Cryptanalysis of 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 paper, the internal *keyed* permutation of FlexAEAD is analyzed. In our analysis, we report an iterated truncated differential for one round which holds with a probability of 2^{-7} and can penetrate the same number of rounds as claimed by the designers with much less complexity and can be easily converted to a key-recovery attack. We further report a Super-Sbox construction in the internal permutation, which is 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 the 64-bit and 256-bit variants. Success probabilities of all the reported distinguishing attacks are shown to be high. All practical attacks are experimentally verified.

Keywords: Distinguisher, FlexAEAD, Iterated Differential, Key Recovery, NIST lightweight cryptography competition, Yoyo

1 Introduction

In modern era, the aim is to connect each of the physical devices, even the miniature ones, with internet so that they can be monitored and controlled remotely for maximum utilization. These devices are powered with the ability of communicating among themselves. Such a huge interconnected system, consisting of numerous tiny devices, is not free from vulnerabilities. Moreover, security breach in such systems can be catastrophic. So, a major concern in the world of *internet-of-things* is how to provide security and privacy to each system with the constraints of limited power and area. NIST LightWeight Cryptography (LWC) competition [NIS] is major step towards addressing these issues. There are total of 57 submissions in this competition. Apart from authenticated encryption algorithms in lightweight environment, some of the designs also comprise of hash functions. Some of them have also provided new primitives for block cipher design.

FlexAEAD is one of the round 1 candidates proposed by Nascimento and Xexo [dNX19]. For authenticated encryption it has three variants-

1. FlexAEAD 128b064 128-bit key, 64-bit block, 64-bit nonce and 64-bit tag
2. FlexAEAD 128b128 128-bit key, 128-bit block, 128-bit nonce and 128-bit tag
3. FlexAEAD 256b256 256-bit key, 256-bit block, 256-bit nonce and 256-bit tag

It has its own primitive; internal *keyed* permutation (PF_k) of 64-bit, 128-bit and 256-bit. We have analyzed the PF_k function and reported several results. The internal *keyed* permutation of FlexAEAD with x-bit state is refer to as Flex-x.

1.1 Existing Security Claims

The designers have claimed that mounting an attack on Flex-x based on differential and linear characteristics is more difficult than the brute force attack. According to their analysis, the probability of best differential characteristic for Flex-64, Flex-128 and Flex-256 is 2^{-168}, 2^{-204} and 2^{240} respectively. The number of chosen plaintext pairs required for a linear trail in Flex-64, Flex-128 and Flex-256 are 2^{272}, 2^{326} and 2^{380} respectively [dNX19]. Eichlseder et al. have claimed several forgery attacks on FlexAEAD with complexities less than those given by the designers. For forging attacks they have followed several different approaches; like changing associated data, truncating ciphertexts and reordering ciphertexts. They have reported differential characteristics for 5-round Flex-64, 6-round Flex-128 and 7-round Flex-256 with probability 2^{-66}, 2^{-79} and 2^{-108} respectively. They have also reported clustered characteristics for all variants of PF_k [EKS19a,EKS19b]. Length extension attack based on associated data have also been shown [M'e19].

1.2 Our Contributions

First of all, we report an iterated truncated differential for all the variants of PF_k using the property of AES Difference Distribution Table (DDT) where the output difference of a byte is confined to either upper or lower nibble. These differentials are further exploited to recover the subkeys.

Next, a deterministic Yoyo distinguisher of 4, 6 and 8 rounds for Flex-64, Flex-128 and Flex-256 respectively are devised. All these distinguishers are used for mounting key recovery attacks for one more extra round. The key recovery attacks with their computed complexities are summarized in table 1. For the iterated truncated differential, the maximum number of rounds that is penetrable for a Flex-x variant are enlisted in the table.

