Dear all,

We think that the FlexAEAD candidate permits forgery attacks with higher success probability than $2^{\text{tagsize}}$.

Based on our understanding of the specification, we suggest the following:

**Target variants:**
- FlexAEAD128b128 (128-bit key, block, nonce, tag)
- FlexAEAD256b256 (256-bit key, block, nonce, tag)

Consider two associated data blocks $\text{AD}_i$ and $\text{AD}_j$.

Their contribution to the checksum is $\text{PF}_K2(S_i+\text{AD}_i) + \text{PF}_K2(S_j+\text{AD}_j)$, where $S_k$ is generated from the nonce $N$ and key $K3$ as $S_k = \text{PF}_K3(\text{INC32}^k(\text{PF}_K3(N)))$ and $\text{INC32}$ increases each 32-bit chunk of its input $N' = \text{PF}_K3(N)$ by 1 (addition mod $2^{32}$), in little endian notation (i.e., $1 = 0x01,0x00,0x00,0x00$).

With probability $2^{\text{-4}}$ (resp. $2^{\text{-8}}$), these 4 (resp. 8) modular additions correspond to XORs. In the following, let $j = i+16$, with similar reasoning.

Assume we have a 6-round (resp. 7-round) differential for 128-bit (resp. 256-bit) $\text{PF}_K3$ with input difference

$\text{Delta}_\text{in} = 10000000$ $10000000$ $10000000$ $10000000$ $10000000$ (or 2x the same) and some $\text{Delta}_\text{out}$ with probability $p > 2^{\text{-124}}$ (resp. $2^{\text{-248}}$).

With prob. $2^{\text{-4}}$ (resp. $2^{\text{-8}}$), the input difference to $\text{PF}_K3$ in $\text{INC32}(N')$ between some $\text{AD}_i$ and $\text{AD}_j$ is $\text{Delta}_\text{in}$, and then, with probability $p$, the output difference in $S_i$, $S_j$ is $\text{Delta}_\text{out}$.

Query the tag for some plaintext with associated data of at least $j+1$ blocks with $\text{AD}_i + \text{AD}_j = \text{Delta}_\text{out}$.

With prob. $p*2^{\text{-4}}$ (resp. $p*2^{\text{-8}}$), $\text{AD}_i + \text{AD}_j + S_i + S_j = 0$, so $S_i + \text{AD}_i = S_j + \text{AD}_j$, so the contribution to the checksum is

$\text{PF}_K2(S_i+\text{AD}_i) + \text{PF}_K2(S_j+\text{AD}_j) = 0$.

If we swap $\text{AD}_i$ and $\text{AD}_j$, with the same reasoning, the contribution will also be 0, so the old tag is valid for the modified associated data with swapped blocks.

This forgery attack is successful with probability $p*2^{\text{-4}}$ (resp. $p*2^{\text{-8}}$).

Now we need to find a suitable differential characteristic for $(\text{Delta}_\text{in}, \text{Delta}_\text{out})$.

For FlexAEAD128b128, a suitable differential with probability $p = 2^{\text{-79}}$ is given by

$10000000$ $10000000$ $10000000$ $10000000$ $00000000$ = $\text{Delta}_\text{in}$
$00000000$ $00005000$ $00000000$ $00003000$ = $\text{Delta}_\text{out}$
The resulting forgery attack has a success probability of $2^{-83}$.

For FlexAEAD256b256, a suitable differential with probability $p = 2^{-108}$ is given by

\[
\begin{array}{cccccccc}
10000000 & 10000000 & 10000000 & 10000000 & 10000000 & 10000000 & 10000000 & 10000000 \\
00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 \\
\end{array}
= \Delta_{\text{in}}
\]

\[
\begin{array}{cccccccc}
00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 & 00000000 \\
\end{array}
= \Delta_{\text{out}}
\]

The resulting forgery attack has a success probability of $2^{-116}$.

The corresponding characteristics are illustrated in the attached PDF file. Note that the cipher is an Even-Mansour construction without round keys, so the real probability might differ.

Is this interpretation of the specification correct?

Best regards,
Maria Eichlseder, Daniel Kales, Markus Schofnegger
Dear Researchers,

First of all, thank you for the time expend on analyzing the algorithm.

Yes, your interpretation on the specification is correct.

The problem occurs because addition (when it is equivalent to XOR operation) made by the INC32 function activates only two (2) SBoxes on the first round of the PF_K3 for the FlexAEAD128b128 and only four (4) SBoxes for the FlexAEAD256b256

Several possibilities were reviewed to solve the problem like:

1) reduce the tag size (80 bits on FlexAEAD128b128 and 112 bits on FlexAEAD256b256), so the forgery attack probability will be harder than a brute force attack on the tag.

2) increase the number of rounds of the PF_K3 after the function INC32 to increase differential probability of the characteristic appointed by the researchers;

3) change the constant (0x1) added on the inc32 function to another one, so there would be more SBoxes for a certain number of associate data block;

The first option would only reflect the weakness on the tag size. The second would impact directly the cipher performance. So the third option were explored and showed fair result.

Through an exhaustive search, the value 0x11111111 was chosen to replace the 0x1 on the INC32 function (other values like 0x1FFFFFFF, 0x33333333, etc would bring the same results).

With this change, the first modular addition that activates only two (2) SBoxes on the first round for the FlexAEAD128b128 would occur only after 268435456 blocks (4 GBytes) of associate data.

In this case $p = 2^{-126}$ for:

Delta_IN = "00000011000000110000001100000011"

Delta_OUT = "000000000000000001000000000000000"

*Although the probability of this characteristic combined to the probability of the addition operation to be equal to the XOR is greater than the brute force attack to the tag, it was considered our limit for safety reasons.*
For the FlexAEAD256b256, the problem appears if the first modular addition activates only eight (8) SBoxes and after 16777216 blocks (0.5 GBytes) of associate data.

In this case \( p = 2^{-168} \) for:

\[
\text{Delta\_IN} = "0000001100000011000000110000001100000011000000110000001100000011"
\]

\[
\text{Delta\_OUT} = "0000000000000000000000000000001000000000000000000000000000000000"
\]

For the FlexAEAD128b64, the difficulty of a differential attack has been calculated to be bigger than a brute force attack to the tag.

It is also important to observe that the probability of the constant addition 0x111111111 to be equivalent to XOR is \( 2^{-32} \) for 128bit block and \( 2^{64} \) for 256bit block.

So to avoid the problem the following changes must be done on the specifications:

Page 4 - second paragraph should be now:

(…) Each chunk is treated as an unsigned number (little-endian) that is added with the constant 0x1111111111 for every block of the sequence by the function INC32. If the counter for a 64 bit block has the following bytes (x01,x02,x03,x04,xFF,x01,x02,x03), after the INC32 function, the result is (x12,x13,x14,x15,x10,x14,x13,x14).

Page 4 - forth paragraph should be now:

(…) On a multi-thread environment, the S0 can be generate adding 0xBFFFFBBB (or 0xB x 0x1111111111) to the base counter.

Also the following restrictions must be considered:

For FlexAEAD256b256 the maximum size of (associate data + message) is (2^37) bytes, the maximum size of the associate data is (2^28) bytes

For FlexAEAD128b128 the maximum size of (associate data + message) is (2^36) bytes, the maximum size of the associate data is (2^32) bytes

For FlexAEAD64b64 the maximum size of (associate data + message) is (2^35) bytes, the maximum size of the associate data is (2^35) bytes

Best Regards,

FlexAEAD Team