From: Maria Eichlseder <maria.eichlseder@iaik.tugraz.at>
Sent: Friday, April 19, 2019 1:14 PM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov; Daniel Kales; Markus Schofnegger
Subject: OFFICIAL COMMENT: FlexAEAD
Attachments: FlexAEAD-128-6.pdf; FlexAEAD-256-7.pdf

Dear all,

We think that the FlexAEAD candidate permits forgery attacks with higher success probability than 2^-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 AD_i and AD_j.
Their contribution to the checksum is PF_K2(S_i+AD_i) + PF_K2(S_j+AD_j), where S_k is generated from the nonce N and key K3 as S_k =
PF_K3(INC32^k(PF_K3(N))) and INC32 increases each 32-bit chunk of its input N' = PF_K3(N) by 1 (addition mod 2^32), in little endian notation (i.e., 1 = 0x01,0x00,0x00,0x00).
With probability 2^-4 (resp. 2^-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) PF_K3 with input difference
Delta_in = 10000000 10000000 10000000 10000000 (or 2x the same) and some Delta_out with probability p > 2^-124 (resp. 2^-248).

With prob. 2^-4 (resp. 2^-8), the input difference to PF_K3 in INC32(N') between some AD_i and AD_j is Delta_in, and then, with probability p, the output difference in S_i, S_j is Delta_out.

Query the tag for some plaintext with associated data of at least j+1 blocks with AD_i + AD_j = Delta_out.
With prob. p*2^-4 (resp. p*2^-8), AD_i + AD_j + S_i + S_j = 0, so S_i + AD_i = S_j + AD_j, so the contribution to the checksum is
PF_K2(S_i+AD_i) + PF_K2(S_j+AD_j) = 0.
If we swap AD_i and 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^-4 (resp. p*2^-8).

Now we need to find a suitable differential characteristic for (Delta_in, Delta_out).

For FlexAEAD128b128, a suitable differential with probability p = 2^-79 is given by

10000000 10000000 10000000 10000000 = Delta_in
00000000 00005000 00000000 00003000 = Delta_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

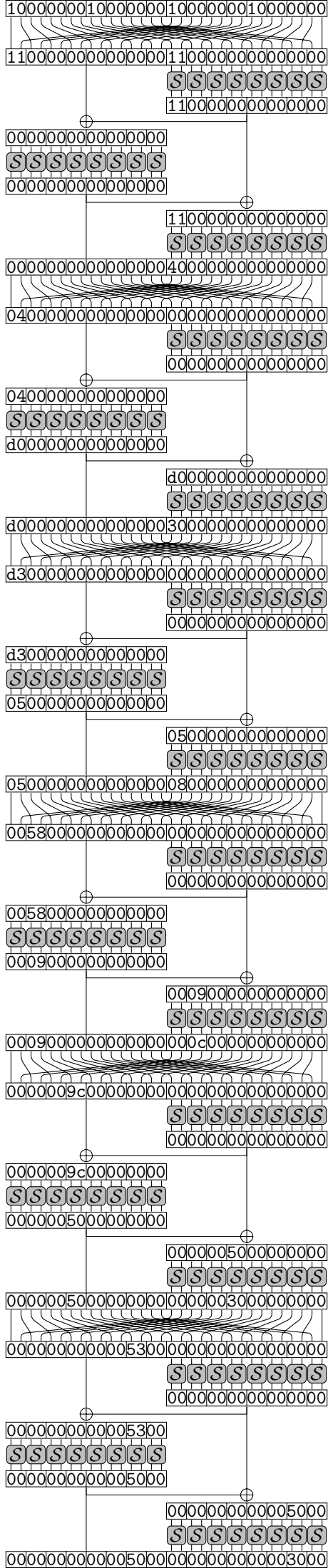
10000000 10000000 10000000 10000000 10000000 10000000 10000000 10000000 = Delta_in
00000000 00000000 00009000 00000000 00000000 00000000 00000000 00000000 = Delta_out

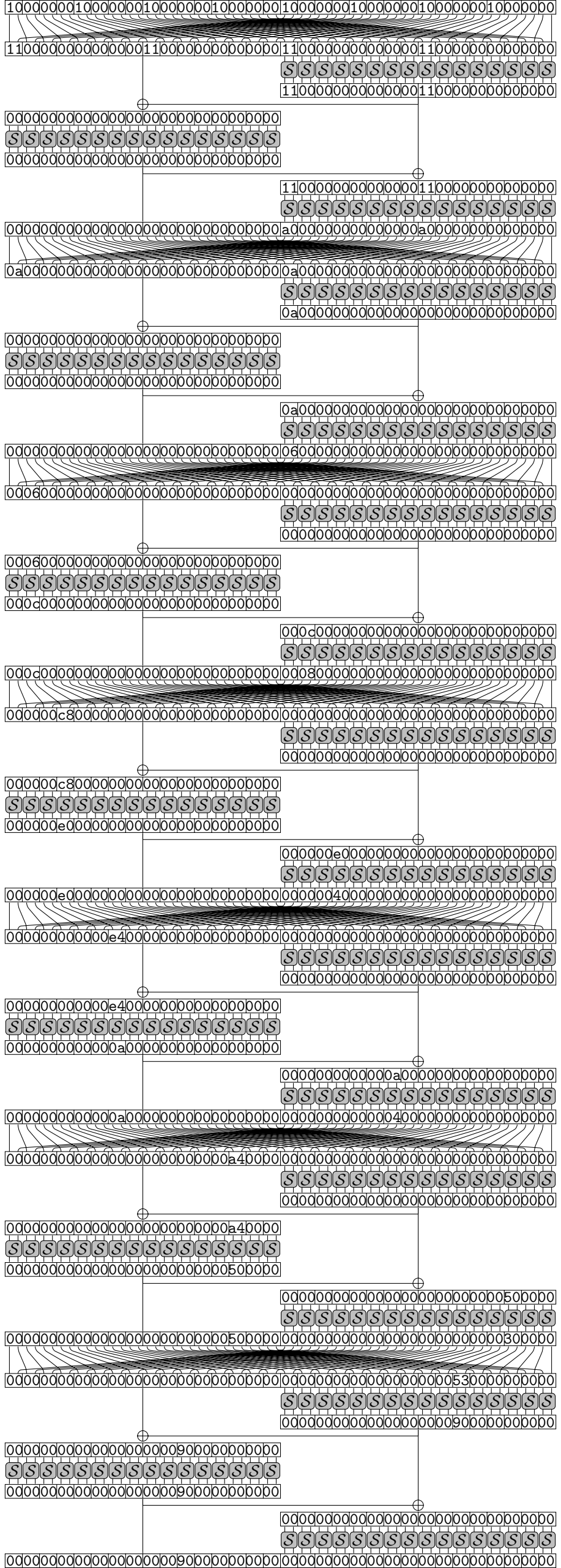
The resulting forgery attack has a success probability of 2^-116.