Outline The necessary details about internal *keyed* permutation of FlexAEAD can be found in [dNX19]. Rest of the paper is organized as follows. Section 2

Table 1: Comparison of Key Recovery Attacks. Encs, Decs, MAs refers to encryption queries, decryption queries and Memory Accesses respectively. For uniformity, memory accesses and memory complexity has been provided in terms of Flex-128 state. 1 MA for Flex-128 corresponds to 2 MA in Flex-64 and 0.5 MA in Flex-256. Memory complexity is also normalized by the same ratio.

Block Size	**rounds**	**Data Complexity**		**Time Complexity**	**Memory Complexity**	**Attack Type**	**Section No. of Current Work**
		Encs	**Decs**	**MAs**			
64	7	$2^{30.5}$		$2^{34.5}$	$2^{18.5}$	Iterated Truncated Differential	2.4
	5	2^{10}	$2^{16.5}$	$2^{15.5}$	2^{10}	Yoyo Attack	3.3
128	16	$2^{93.5}$		$2^{108.5}$	$2^{20.5}$	Iterated Truncated Differential	2.4
	7	$2^{10.5}$	$2^{16.5}$	$2^{16.5}$	$2^{11.5}$	Yoyo Attack	3.3
256	21	$2^{109.5}$		$2^{125.5}$	$2^{22.5}$	Iterated Truncated Differential	2.4
	9	2^{11}	$2^{16.5}$	$2^{17.5}$	2^{13}	Yoyo Attack	3.3

describes the *key*-recovery attacks based on Iterated Truncated Differential. Section 3 details the attacks based on Yoyo game. The success probabilities of distinguishing attacks and their experimental verification are illustrated in Section 4. Finally, the concluding remarks are furnished.

2 Iterated Truncated Differential Attacks on PF_k

Differential of iterative characteristics can be easily exploited to penetrate full rounds of a cipher. The fundamental strategy behind devising an iterated differential is to choose the output differential in a way such that after some operations the input differential can be produced easily. Recently, a deterministic iterated differential has been reported for SNEIK permutation [Per19]. Earlier, Alkhzaimi et al. have reported such differentials for SIMON family of block ciphers [AL13]. In this work, iterated differential in truncated form have been considered. First of all, a particular property of AES Sbox which have been exploited needs to be discussed.

2.1 Property of AES DDT Table

From AES DDT table it has been observed that the number of randomly chosen input differences that map to output differences, such that the non-zero bits in each output difference are confined to the upper nibble is 4096. Same is true if they are confined to the lower nibble. In other words,

$$\left|\left\{(x_1,x_2)\middle|\big(S(x_1)\oplus S(x_2)\big)\ \&\ \texttt{0xf0}\ =0,\forall x_1,x_2\in\mathbb{F}_{2^8}\right.\right|=4096$$
$$\left|\left\{(x_1,x_2)\middle|\big(S(x_1)\oplus S(x_2)\big)\ \&\ \texttt{0x0f}\ =0,\forall x_1,x_2\in\mathbb{F}_{2^8}\right.\right|=4096$$

S is the AES Sbox. Therefore, with probability $\frac{4096}{2^{16}}=2^{-4}$ a random input difference transits to upper nibble in the output difference. With same probability, random input difference transits to lower nibble.

2.2 One Round Probabilistic Iterated Truncated Differential

Refer to Fig. 1 for the iterated differential of Flex-128. 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 in X_2. In X_2 the active byte can be either $B[0]$ or $B[1]$, depending on whether the upper or lower nibbles in Y_1 are active. Similar kinds of iterated truncated differential with same probability exists for Flex-64 and Flex-256.

