After byte swapping c_1, c_2

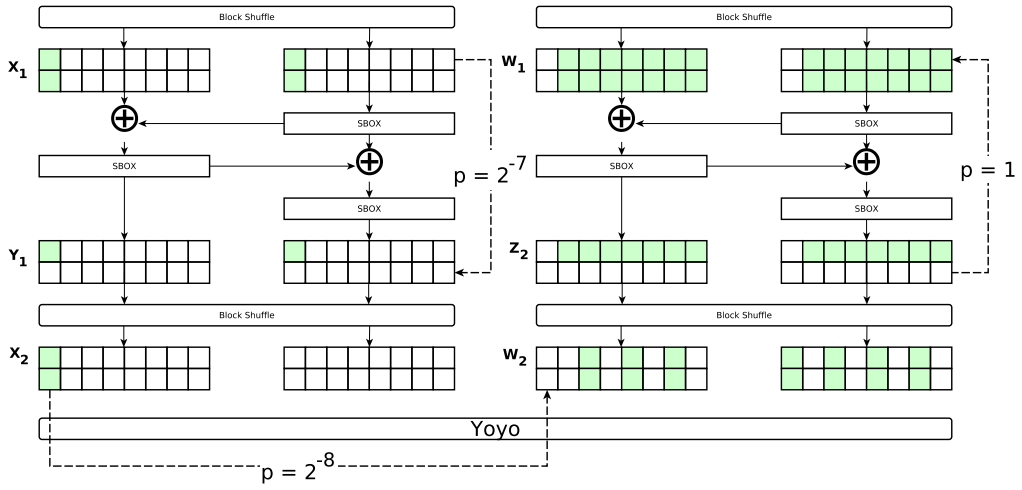

Figure 4: 7-round Yoyo Distinguisher for FLEX-128

and c_3, c_4 becomes $c_1^{'}, c_2^{'}$ and $c_3^{'}, c_4^{'}$. `BlockShuffle` is applied on the decrypted value of these modified ciphertexts to obtain $p_1^{'}, p_2^{'}, p_3^{'}, p_4^{'}$.

3. Guess key bytes 0 and 8 for K_0, run one round encryption for $p_1^{'}, p_2^{'}$ and observe whether same nibble in B_0 and B_8 remains free or not for the pair. Using the nibble transition, out of 2^{16} candidates, 2^7 are filtered out. Then $p_3^{'}, p_4^{'}$ is used for the remaining 2^9 candidates and the number of key candidates for B_0 and B_8 is reduced to 2^2.

4. For the remaining 7 symmetric pairs of bytes step 3 is repeated 7 more times. At, the end $2^{2\times 8} = 2^{16}$ key candidates are expected for K_0. Using K_0, K_1 is recovered. If there are more than one right key candidate, then they are exhaustively tried for finding the right key candidate.

Complexity Evaluation The attack needs $(2 \times 2^6 + 2^{16})$ encryption queries and 2×2^{12} decryption queries; its time complexity is 2^7 inverse `BlockShuffle` operation, 2^{13} `BlockShuffle` operation, 2^7 memory accesses for retrieving the plaintexts, 2^7 memory accesses for storing the ciphertexts and 2^{13} memory accesses for retrieving the ciphertexts, and the memory complexity is 2^7 FLEX-128 states for storing the plaintexts and ciphertexts.

References

[EMdN19] JosÃľ AntÃ±nio Moreira XexÃľo Eduardo Marsola do Nascimento. FlexAEAD -A Lightweight Cipher with Integrated Authentication. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/FlexAEAD-spec.pdf, 2019.

[RBH17] Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 217–243, Cham, 2017. Springer International Publishing.