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


10000000 10000000 10000000 10000000
11000000000000000 11000000000000000
S S S S S S S S S
11000000000000000
0000000000000000
S S S S S S S S S
0000000000000000
11000000000000000
S S S S S S S S S
0000000000000000 4000000000000000
04000000000000000 0000000000000000
S S S S S S S S S
0000000000000000
04000000000000000
S S S S S S S S S
d0000000000000000
d0000000000000000
S S S S S S S S S
d0000000000000000 3000000000000000
d3000000000000000 0000000000000000
S S S S S S S S S
0000000000000000
d3000000000000000
S S S S S S S S S
0500000000000000
0500000000000000
S S S S S S S S S
0500000000000000 0800000000000000
0058000000000000 0000000000000000
S S S S S S S S S
0000000000000000
0058000000000000
S S S S S S S S S
0009000000000000
0009000000000000
S S S S S S S S S
0009000000000000 000c000000000000
0000009c00000000 0000000000000000
S S S S S S S S S
0000000000000000
0000009c00000000
S S S S S S S S S
0000005000000000
0000005000000000
S S S S S S S S S
0000005000000000 3000000000000000
0000000000005300 0000000000000000
S S S S S S S S S
0000000000000000
0000000000005300
S S S S S S S S S
0000000000005000
0000000000005000
S S S S S S S S S
0000000000005000 3000000000000000



From: Eduardo Nascimento <edunasci@yahoo.com>
Sent: Saturday, May 04, 2019 7:39 AM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: FlexAEAD
Attachments: 2019-05-01-FlexAEAD128b128-dif-example.png; 2019-05-01-FlexAEAD256b256-dif-example.png

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 characterisc 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 = "00000000000000010000000000000000"

*Although the probability of this characterisc 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:

Delta_IN = "00000011000000110000001100000011000000110000001100000011000000011"

Delta_OUT = "00000000000000000000000000000010000000000000000000000000000000000

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 0x11111111 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 0x11111111 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 0x11111111 to the base counter and in a parallel thread the S0 can be generate adding 0xBBBBBBBB (or 0xB x 0x11111111) to the base counter.

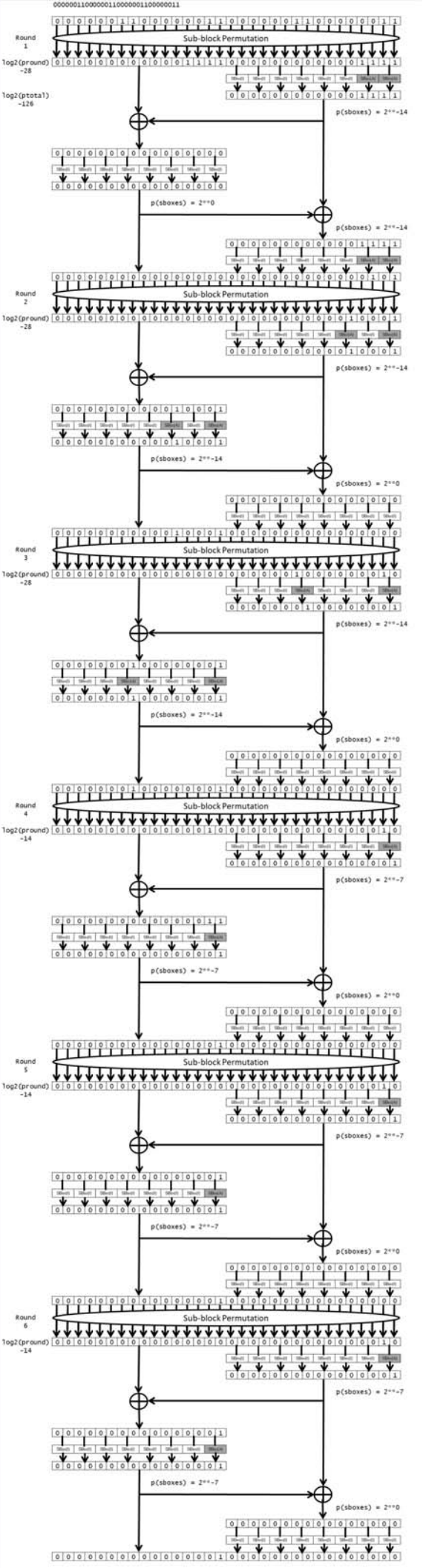
Also the following restrictions must be considered:

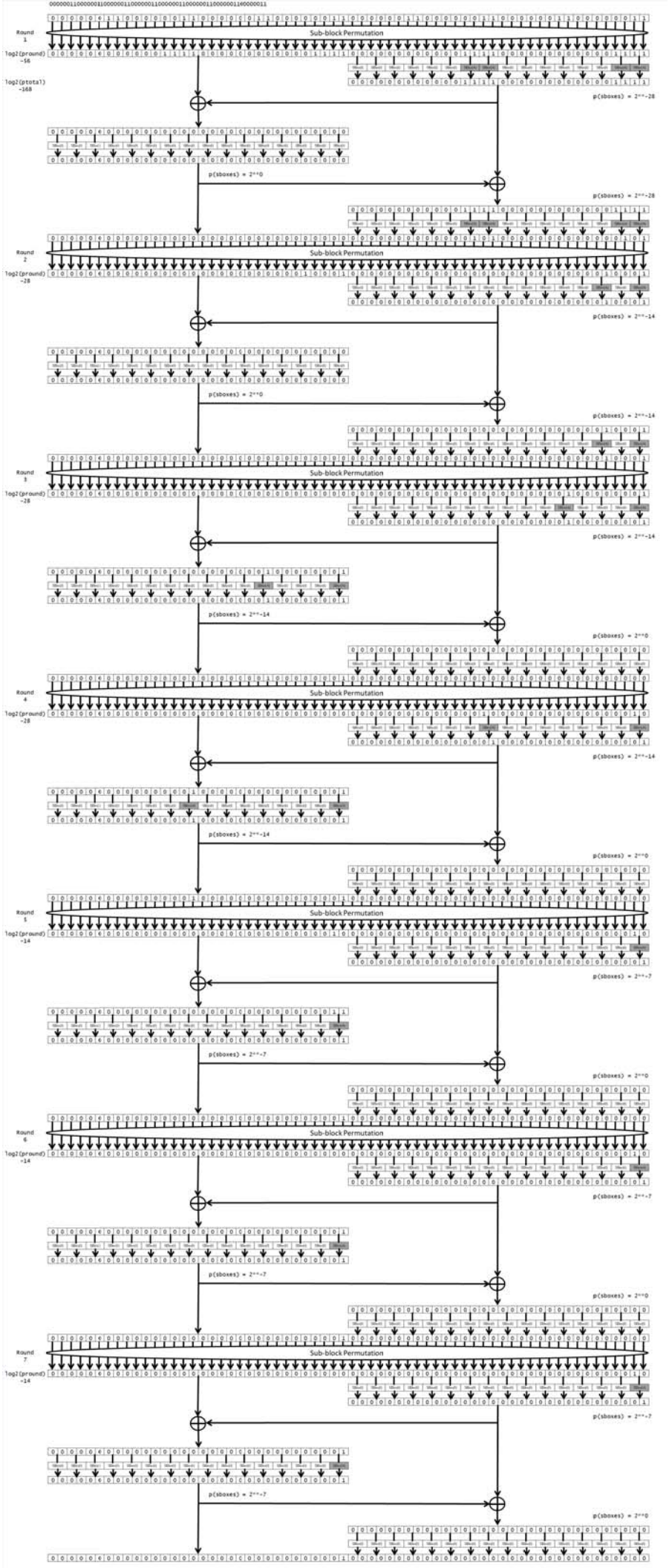
For FlexAEAD256b256 the maximum size of (associate data + message) is (2^37) bytes, the maximum size of the associate data is (2^28) bytes

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


Round 1
Sub-block Permutation
log2(pround) -28
log2(ptotal) -126
p(sboxes) = 2**-14
p(sboxes) = 2**0
Round 2
Round 3
Round 4
p(sboxes) = 2**-7
Round 5
Round 6


Round 1
Sub-block Permutation
p(sboxes) = 2**-28
p(sboxes) = 2**0
Round 2
p(sboxes) = 2**-14
Round 3
Round 4
Round 5
p(sboxes) = 2**-7
Round 6
Round 7

From: MEGE, Alexandre <alexandre.mege@airbus.com>
Sent: Monday, June 3, 2019 5:27 AM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: [lwc-forum] OFFICIAL COMMENT: FlexAEAD

Dear All,

It seems flexaead is vulnerable against length extension attack in the Associated Data.
This comes from the Associated Data padding being only padding with '0', so the same tag can be generated by adding '00' to the Associated Data.
This can be solved by using a resistant padding such as pad10*.

for flexaead 28b064v1, here is an example:
Key=0x000102030405060708090a0b0c0d0e0f, Nonce=0x000102030405060708090a0b0c0d0e0f,
Pt=0x,
Ad=0x00,
Ct=0xd052a99fd6826a4d

Key=0x000102030405060708090a0b0c0d0e0f, Nonce=0x000102030405060708090a0b0c0d0e0f,
Pt=0x,
Ad=0x0000,
Ct=0xd052a99fd6826a4d

And with non-empty PT:

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT = 000000000000
AD = 0000
CT = FEED07DFEB57CC9992C168BE746865E0

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT = 000000000000
AD = 000000
CT = FEED07DFEB57CC9992C168BE746865E0

Best regards,
Alexandre Mège

Kerman, Sara J. (Fed)

From: Eduardo Nascimento <edunasci@yahoo.com>
Sent: Tuesday, July 9, 2019 7:50 PM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: FlexAEAD

Dear Alexandre Mège,

Yes, this vulnerability exists on the original FlexAEAD. To solve the problem, a minor change need to be done.

As you suggested, the Associated Data is padded with 10...0. If the padding happen, the PFK2 function is executed twice on the last AD block. This is done to avoid a collision with non padded AD block that ends with 10...0.

After the change, here are examples showing that specific forgery is no longer possible.

With empty PT:

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT =
AD = 00
CT = 93E60FC2C0C84370

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT =
AD = 0000
CT = 0A56F8AFF7EDE6C5

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT =
AD = 0000800000000000
CT = D61B3B97CA9AC111