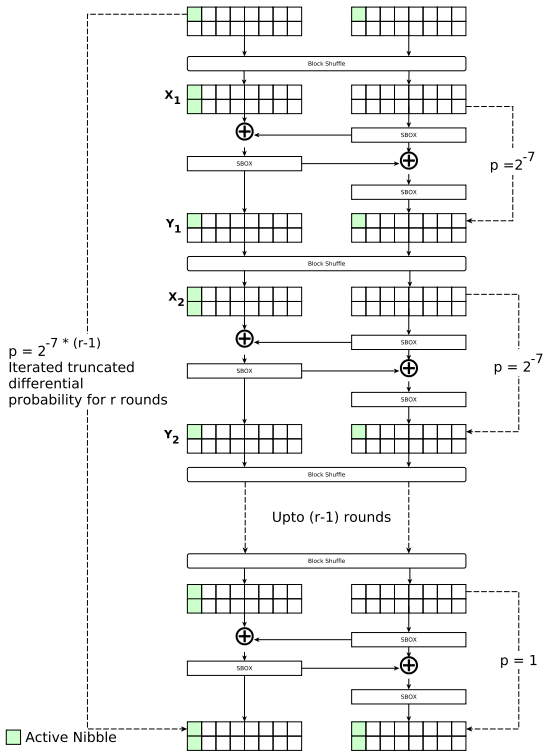

Fig. 1: Iterated Truncated Differential with One-round probability of 2^{-7}. Note that the key-addition is not shown, since it has no effect on the trail

2.3 Application to Variants of FlexAEAD Permutation

The one round iterated truncated differential can be applied to all the versions of internal *keyed* permutation PF_k. The iterated differential occurs with probability 2^{-7}. Depending on the block size, last few rounds can be made free as no byte to nibble transition is needed for those rounds.
Let, the iterated truncated differential is kept free for last f rounds for Flex-x. Then the probability of the trail is $2^{-7\times(r-f)}$. For uniform random discrete distribution, the same event will occur with probability $2^{-8\times(\frac{x}{8}-2^f)} = 2^{-(x-8*2^f)}$. For devising a distinguisher for x-bit flex,

$$2^{-7\times(r-f)} > 2^{-(x-8*2^f)}$$

$$\implies r < \frac{(x-8*2^f)}{7} + f \tag{1}$$

Then, probability of the iterated truncated differential trail for r-round Flex-x is $2^{-7\times(r-f)}$. Table 2 shows the trail probabilities for different Flex-x. r_{max} denotes the maximum number of rounds reachable under the constraints of fixed f. Table 3 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.

Table 2: Iterated Differential Trails

Block Size	f	r_{max}	**Trail Probability**
64	1	7	2^{-42}
	2	6	2^{-28}
128	1	16	2^{-105}
	2	15	2^{-91}
	3	12	2^{-63}
256	1	21	2^{-140}
	2	21	2^{-123}
	3	21	2^{-126}
	4	21	2^{-119}

Another aspect of such kind of trails is the position of active byte in each round. As mentioned in 2.2, if $B[0]$ is active in X_0, then either $B[0]$ or $B[1]$ is active in X_2. If $B[1]$ is active in X_2, then either $B[2]$ or $B[3]$ is active in X_3. In general, for Flex-x if $B[m]$ or $B[\frac{x}{2\times 8}+m]$ is active in X_i, then either $B[2m]$ or $B[2m+1]$ is active in $X_{(i+1)}$.

2.4 Key Recovery Using Iterated Truncated Differential

At the end of each round, difference in a pair of symmetric bytes after S-box transits to same nibble with probability 2^{-7}. Symmetric bytes refer to the byte

Table 3: Comparison of Differential Probabilities

Block Size	**Rounds** r	**Active S-boxes**	$\mathcal{P}^{\dagger}_{\mathbf{D}}$	$\mathcal{Q}^{*}_{\mathbf{D}}$
64	15	28	2^{-168}	2^{-98}
128	18	34	2^{-204}	2^{-119}
256	21	40	2^{-240}	2^{-119}

† Probability of the classical differential trail claimed by the designers

∗ Probability of the iterated truncated differential trail

pair in identical positions in left half and right half of the state. In same way, symmetric nibbles are defined. 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 $2^{-7\times(r-f)}$, where f is number free rounds. Suppose, in the right pair the initial difference is in $B[i]$ and $B[i+8]$. So, we guess key byte $K[i]$ and $K[i+8]$. 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 remain. For further filtering, two more right pairs are used. The second right pair reduces the candidate numbers to 2^2. After filtering using three different right pairs, it is expected only one candidate should remain for the key 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.

