THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 21, Number 1/2020, pp. 61–68

RELATED-KEY DIFFERENTIAL SLIDE ATTACK AGAINST FOUNTAIN V1

Raluca POSTEUCA

COSIC, KU Leuven, Belgium
Corresponding author: Raluca POSTEUCA, E-mail: raluca.posteuca@esat.kuleuven.be

Abstract. The stream cipher FOUNTAIN was introduced in April 2019 as one of the candidates in the NIST lightweight crypto standardization process. In this paper we introduce a slide attack that leads to the construction of 32 relations on key bits, with time complexity around 17×2^{80}. The success of the attack is around 98%. We also present some properties of the internal state transitions that allow the identification of (key-iv-ad) input data that produce identical ciphertexts, with probability of 2^{-32}.

Key words: lightweight cryptography, Fountain, slide attacks, internal states collisions, invertible states transition.

1. INTRODUCTION

Lightweight cryptography represents nowadays a very popular research area due to the necessity of designing and implementing efficient and secure cryptographic primitives for devices with constrained resources. During the last years various primitives were designed, having improved properties regarding the suitability for software and hardware implementation, performance and efficiency.

More details regarding the State of the Art in Lightweight Symmetric Cryptography can be found in [1].

The lightweight cryptography became even more attractive since NIST initiated a process meant to lead to the standardization of lightweight algorithms that are suitable for usage in constrained environments. The first round of this competition is on-going, having 56 submissions.

Our contribution. In order to contribute to the public research efforts dedicated to the analysis of NIST Round 1 candidates, we focused on one of the proposals, the Fountain stream cipher [3]. In this paper we introduce a slide attack with time complexity around 17×2^{80}. The goal of this attack is to construct a system of 32 low degree equations using the key bits, this being equivalent to recovering 32 bits of the key (by solving this equation system, the complexity of the exhaustive search is decreased from 2^{128} to 2^{96}). We have also identified some properties of the internal states transitions that allow the identification of (key, IV, AD) input data that produce identical ciphertexts, with probability of 2^{-32}.

Organization of the paper. The paper is organized as follows: the Fountain cipher is briefly described in Section 2; in Section 3 we present a set of properties of the state update function and present an attack on a slightly modified version of the cipher Fountain; in Section 4 we present an extension of this attack applicable to the original description of Fountain; Section 5 presents an algorithm that computes (key, IV, AD) tuples that produce identical ciphertexts; the last section concludes our paper.

2. DESCRIPTION OF FOUNTAIN CIPHER

Fountain is a lightweight stream cipher with 16-byte secret key K and 12-byte initialization vector IV. The input data of the cipher is the quartet (K, IV, AD, M), where AD represents the associated data and M the plaintext. The output of the authenticated encryption is a pair (C, T) where C represents the encryption of M and T is the authentication tag of AD and M.

The state update function of Fountain operates on 256-bits internal states, using 4 LFSRs, 4×4 S-boxes and a Boolean (filter) function used for the generation of keystream bits.

The 4 LFSRs used in Fountain have the same length, but different feedback polynomials:

$$LFSR1:\ \alpha_{64+i}=\alpha_{31+i}+\alpha_{25+i}+\alpha_{12+i}+\alpha_i,$$
$$LFSR2:\ \beta_{64+i}=\beta_{31+i}+\beta_{19+i}+\beta_{9+i}+\beta_i,$$
$$LFSR3:\ \gamma_{64+i}=\gamma_{31+i}+\gamma_{20+i}+\gamma_{14+i}+\gamma_i,$$
$$LFSR4:\ \zeta_{64+i}=\zeta_{31+i}+\zeta_{10+i}+\zeta_{6+i}+\zeta_i.$$

Three S-boxes are used in Fountain: SR_{kg}, SR_{ad} and SR_{tag}. At step *i*, an S-box is applied to the nibble $\zeta_{i+1}\gamma_{i+1}\beta_{i+1}\alpha_{i+1}$. The output nibble and the component bits are denoted by $f^i=f_1f_2f_3f_4$.