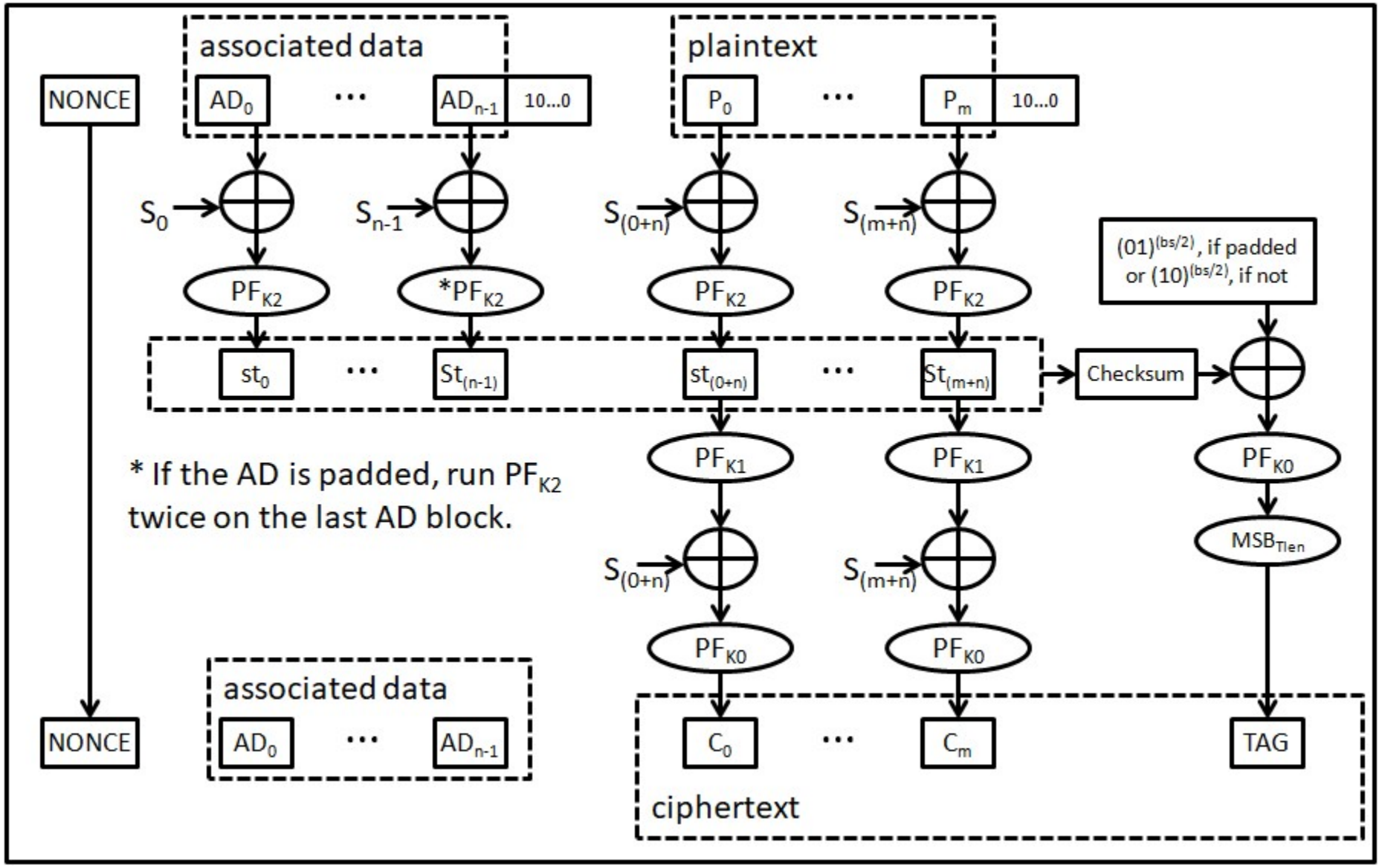
With non-empty PT:

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT = 000000000000
AD = 00
CT = 54AD9D03D2791A4156AC798B3A3F649B

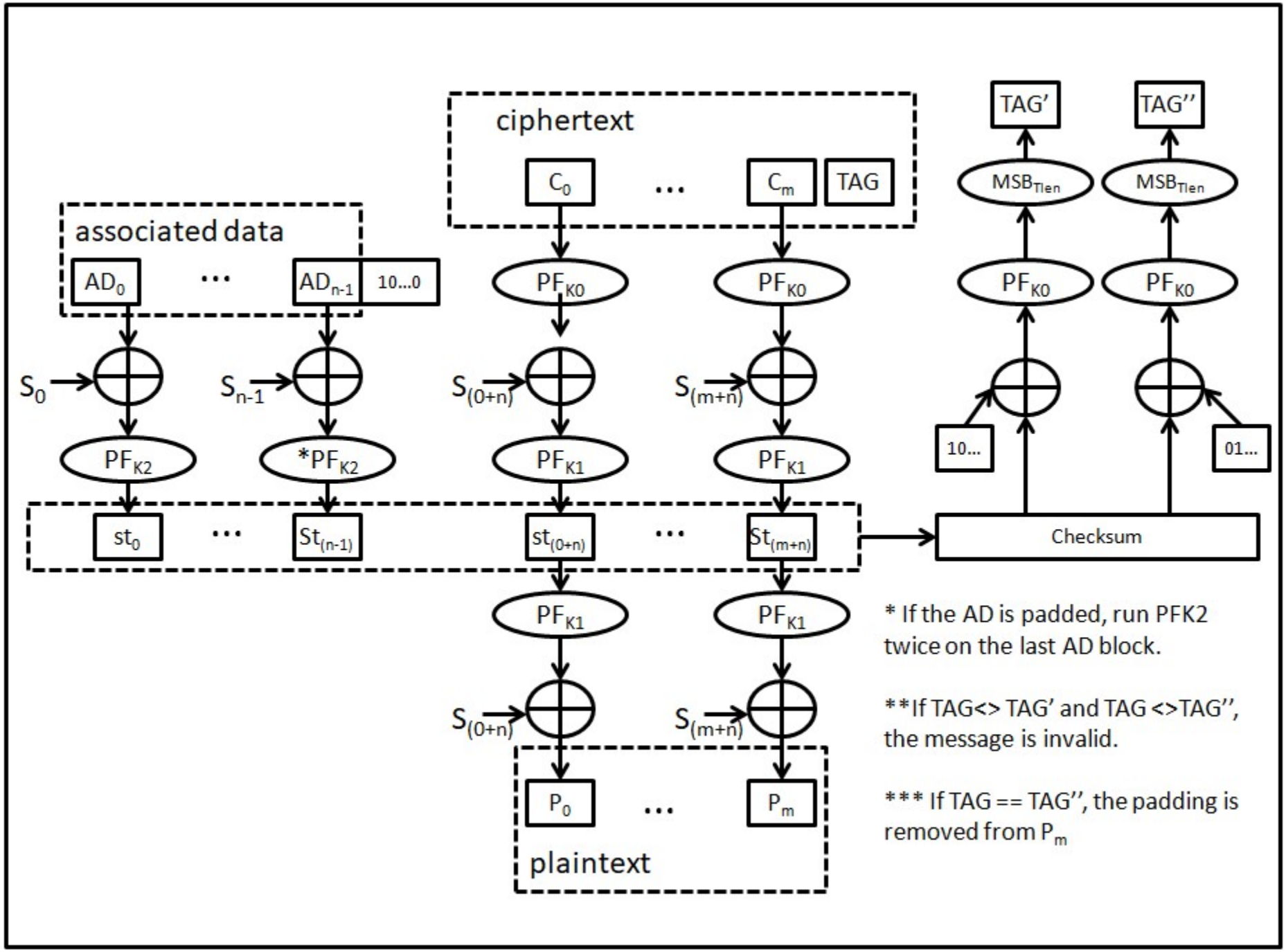
Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT = 000000000000
AD = 0000
CT = 54AD9D03D2791A417CF016CD20AF63D2

Key = 000102030405060708090A0B0C0D0E0F
Nonce = 0001020304050607
PT = 000000000000
AD = 0000800000000000
CT = 54AD9D03D2791A41BC3DC59057287649

--- modified encryption diagram:



--- modified decryption diagram:


ciphertext
C0
...
Cm
TAG
associated data
AD0
...
ADn-1
10...0
S0
Sn-1
S(0+n)
S(m+n)
PFK2
*PFK2
PFK1
PFK0
st0
St(n-1)
st(0+n)
St(m+n)
P0
Pm
plaintext
TAG'
TAG''
MSBTlen
10...
01...
Checksum
* If the AD is padded, run PFK2 twice on the last AD block.
**If TAG<> TAG' and TAG <>TAG'', the message is invalid.
*** If TAG == TAG'', the padding is removed from Pm

From: 'Eduardo Marsola Nascimento' via lwc-forum <lwc-forum@list.nist.gov>
Sent: Tuesday, July 9, 2019 8:16 PM
To: lwc-forum
Cc: xexeo@ime.eb.br; dhiman@iitbhilai.ac.in
Subject: [lwc-forum] Re: Attacks Against FlexAEAD
Attachments: Figure 2.png; Figure 3.png; Figure 1.png

Dear Researchers,

First of all congratulations for the quality of your work. Our team has analyzed deeply your document and your analysis is correct. The FlexAEAD is indeed vulnerable to the attacks you propose.

After evaluating several changes on the cipher, the solution we found to avoid the attacks is to add another linear transformation after the Block Shuffle Layer on the keyed permutation function.

This transformation divides the internal state in 8 bytes sub-blocks and make the XOR in between every 3 adjacent bytes within the sub-block: B0'=B7⊕B0⊕B1,B1'=B0⊕B1⊕B2,…,B7'=B6⊕B7⊕B0 (⊕ = XOR). The reason for dividing in 8 bytes sub-blocks is The figure 1 shows the diagram of the new PFK function.

Now in one round, if there is one different byte after the Block Shuffle, it will reflect in 3 different bytes on after Mix Adjacent Bytes. With this change, it will not be possible to create the Super-Sbox proposed on section 2 of the paper. The figure 2 shows how the bytes are mixed together in a way that prevents the creation of the proposed Super-Sbox1 and Super-Sbox2.

The proposed change also increases the cipher security against classical differential cryptanalysis attacks as well as on linear cryptanalysis attacks. After 3 rounds, all SBoxes will be active for 128 bits and bits 256 block size (2 rounds for 64 bits block size) (Figure 3).

The new difficult for differential cryptanalysis attacks had been calculate for each proposed variant. They are FlexAEAD128b064 - 2^978, FlexAEAD128b128 - 2^2448 and FlexAEAD256b256 - 2^5580.

For Linear Cryptanalysis attacks, there are FlexAEAD128b064 - 2^1052, FlexAEAD128b128 - 2^2594 and FlexAEAD256b256 - 2^5870.

On the paper there is also an iterated truncated differential attack based on the fact that the difference on byte B0 of X1 affects only bytes B0 and B8 of Y1. After the change, this assumption is no longer true causing the attack ineffective.

The algorithm amendment with the proposed changes will be submitted to NIST.

Kind Regards

FlexAEAD Team

On Wednesday, May 22, 2019 at 2:34:25 AM UTC-3, Dr. Dhiman Saha wrote:

> Dear All,

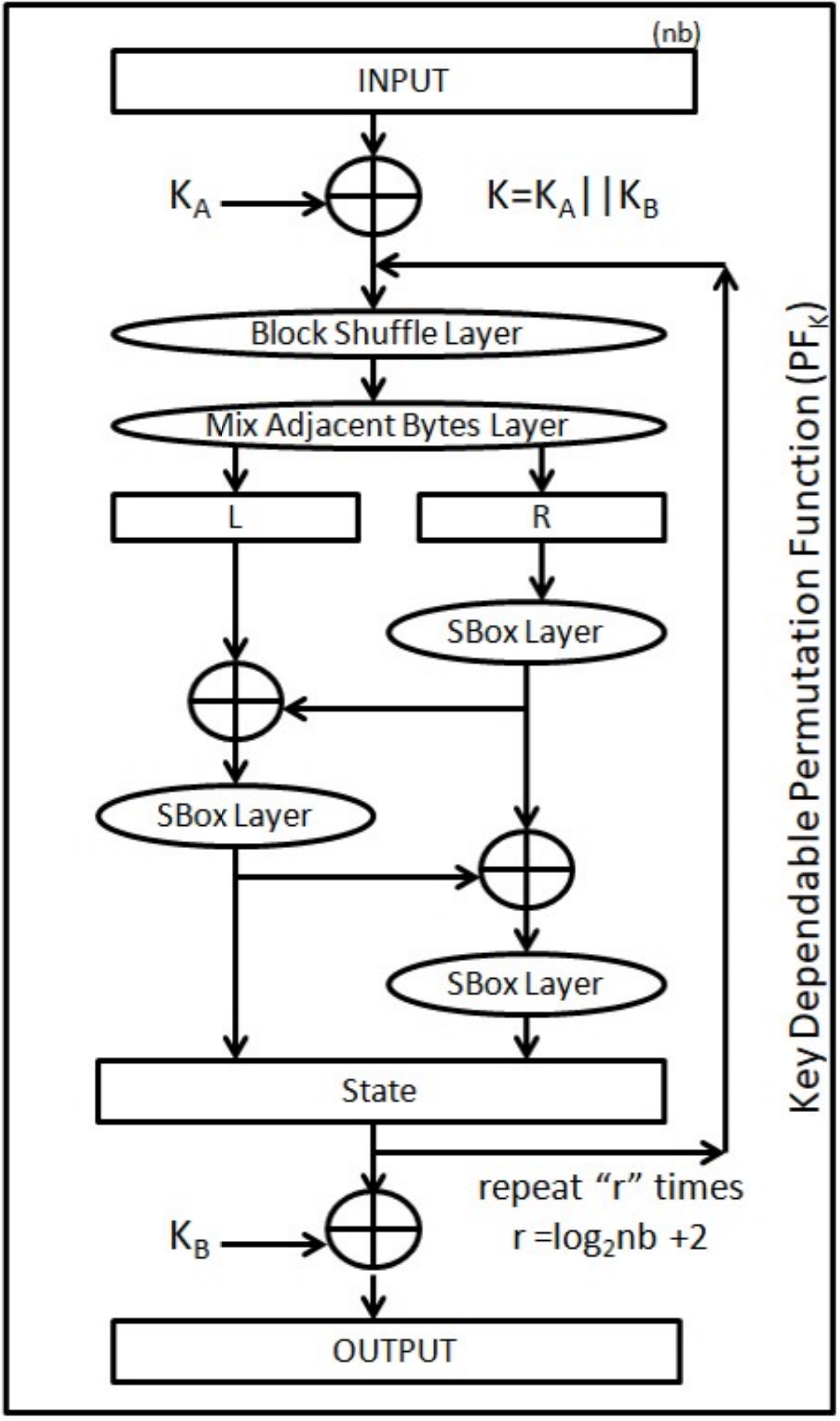


Figure 1 – New PF_K fuction with the Mix Adjacent Bytes Layer

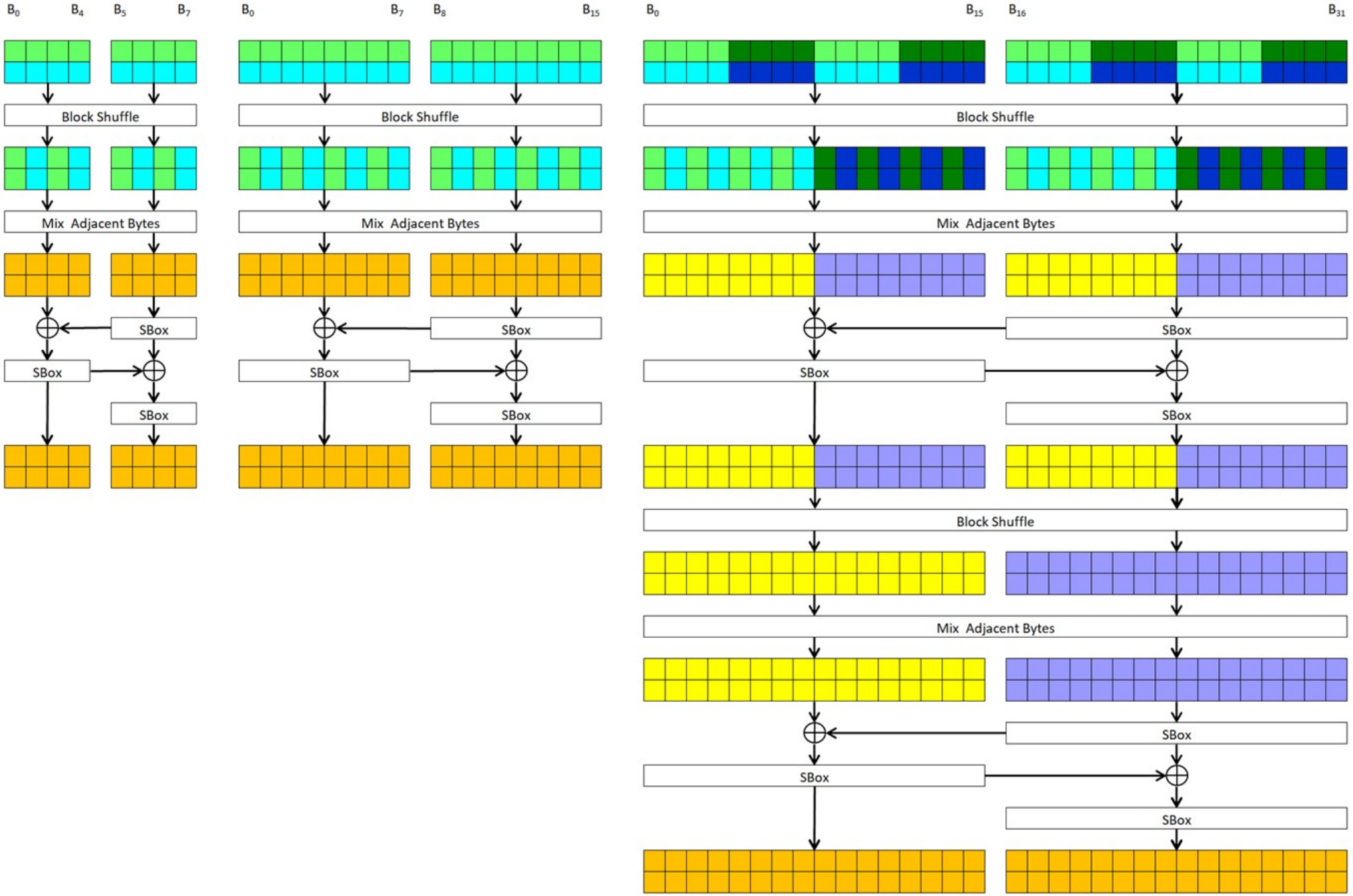


Figure 2 - how the bytes are mixed together on 64, 128 and 256 bits block size

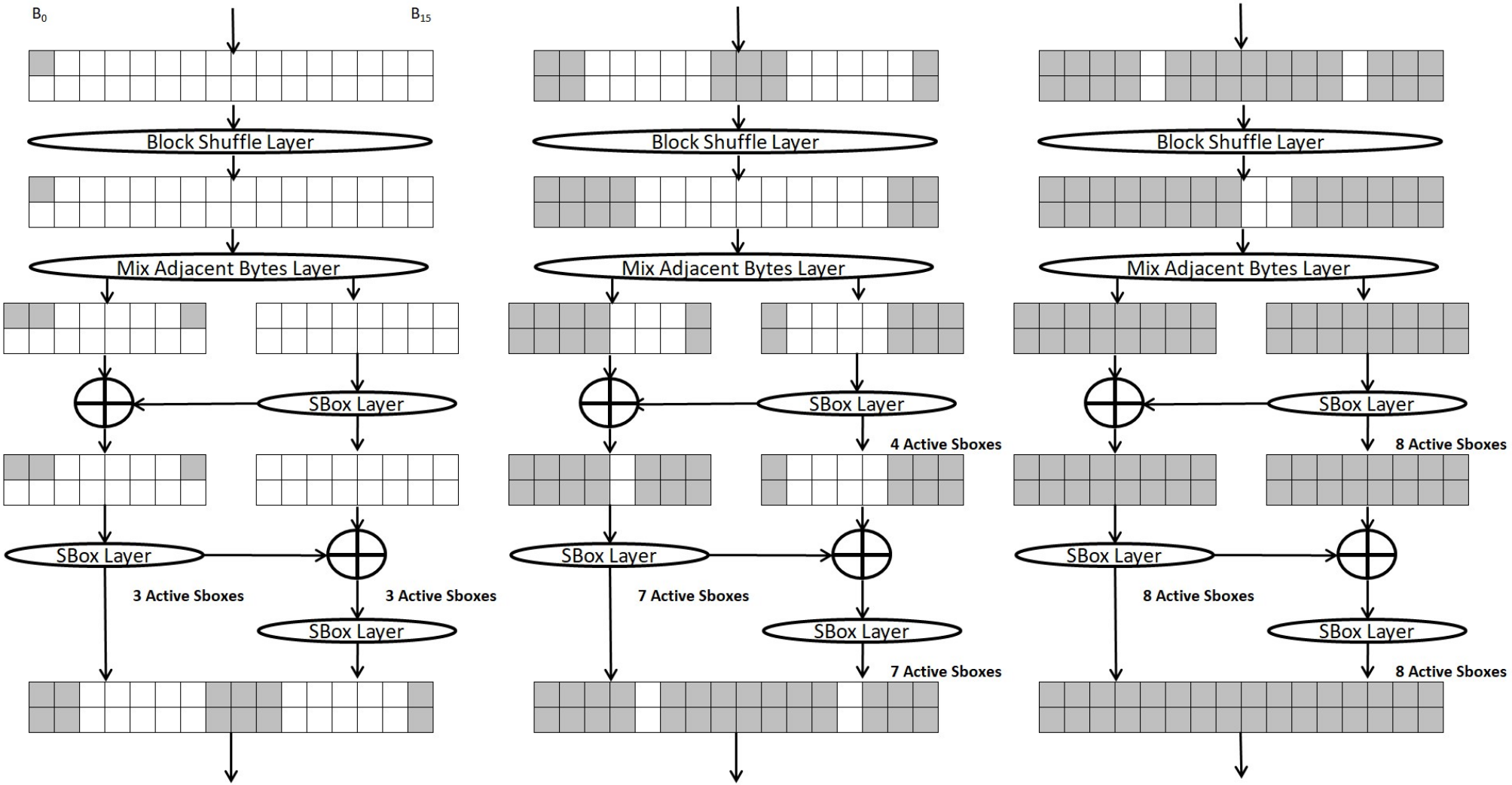


Figure 3 - for 128 bits block size, all SBox are Active after 3 Rounds

From: Maria Eichlseder <maria.eichlseder@iaik.tugraz.at>
Sent: Wednesday, July 10, 2019 8:26 AM
To: Eduardo Nascimento; lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: Re: [lwc-forum] OFFICIAL COMMENT: FlexAEAD

Dear FlexAEAD team,

Thanks for your updates.

I think the updated mode still has some potential issues:

(1) There is no sufficient separation between associated data A and message M, at least when the last block of A is not padded: if (N, A, C, T) with C = C_0 || C_rest is the encryption of some M = M_0 || M_rest and A has not been padded, then (N, A || M_0, C_rest, T) is a valid forgery.