2.5 Complexity Evaluation

Distinguisher To distinguish iterated truncated differential for r rounds, $2^{7\times(r-f)}$ number of plaintext pairs are required, where f is the number of free rounds at the end. In devising the distinguishers, difference can be kept in 2 bytes only in X_1, which yields $\binom{2^{16}}{2} \approx 2^{31}$ pairs of plaintexts. For distinguishers requiring more than 2^{31} pairs, a different set of states is needed. So, the data complexity is $\frac{2^{7\times(r-f)}}{2^{31}} \times 2^{16} = \frac{2^{7\times(r-f)}}{2^{15}}$ encryption queries. Time complexity involves the memory accesses required to compute the specified collisions, which is the number of plaintext pairs needed, i. e., $2^{7\times(r-f)}$. Memory complexity is 2^{16} Flex-x states, which is the memory required for storing different states.
Consider a particular case for 21-round Flex-256. According to inequality 1, the value of f can be set to 4. For this case

1. Data Complexity is $\frac{2^{7\times 17}}{2^{15}} = 2^{104}$ encryption queries..
2. Time Complexity is 2^{119} memory accesses.
3. Memory Complexity is 2^{16} Flex-256 states $= 2^{17}$ Flex-128 states.

Key Recovery Complexities of key recovery attack of FLEX-x depends on distinguisher. To recover each pair of key-byte, three different right pairs are required. This procedure also needs to be repeated $\frac{x}{16}$ times for recovering the full key. Therefore, data complexity, time complexity and memory complexity of distinguisher needs to be multiplied by a factor of $3 \times \frac{x}{16}$. Moreover, candidate key-byte recovery for each pair of byte can be computed in parallel. To recover the other subkey, a plaintext, ciphertext pair (p_1, c_1) is chosen and PF_k round functions till the second subkey whitening is computed offline and XOR-ed with c_1. So, the complexities of r-round FLEX-x with f free rounds are-

1. Data Complexity is $3 \times \frac{x}{16} \times \frac{2^{7\times(r-f)}}{2^{15}}$ encryption queries.
2. Time Complexity is $3 \times \frac{x}{16} \times 2^{7\times(r-f)}$ memory accesses.
3. Memory Complexity is $3 \times \frac{x}{16} \times 2^{16}$ FLEX-x states.

The complexities of particular cases for 7-round FLEX-64 with f=1, 16-round FLEX-128 with f=1 and 21-round FLEX-256 with f=4 have been listed in table 1.

2.6 Experimental Verification

The key recovery attack using iterated differentials have been experimentally verified for 8 rounds FLEX-128 with f=3. The attack initiates after a key is chosen randomly. The number of key candidates after using the first right pairs for each pair of symmetric bytes (from ($K[0]$,$K[8]$) to ($K[7]$,$K[15]$)) are 316, 520, 632, 448, 568, 484, 368 and 356 respectively. It conforms to the theoretical analysis, which states that the number of candidates should be around 2^9. After using the second right pairs, the number of candidates is reduced to 2, 12, 4, 4, 6, 5, 2 and 5 respectively which is close to the theoretical value of 2^2. The third right pair reduces the number for all pair of bytes to 1. The key recovery attack correctly recovers the subkeys.

3 Yoyo Attacks on PF_k

Details about Yoyo game can be found in [RBH17]. The result of Yoyo game on 2-generic Substitution-Permutation (SP) rounds (Theorem 2, [RBH17]) have been applied for devising r-round FLEX-x deterministic distinguisher. Then cipher specific properties have been exploited to penetrate one more extra round and recover the key. Here, r is 4, 6 and 8 for FLEX-64, FLEX-128 and FLEX-256 respectively.