The filter function computes the keystream bits z_i such that

$$z_i=\alpha_{i+3}+\alpha_{i+11}+\beta_{i+20}+\gamma_{i+5}+\gamma_{i+16}+\zeta_{i+7}+\zeta_{i+29}+h(x),$$

where h is the Boolean function

$$h(x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)=x_0x_1\oplus x_2x_3\oplus x_4x_5\oplus x_6x_7\oplus x_0x_4x_8 \text{ and}$$

$$(x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)=(\zeta_{i+2},\ \alpha_{i+5},\ \beta_{i+4},\ \gamma_{i+11},\zeta_{i+23},\gamma_{i+27},\ \beta_{i+24},\ \alpha_{i+29},\ \zeta_{i+30}).$$

In order to produce an authenticated ciphertext using Fountain, the following phases are performed: the loading, the initialization phase, the associated data processing, the middle separation, the plaintext processing and the finalization and tag generation. A complete description of these phases can be found in [3].

In the first phase, the key and the nonce are loaded to the internal state, bit by bit. The 3rd, 6th, 7th and 8th bytes of the 4th LFSR are loaded with the constants $0x\mathrm{FF},0x\mathrm{FC},0x00,0x01$, as described in [3]. We denote this function by $load(K,IV)$.

The state update functions corresponding to each of the following phases can be viewed as a function parameterized by the used S-box and by the function used to compute the feedback of the 4 LFSRs. We denote the state update function by $Round_{x,S}$. The formal description of this function is presented in Fig. 1.

```
$Round_{x,S}$ ()
  Compute the keystream bit $z_i$
  Compute $f_1f_2f_3f_4 = S[\zeta_1\gamma_1\beta_1\alpha_1]$
  Run the 4 LFSRs 1 step with $z_i \oplus x \oplus f_j$ as feedback
```

Fig. 1 – One round of the cipher, parameterized by the bit x and the Sbox S.

Let $Round^n_{x,S}=\underbrace{Round_{x,S}\circ\ldots\circ Round_{x,S}}_{n \text{ times}}$. Then, the cipher's phases can be written as depicted in Table 1.

Observation 1. After the Middle separation phase, the first bit of the second LFSR is xored with 1:

$$\beta_{448+adLen}=\beta_{448+adLen}\oplus 1. \quad (1)$$

The initialization and the middle separation phases use the same state transition function, the only difference being given by the number of iterations performed (in the initialization phase are performed 384 iterations, while in the middle separation phase are performed only 64 iterations). Moreover, the initialization, the middle separation and plaintext processing phases use the same S-box. By getting $p_i=z_i$, where p_i is a plaintext bit and z_i is a keystream bit, and a null associated data, then the encryption process is composed by $384+64+pLen$ identical steps, where $pLen$ denotes the length of the plaintext. In this case, the ciphertext will be represented by the string 0^{pLen}.

Table 1

Fountain phases as a composition of the function $Round_{x,S}$ and the associated parameters

Phase	Description
Initialization	$Round^{384}_{0,SR_{kg}}$
Associated Data Processing	$Round^{ADlen}_{ad_i,SR_{ad}}$
Middle Separation	$Round^{64}_{0,SR_{kg}}$
Plaintext Processing	$Round^{Plen}_{c_i,SR_{kg}}$
Finalization	$Round^{384}_{0,SR_{tag}}$

3. PROPERTIES OF THE STATE UPDATE FUNCTION

3.1. Slide-based property of $Round^n_{x,S}$

In this section we introduce a slide-based property of the state update function $Round^n_{x,S}$. Slide attacks were introduced by Alex Biryukov and David Wagner in 1999 [2]. This type of attacks is based on the identification of a relation between two inputs that also holds for the corresponding outputs. This relation could be the composition of a fixed number of rounds, in the case of the block ciphers, or a set of state transitions, for stream ciphers. The identification of this type of relation may lead to the recovery of some secret data (key or plaintext bits).

Definition 1. Let (K_1, IV_1) and (K_2, IV_2) be two key-IV pairs. The pairs are called n-slid pairs if

$$Round^n_{0,SR\_kg}\left(load(K_1, IV_1)\right) = load(K_2, IV_2),\ n > 0 .$$

Observation 2. For a pair (K_1, IV_1) and a value n, there exists an *n-slid pair* if and only if, after computing $Round^n_{0,SR_{kg}}\left(load(K_1, IV_1)\right)$, the internal state satisfies the same property as a loaded state, i.e. the 3rd, 6th, 7th and 8th bytes of the 4th LFSR are equal to the constants $0x\mathrm{FF}, 0x\mathrm{FC}, 0x00, 0x01$.