(2) For empty A and M, the tag T is independent of the nonce N, i.e, if the tag for empty A, M for one nonce is known, the same tag is a forgery when combined with any other nonce.

For more details, see Section 4.1 of the ePrint version of our previous post on this mailing list:

https://eprint.iacr.org/2019/679

Kind regards,
Maria

On 10.07.19 01:50, 'Eduardo Nascimento' via lwc-forum wrote:

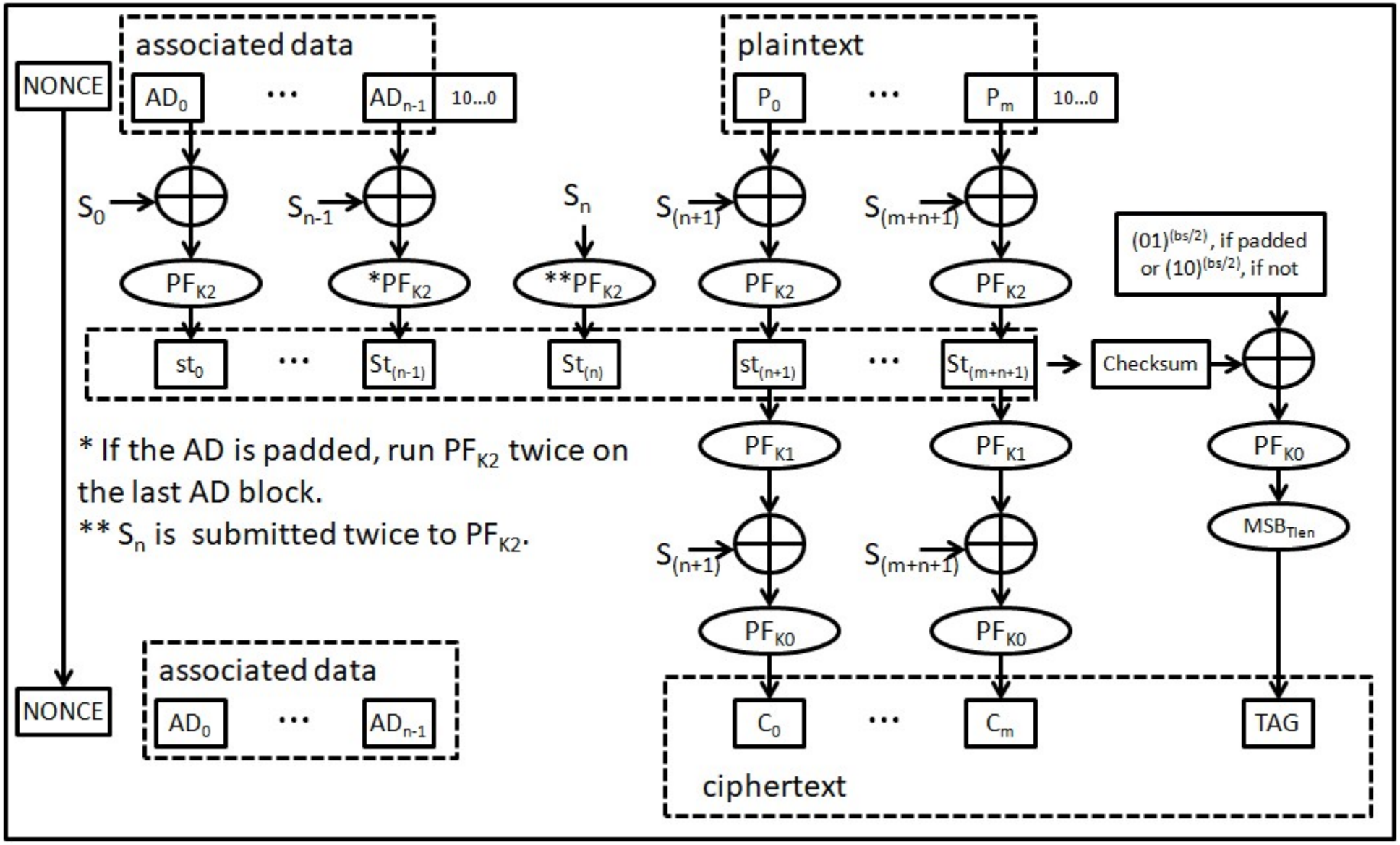
> Dear Alexandre Mège,
>
> Yes, this vulnerability exists on the original FlexAEAD. To solve the
> problem, a minor change need to be done.
>
> As you suggested, the Associated Data is padded with 10...0. If the
> padding happen, the PFK2 function is executed twice on the last AD
> block. This is done to avoid a collision with non padded AD block that
> ends with 10...0.
>
> After the change, here are examples showing that specific forgery is
> no longer possible.
>
> With empty PT:
>
> Key = 000102030405060708090A0B0C0D0E0F Nonce = 0001020304050607 PT =
> AD = 00 CT = 93E60FC2C0C84370
>
> Key = 000102030405060708090A0B0C0D0E0F Nonce = 0001020304050607 PT =
> AD = 0000 CT = 0A56F8AFF7EDE6C5