3.1 Super-Sbox

Refer to Fig. 2 for the Super-Sbox construction in FLEX-128 block cipher. Consider the bytes $\{B[0], B[2], \cdots B[nb-2]\}$ 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, $B[2i]$, $B[2i+8]$ $(0 \leq i \leq 3)$

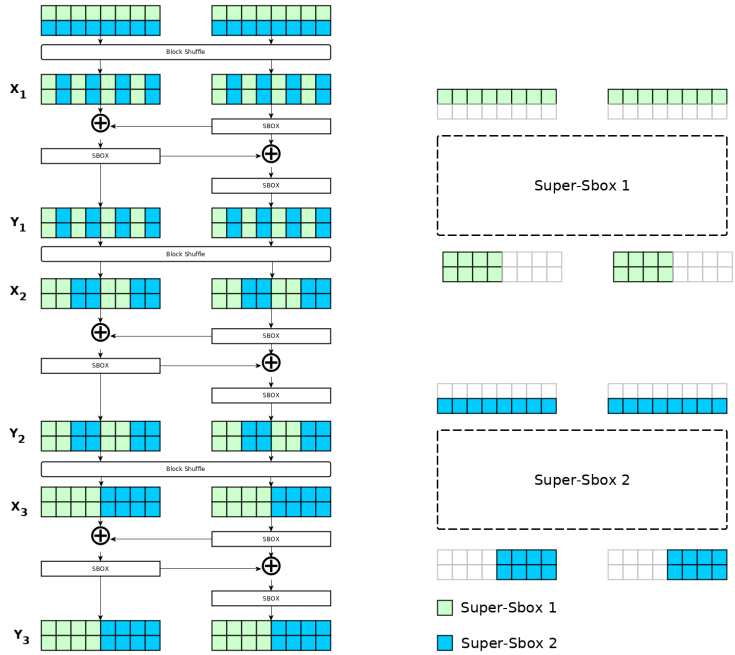

Fig. 2: Super-Sbox of FLEX-128 Block Cipher

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 64-bit Super-Sbox in the FLEX-128 state. In similar way, FLEX-64 and FLEX-256 has 32-bit and 128-bit Super-Sbox which span over 1.5 and 3.5 rounds respectively.

3.2 Deterministic Distinguisher for r-round FLEX-x

In devising this distinguisher, generalized result for 2-generic SP rounds (Theorem 2, [RBH17]) 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 3.1 whereas the L corresponds to the `BlockShuffle` layer. A pair of plaintexts 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 $\frac{1}{2^{\frac{x}{2}}}$. In attack procedure, steps pertaining to FLEX-128 has been described. Same attack strategy follows for FLEX-64 and FLEX-256.

Attack Procedure

1. Choose two 128-bit plaintexts p_1, p_2 such that, $wt^1(\nu^2(p_1 \oplus p_2)) = 1$. 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[3] 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 decryption 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.

3.3 Key Recovery for $(r+1)$-round Flex-x

For attacking $(r+1)$-round Flex-x, Yoyo distinguishing attack on r-round is composed with the one round trail of iterated truncated differential 2.2. The attack for Flex-128 is shown in Fig 3. With probability 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 `BlockShuffle` , the differences should be confined to either upper nibbles or lower nibbles in Z_1; the other half should be free. With probability 2^{-8}, two symmetric bytes becomes free at Z_1. There are 8 (4 and 16 for Flex-64 and Flex-256 respectively) choices for symmetric byte positions which increases the probability to 2^{-5} $\left(2^{-6}\right.$ and 2^{-4} for Flex-64 and Flex-256$\left.\right)$. Therefore, at the cost of 2^{-12}, two symmetric bytes become free for the 7-round Flex-128. The probability of same event for 5-round Flex-64 and 9-round Flex-256 is 2^{-13} and 2^{-11} respectively.