LEMMA 1. *The minimum value of n for which there exist n-slid pairs is 47.*

Proof. For $n < 47$, the system of equations (3) is incompatible, i.e. the initial values of the constants directly influence the values of the bits $\zeta_{n+16} \cdots \zeta_{n+23}, \zeta_{n+40} \cdots \zeta_{n+63}$, making the constants pattern impossible. □

For $n = 47$, we conjecture that the system of equations corresponding to the bits $\zeta_{n+16} \cdots \zeta_{n+23}, \zeta_{n+40} \cdots \zeta_{n+63}$ is incompatible or that the last 47 bits of the 4th LFSR are not uniformly distributed, resulting in a very low probability of finding the correct values of the constants.

For $n = 48$, we have experimentally found *n-slid pairs* with probability 2^{-32}. One such example can be found in the technical report uploaded by the author on Cryptology ePrint Archive [4].

Observation 3. Since the pair (K_2, IV_2) is obtained after 48 rounds, then the following relations will always hold for a *48-slid pair*:

$$K_2[4 \times i] = K_1[4 \times i + 3],\ \ \forall i \in \{0,1,2\} ;$$

$$IV_2[4 \times i] = IV_1[4 \times i + 3], \forall i \in \{0,1,2\};;$$

$$K_2[12] = 0x00;\ \ K_2[13] = 0x80 .$$

Definition 2. Let $f:\mathbb{F}_q^n \to \mathbb{F}_q^n$ and $g:\mathbb{F}_q^n \to \mathbb{F}_q^n$. We say that two pairs $\left(a, f(a)\right)$ and $\left(b, f(b)\right)$ have a slide-based property with respect to the function g if $a = g(b)$ and $f(a) = g\left(f(b)\right)$.

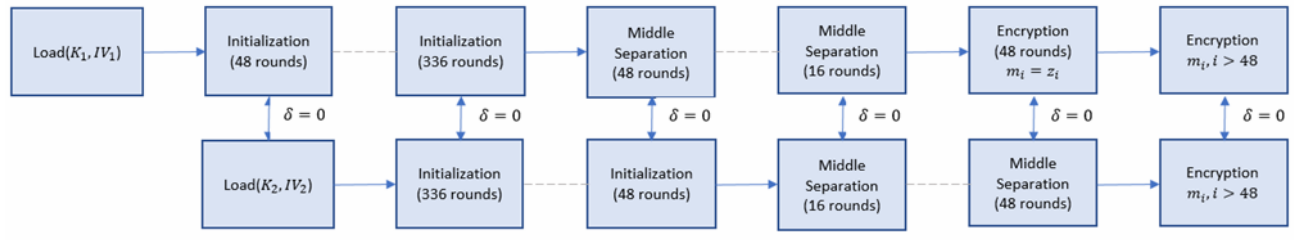
LEMMA 2. *Two n-slid pairs* (K_1, IV_1) *and* (K_2, IV_2) *define a slide-based property with respect to the state update function* $Round_{0,SR\_kg}^m$, $\forall m > 0$.

Proof. The proof is straightforward for $f = Round_{0,SR\_kg}^n$, $a = load(K_2, IV_2)$ and $b = load(K_1, IV_1)$. □

3.2. Slide attack on a modified version of Fountain

In this section we present a slide attack on a modified version of Fountain. The scope of the attack is to prove the existence of collisions of the keystream, independent of the length of the plaintexts.

Let us denote by Fountain$_{wx}$ the cipher Fountain without the Operation (1) and let us use a null AD. In this case, two *48-slid pairs* (K_1, IV_1) and (K_2, IV_2) will generate a slide behavior as follows:



Load(K_1, IV_1)
Initialization (48 rounds)
Initialization (336 rounds)
Middle Separation (48 rounds)
Middle Separation (16 rounds)
Encryption (48 rounds) $m_i = z_i$
Encryption $m_i, i > 48$
$\delta = 0$
Load(K_2, IV_2)
Initialization (336 rounds)
Initialization (48 rounds)
Middle Separation (16 rounds)
Middle Separation (48 rounds)
Encryption $m_i, i > 48$

Fig. 2 – Slide behavior of Fountain$_{wx}$.