From: Eduardo Nascimento <edunasci@yahoo.com>
Sent: Monday, July 15, 2019 6:05 AM
To: lightweight-crypto; lwc-forum@list.nist.gov; Maria Eichlseder
Subject: Re: [lwc-forum] OFFICIAL COMMENT: FlexAEAD

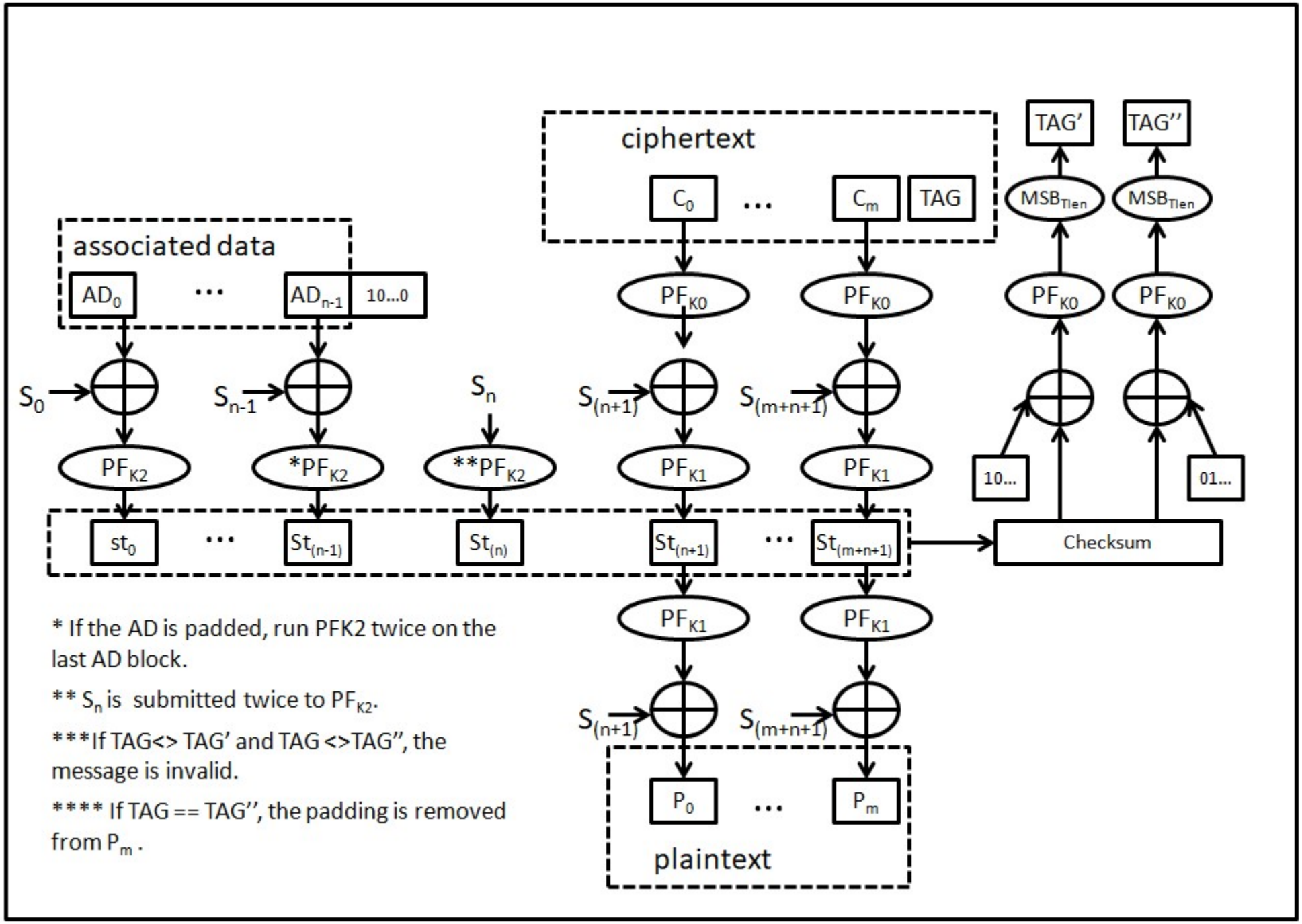
Dear Maria,

Thank you again for your analysis. We are proposing a small change on the algorithm that address these two potential problems.
Our preliminary analysis had shown the change will be enough. Please fell fell free to add any other valuable comments.
The solution is to include PFK2(PFK2(Sn)) on the checksum calculation, even when AD and M are empty (in this specific case PFK2(PFK2(S0)). When it is done, the NONCE will affect the calculation of the tag. This will avoid the forgery of the empty message and AD.
Normally the blocks are processed executing the PFK2 just once, in this case the PFK2 is executed twice. This will create enough separation from AD and M. The forgery (N, A || M_0, C_rest, T) will not be possible.

Encryption Diagram:



Decryption Diagram:



Kind Regards.
The FlexAEAD Team