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 more right pairs. Let, (c_1, c_2), (c_3, c_4) and (c_5, c_6) be such pairs and their corresponding plaintexts are (p_1, p_2), (p_3, p_4) and (p_5, p_6). After byte swapping, (c_1, c_2), (c_3, c_4) and (c_5, c_6)

[1] wt calculates the number of inactive words
[2] ν is Zero Difference Pattern which denotes the position of inactive words [RBH17].
[3] For details on swapping mechanism, refer [RBH17]

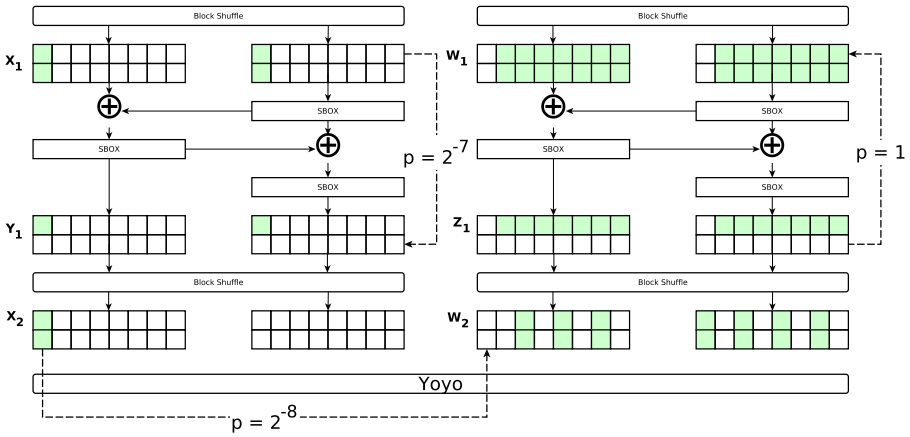

Fig. 3: 7-round Yoyo Distinguisher for Flex-128

becomes (c'_1, c'_2), (c'_3, c'_4) and (c'_5, c'_6). `BlockShuffle` is applied on the decrypted value of these modified ciphertexts to obtain (p'_1, p'_2), (p'_3, p'_4) and (p'_5, p'_6).

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 nibble transition, out of 2^{16} candidates, 2^7 are filtered out. Then the remaining two right pairs subsequently reduces the number of candidates for $K[0]$ and $K[8]$ to 2^2 and 1 respectively.
4. For the remaining 7 symmetric pairs of bytes, step 3 is repeated 7 more times. At, the end 1 key candidates are expected for K_0. For each K_0, K_1 is computed by using a plaintext-ciphertext pair. If there are more than one K_0, K_1 pair, they are exhaustively tried for finding the right key candidate.

Complexity Evaluation Let, probability of the event that "two symmetric bytes become free" is 2^{-p}. So, for retrieving a right pair, $2^{\frac{p}{2}}$ encryption queries and 2^p decryption queries are required. For guessing each pair of key byte, 3 such right pairs are needed and to recover the key, this process need to be repeated $\frac{x}{16}$ times. Therefore, data complexity of the attack is $\frac{3\times x}{16} \times 2^{\frac{p}{2}}$ encryption queries and $\frac{3\times x}{16} \times 2^p$ decryption queries.
Time complexity is $\frac{3\times x}{16} \times 2^p$ memory accesses for retrieving the stored ciphertexts.
Memory complexity is $\frac{3\times x}{16} \times 2^{\frac{p}{2}+1}$ Flex-x states for storing the plaintexts and ciphertexts.
The complexities of 7-round Flex-128 key recovery attack are-

1. Data Complexity is $24 \times 2^6 \approx 2^{10.5}$ encryption queries and $24 \times 2^{12} \approx 2^{16.5}$ decryption queries.
2. Time Complexity is $2^{16.5}$ memory accesses.
3. Memory Complexity is $2^{11.5}$ Flex-128 states.

Experimental Verification The Yoyo attack for 7-round FLEX-128 has been experimentally verified. Initially the oracle chooses a master key randomly and computes the subkeys. Adversarial algorithm queries according to 3.3 and retrieves right pairs. The number of key candidates corresponding to each symmetric bytes $\big($from $(K[0],K[8])$ to $(K[7],K[15])$ $\big)$ after filtering with first right pairs are 502, 618, 546, 496, 510, 486, 552 and 538 respectively. The second right pairs further reduces it to 6, 7 6, 7, 7, 3, 3 and 5. The third pairs reduces all these values to 1. These reduction in the number of key candidates using right pairs conforms to the theoretical analysis. At last, the algorithm successfully recovers the subkeys.