The first process will perform the encryption of the plaintext $p_1 = z_{448}, \ldots z_{495}, m_{49}, \ldots m_{pLen-1}$ using the pair (K_1, IV_1), while the second process will encrypt the plaintext $p_2 = m_{49}, \ldots m_{pLen-1}$, using the pair (K_2, IV_2). Note that in both cases the associated data is null.

Since the Initialization and the Middle separation phases use the same parameters of the state update function, the internal states will satisfy the slide property for 448 rounds, until the first pair finishes the Middle Separation phase and begins the processing of the plaintext. Since the first 48 bits of the plaintext are the keystream bits, then, after the encryption of the first 48 bits, the two processes are perfectly synchronized (have the same internal state). Therefore, after this step, if the two processes are fed with the same plaintext, no matter the length of it, the corresponding ciphertexts will be equal.

Since the attack assumes the black-box hypothesis, an attacker does not have access to the correct values of the keystream bits $z_{448}, \ldots z_{495}$. If these 48 bits are uniformly distributed, the probability of correctly guessing them is 2^{-48}. Because the complexity of finding a *48-slid pair* is 2^{-32}, the probability of finding this slide behavior of Fountain$_{wx}$ is 2^{-80}. So, using approximately 2^{80} data of the type $(K_1, IV_1, z_{448}, \ldots z_{495})$, we will be able to find a collision on the ciphertexts.

4. DIFFERENTIAL SLIDE ATTACK ON FOUNTAIN

Using the original description of Fountain, including Operation (1), two *48-slid pairs* (K_1, IV_1) and (K_2, IV_2) generate the behavior depicted in Fig.3. Therefore, the behavior of the internal states for the first 448 rounds will remain unchanged. After the first process finishes the Middle separation phase, Operation (1) is applied, inducing a 1-bit difference between the corresponding internal states. In order to attack the Fountain cipher, we have analyzed the propagation of the differences induced after applying Operation (1) on both processes.

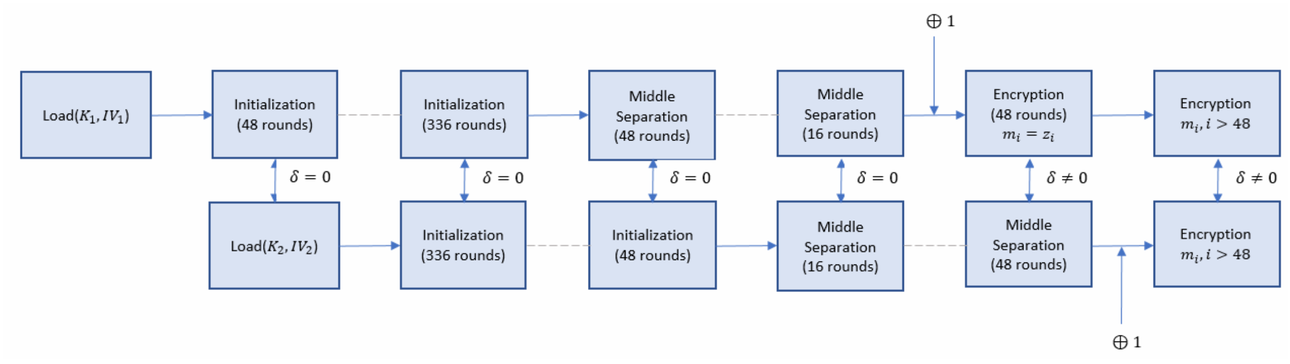

Load(K_1, IV_1)
Initialization (48 rounds)
Initialization (336 rounds)
Middle Separation (48 rounds)
Middle Separation (16 rounds)
Encryption (48 rounds) $m_i = z_i$
Encryption $m_i, i > 48$
⊕ 1
$\delta = 0$
$\delta = 0$
$\delta = 0$
$\delta = 0$
$\delta \neq 0$
$\delta \neq 0$
Load(K_2, IV_2)
Initialization (336 rounds)
Initialization (48 rounds)
Middle Separation (16 rounds)
Middle Separation (48 rounds)
Encryption $m_i, i > 48$
⊕ 1

Fig. 3 – Slide behavior of Fountain.