4 Success Probability

To deduce the theoretical estimation of success probability, the following theorem from [PR18] has been applied.

Theorem 1. *[PR18] Suppose, the event e happens in uniform random bitstream with probability p and in keystream of a stream cipher with probability $p(1+q)$. Then the data complexity of the distinguisher with false positive and false negative rates α and β is given by*

$$n > \frac{\left(\kappa_1\sqrt{1-p} + \kappa_2\sqrt{\big(1+q\big)\big(1-p(1+q)\big)}\right)^2}{pq^2} \tag{2}$$

where $\Phi(-\kappa_1) = \alpha$ and $\Phi(\kappa_2) = 1-\beta$.

For computing success probability, we consider $\kappa_1 = \kappa_2$ in theorem 1, which gives us $\alpha = \beta$. Then the success probability is given by $(1-\beta)$. Table 4 lists success probabilities of different distinguishers presented in this paper. The success probabilities of distinguishers with practical complexities have been experimentally verified.

Table 4: Success Probabilities of Various Distinguishers

Distinguisher Type	**Block Size**	f	**Rounds**	$p \times (1+q)$	p	**Success Prob**
Iterated	64	1	7	2^{-42}	2^{-48}	0.8
	128	1	16	2^{-105}	2^{-112}	0.82
	256	4	21	2^{-119}	2^{-192}	0.84
Yoyo	64	n/a	5	2^{-13}	2^{-14}	0.61
	128	n/a	7	2^{-12}	2^{-13}	0.61
	256	n/a	9	2^{-11}	2^{-12}	0.61

5 Conclusion

In this work, we have shown several key recovery attacks on internal permutation of FlexAEAD , which are applicable across all variants. All these attacks are based on either iterated truncated differential or Yoyo game. The iterated truncated differential based attacks on round-reduced versions and Yoyo attacks have been experimentally verified (the code of all practical attacks are available online[4]). Although, the attacks immediately do not pose a threat to authenticated encryption modes; but forgery or key recovery based on this results might exists.

References

[AL13] Hoda A. Alkhzaimi and Martin M. Lauridsen. Cryptanalysis of the SIMON Family of Block Ciphers. Cryptology ePrint Archive, Report 2013/543, 2013. https://eprint.iacr.org/2013/543.

[dNX19] Eduardo Marsola do Nascimento and José Antônio Moreira Xexéo. 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.

[EKS19a] Maria Eichlseder, Daniel Kales, and Markus Schofnegger. Forgery Attacks on FlexAE and FlexAEAD. Cryptology ePrint Archive, Report 2019/679, 2019. https://eprint.iacr.org/2019/679.

[EKS19b] Maria Eichlseder, Daniel Kales, and Markus Schofnegger. OFFICIAL COMMENT: FlexAEAD. Posting on the NIST LWC mailing list. "https://groups.google.com/a/list.nist. gov/d/msg/lwc-forum/cRjs9x43G2I/KsBQLdDODAAJ", 2019.

[M'e19] A M'ege. OFFICIAL COMMENT: FlexAEAD. Posting on the NIST LWC mailing list. "https://groups.google.com/a/list.nist.gov/d/msg/lwc-forum/DPQVEJ5oBeU/YXW0QjfjBQAJ", 2019.

[NIS] National Institute of Standards and Technology (NIST): Lightweight cryptography standardization process (2019). "https://csrc.nist.gov/projects/lightweight-cryptography".

[Per19] Lo Perrin. Probability 1 Iterated Differential in the SNEIK Permutation. Cryptology ePrint Archive, Report 2019/374, 2019. https://eprint.iacr.org/2019/374.

[PR18] Goutam Paul and Souvik Ray. On data complexity of distinguishing attacks versus message recovery attacks on stream ciphers. *Des. Codes Cryptogr.*, 86(6):1211–1247, 2018.

[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.

[4] https://drive.google.com/open?id=1GiIIAY5AoeMwW7OAh3nSAVsmtx1h5Qix