4.1. Differential characteristics of Fountain

In this section we will describe the manner in which we analyzed the distribution of the differences through the cipher Fountain. Note that one difference can be involved either in a linear manner (in the computation of a LFSR feedback or the linear part of the keystream bit generation function) or in a non-linear manner (in the non-linear part of the keystream bit generation function or in the S-box).

1. Linear behavior of the differences

Let ℓ be a linear function and let δ be an input difference. The output difference Δ is computed by applying the linear function ℓ over the input difference: $\Delta = \ell(x) \oplus \ell(x \oplus \delta) = \ell(\delta)$.

In this case, the probability of computing Δ knowing δ is $p = 1$.

2. Non-linear behavior of the differences through the S-box SR_{kg}

The behavior of differences propagation through an S-box are usually analyzed using the *Differential Distribution Table (DDT)*. For SR_{kg}, the input difference δ is connected to the output difference Δ, with a probability of $p = \frac{DDT[\delta, \Delta]}{16}$.

3. Non-linear behavior of the differences through the h function

Let $\delta = (\delta_0, \ldots, \delta_8)$ be the input difference of h. The output difference is computed as follows:

$$\Delta = h\left(\{x_i\}_i\right) \oplus h\left(\{x_i \oplus \delta_i\}_i\right)$$

Therefore, for a fixed input difference, the output difference Δ can be described as a Boolean function f_δ of the input bits $\{x_i\}_i$, the degree of this function being at most 2. Moreover, by observing the value of Δ, the attacker can learn a relation between the input bits $\{x_i\}_i$. For example, let's assume that only one $\delta_i = 1$:

- if $i \in \{1,3,5,7\}$, then $\Delta = x_{i-1}$; $P(\Delta = 0) = P(x_{i-1} = 0) = 0.5$;
- if $i = 8$, then $\Delta = x_0 x_4$; $P(\Delta = 0) = P(x_0 x_4 = 0) = 0.75$.

From the analysis of the difference propagation of Fountain, we remark the following:

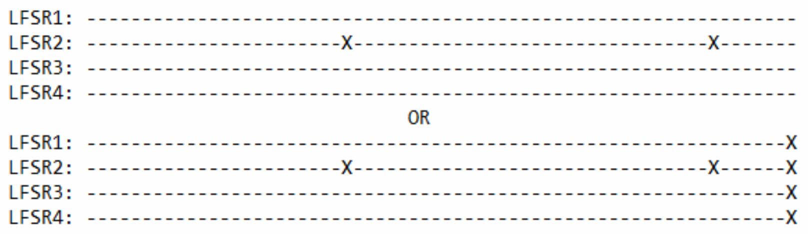
Operation (1) induces a difference between the internal states on the last bit of the 2nd LFSR. For 33 rounds, the difference will not be involved in any operation.

At Round 34, the difference will be involved, in a linear manner, in the computation of the feedback of the 2nd LFSR, resulting in the state difference depicted in Fig. 4. We denote the 1 difference by "X" and the 0 difference by "–". At the 41st round, the difference will be involved in the computation of the h function.

```
LFSR1: ----------------------------------------------------------------
LFSR2: ------------------------------X--------------------------------X
LFSR3: ----------------------------------------------------------------
LFSR4: ----------------------------------------------------------------
```

Fig. 4 – The state difference pattern at round 34.

The two possible output differences after 41 rounds are depicted in Fig. 5. The actual difference pattern will be fixed by the value of the 20th position of the second LFSR. If that value is 0, then the difference pattern will be the first one, thus there will only be two differences between the internal states.



```
LFSR1: ---------------------------------------------------------------------
LFSR2: -----------------------X-------------------------------------X-------
LFSR3: ---------------------------------------------------------------------
LFSR4: ---------------------------------------------------------------------
                                  OR
LFSR1: --------------------------------------------------------------------X
LFSR2: -----------------------X-------------------------------------X------X
LFSR3: --------------------------------------------------------------------X
LFSR4: --------------------------------------------------------------------X
```

Fig. 5 – The state difference patterns at round 41.

The difference will influence the application of the S-box for the first time at the 63rd round, the input difference of the S-box being $\delta = 2$. After this step, there will be 24 possible output difference patterns, having different appearance probabilities.

Observation 4. After computing one output difference pattern after n rounds, we have also generated the "observable" difference pattern, i.e. the keystream difference pattern used in the common part of the ciphertexts. The number of such patterns is smaller than the number of state difference patterns since not all the differences in the state are involved in the computation of the keystream bit difference.

After computing all possible output differences patterns for 102 steps, we obtain 2095680 difference patterns on the internal state and 83200 difference patterns on the observable part of the keystream. We store the 83200 difference patterns in a sorted list. Since one entrance in this list has 54 bits, the memory needed for this is approximately 4.4 MB.

4.2. Differential slide attack on Fountain

Using the observations and properties described in the previous section, we have designed a related-key differential slide attack on the Fountain cipher. The data needed for this attack is represented by 2^{80} random IV_1^i s and arbitrary 102-bit plaintexts p. The first 48 bits of p are used as the keystream bits $z_{448} \ldots z_{495}$.

We noticed that, after performing 45 rounds, the last bit of every LFSR will contain a nonzero difference, with probability 1. In order to cancel these differences, the bit $p[44]$ should introduce a difference in the internal states, so this value should be different from the keystream bit used in the second process, i.e. $p[44] = z_{492} \oplus 1$. If the first 48 bits of p are correctly guessed, then the first 48 bits of the ciphertext will satisfy the following conditions:

$$c_1[i] = 0, \ \forall i \neq 44, \ i < 48 \quad \text{and} \quad c_1[44] = 1. \qquad (2)$$

The same behavior of differences will also appear after performing 90 rounds. In order to cancel these differences, we will introduce a difference in the second plaintext in the 42nd position. More precisely, the second plaintext is defined as $p_2[i] = p[i+48], \forall i \in \{0,53\} \setminus \{41\}$.

The hypothesis of this attack is that the attacker has access to an encryption oracle. The oracle will answer two types of challenges:

- Given an initialization vector IV and a 102-bit plaintext p, the oracle will return the associated ciphertext c, under the secret key K and the input IV;
- Given an initialization vector IV_1 and a 54-bit plaintext p, the oracle will perform the following computations
  - compute $s = Round_{0,SR\_kg}^{48}\left(load(K, IV_1)\right)$
  - extract $\left(K_2^i, IV_2^i\right)$ from s
  - return the encryption of p under the input pair $\left(K_2^i, IV_2^i\right)$

The attack, presented in Fig. 6, works as follows. The attacker generates the list of all possible ciphertext differences, as described in the previous section. Then, he randomly generates 2^{80} initialization vectors IV_1^i and asks for the encryption of an arbitrary 102-bit plaintext p. If the first 48 bits of the ciphertext satisfy (2), then he asks for the encryption of the corresponding p_2. He computes the difference pattern $\{t_i = c_1[i] \oplus c_2[i+48]\}_i$ and checks if the pattern is contained in the precomputed list of differences. If the constraint holds, then the slide property holds.

The data complexity of the attack is 2^{80}, while the time complexity is around 17×2^{80}. The difference between the two complexities is explained by the time complexity of the search in the precomputed sorted list. Since the list contains $83200 < 2^{17}$ and the search in a sorted list can be performed in logarithmic time, the time complexity of the search is around 17.

```
For $i = 0$ to $2^{80} - 1$
    Randomly generate an initialization vector $IV_1^i$ and the arbitrary plaintexts $p, p_2$
    Ask for the encryption $c_1 = \text{Fountain}(K, IV_1^i, p)$
    If $c_1[i] = 0, \forall i \neq 45, i < 48$ and $c_{1[44]} = 1$
        Ask for the encryption $c_2 = \text{Fountain}(K_2^i, IV_2^i, p_2)$
        Compute $\{t_i = c_1[i] \oplus c_2[i+48]\}_i$
        If $\{t_i\}_i \in$ *precomputed list*
            Return $IV_1^i$;
**Key-recovery**
The pairs $(K, IV_1^i)$ and $(K_2^i, IV_2^i)$ are *48-slid pairs* $\Rightarrow$ 32 equations using the 128 bits of *K*
```

Fig. 6 – The pseudocode of the attack on Fountain.

Success probability. In our attack, in order to filter the right value of IV_1^i, we apply two filters. The first one is based on the values of $c_1[i]$, for $i < 48$. The probability that the condition of this filter is accomplished is 2^{-48}. The second filter is based on the identification of a "good" differential pattern on $\{t_i\}_i$. The probability of finding such a difference, for a *48-slid pair* (if the values of $z_{448} \ldots z_{495}$ were correctly guessed) is 1. The probability that such a difference is obtained for random pairs (K_1, IV_1) and (K_2, IV_2) is $2^{-37.66}$. Therefore, the probability that the algorithm described above outputs a IV_1^i that was generated by random input key-IV pairs is $2^{-5.66}$. Thus, the probability of success of our attack is 0.98.

5. OTHER OBSERVATIONS ON FOUNTAIN

The attack presented above is performed under the hypothesis that the associated data is null. We have also analyzed the cipher in the hypothesis that the additional data is not null. In this scenario, if the length of AD is higher than 4 bytes, then the space of the input of the cipher (until the Middle Separation phase) is $len = 128 + 96 + adLen$, where $adLen$ defines the length of the associated data. If $adLen > 32$, then $len > 256$ (the length of the internal state length). So, mathematically, this means that there will be collisions on the internal state.

We have performed a series of experiments with the goal of finding two input pairs (K_1, IV_1, AD_1) and (K_2, IV_2, AD_2) for which the internal state is the same. Such pairs can be found, with probability 2^{-32}, using the algorithm described in Fig. 7. Our algorithm exploits the invertibility of Fountain's state update function.

```
Randomly generate a fixed key $K, IV$ and $AD$
Compute the internal state $s$ obtained after the loading, initialization and associated data processing phases
For $i = 0$ to $2^{32} - 1$
    Generate at random a pair $(IV_i, AD_i)$
    Apply the inverse of the Associated Data Processing phase
    If the internal state is a valid loading state (the 32 constant bits are in the correct place)
        Extract and return $(K_i, IV_i)$
```

Fig. 7 – The pseudocode of the algorithm for finding pairs (K_1, IV_1, AD_1) and (K_2, IV_2, AD_2) that lead to the same internal state.

An example of input data pairs that lead to state collision are the following:

$$K_1 = \{\mathrm{FF}\}_{\mathrm{i}},\ IV_1 = \{\mathrm{F0}\}_i,\ AD_1 = \{00,01,02,03,04,05,06\}$$

$$K_2 = \{\mathrm{DA,7F,A4,1B,D3,0E,1D,EA,9B,CC,C7,AF,E3,3E,83,11}\}$$

$$IV_2 = \{\mathrm{FF,6F,A7,00,57,AF,EE,A0,94,19,91,CC}\}$$

$$AD_2 = \{\mathrm{7F,C1,A5,67,27}\}$$

6. CONCLUSION AND FUTURE WORK

In this paper we introduce a slide attack on full Fountain. The attack may concern questions regarding the security margin of the cipher in the related-key scenario. Although the attack involves the identification of (key, IV) pairs with a particular property, concerns about the structural properties and component operations arises. We also present (key-IV – associated data) tuples that lead to the same ciphertexts, in the case of enciphering the same message (the xor sum of the associated tags being equal to the xor sum of the initial keys).

The work presented can be extended in different directions. For example, it remains to be investigated if and how the attack presented in this paper can be improved in term of both data and time complexity. It will also be interesting to identify an attack scenario using the property regarding the internal state collision. Further research should also consider the analyze of Fountain in the single-key scenario.

ACKNOWLEDGEMENTS

The author would like to thank Vincent Rijmen, Tomer Ashur and all others for all the fruitful discussions and ideas regarding the cryptanalysis of Fountain.

REFERENCES

A. BIRYUKOV. L. PERRIN, *State of the art in lightweight symmetric cryptography*, Cryptology ePrint Archive, 2017, https://eprint.iacr.org/2017/511.pdf.

2. A. BIRYUKOV, D. WAGNER, *Slide attacks,* Proceeding of FSE'99, LNCS, **1636**, pp. 245-259, Springer Verlag, 1999.
3. B. ZHANG, *Fountain: A lightweight authenticated cipher (v1)*, NIST Information Technology Laboratory, CSRC, Lightweight Cryptography, Round 1 Candidates, https://csrc.nist.gov/projects/lightweight-cryptography/round-1-candidates, June 2019.
4. R. POSTEUCA, *Related-key differential slide attack against Fountain V1*, Cryptology ePrint Archive, Technical report, August 2019, https://eprint.iacr.org/2019/920.

Received July 10, 2019