Cryptanalysis of the FLIP Family of Stream Ciphers*

Sébastien Duval, Virginie Lallemand and Yann Rotella

Inria, project-team SECRET, France
{Sebastien.Duval, Virginie.Lallemand, Yann.Rotella}@inria.fr

Abstract. At Eurocrypt 2016, Méaux et al. proposed FLIP, a new family of stream ciphers intended for use in Fully Homomorphic Encryption systems. Unlike its competitors which either have a low initial noise that grows at each successive encryption, or a high constant noise, the FLIP family of ciphers achieves a low constant noise thanks to a new construction called *filter permutator*.
In this paper, we present an attack on the early version of FLIP that exploits the structure of the filter function and the constant internal state of the cipher. Applying this attack to the two instantiations proposed by Méaux et al. allows for a key recovery in 2^{54} basic operations (resp. 2^{68}), compared to the claimed security of 2^{80} (resp. 2^{128}).
Keywords: Stream Cipher, Guess-and-determine attack, FLIP, FHE.

1 Introduction

One of the challenges of recent years is to create an acceptable system of Fully Homomorphic Encryption (FHE) that would allow users to delegate computations to so-called Cloud Services. While Gentry showed in [6] the theoretic feasibility of such a framework, two main difficulties remain: first, the important computational and memory costs, and second the limited homomorphic capacities.

In order to overcome these limitations, one of the important aspects that have to be lessened is the cost of the evaluation of the symmetric encryption algorithm used in the framework, which mainly depends on the multiplicative depth of the circuit implementing the primitive. Since adapting the AES seems hard [4,7,3], several new symmetric schemes purposed for FHE have been proposed, among which the block cipher LowMC [1] and the stream ciphers Trivium and Kreyvium [2].

At Eurocrypt 2016, Méaux et al. [12] proposed the new stream cipher construction FLIP which aims at overcoming some of the drawbacks of previous schemes by, among other things, allowing for constant and smaller noise. This

*Partially supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015 and by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO.

achievement was made possible by the use of a new construction that resembles a filter generator but with a constant register that is permuted before entering the filtering function in order to limit the multiplicative depth of the circuit.

This design has been presented in October 2015 by the authors of FLIP at a national workshop [11], and then submitted to Eurocrypt 2016. Our study shows that the concrete instantiations proposed by the designers suffer from several flaws that can be extended to a cryptanalysis. We reported our findings to the authors which led them to change their design after their paper was accepted, in order to resist our attack. A fixed version of the construction is then described in the final version of the Eurocrypt 2016 article entitled *Toward Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts* [12].

In the following, we only deal with the *preliminary version* of the FLIP family of stream ciphers: so everytime that we mention "FLIP " we mean the version presented in [11] and submitted to Eurocrypt 2016 (which differs from the final version of [12]).

This paper is organised as follows. We start by giving a description of the submitted version of the FLIP family of stream ciphers in Section 2. Then, we discuss its vulnerabilities against guess-and-determine attacks in Section 3 and show how to break the cipher by exploiting these vulnerabilities through an algebraic attack (Section 4). The pseudocode of the attack is given in Section 5. Our analyses are supported by experiments reported in Section 6. The last section concludes this paper.

2 Description of the FLIP Family of Stream Ciphers

2.1 General Idea: the Filter Permutator Structure

When it comes to designing a symmetric construction tailored for FHE applications, both block and stream ciphers can be considered, each with advantages and disadvantages, as discussed in [2].

Since the targeted applications use noise-based cryptography (such as lattice-based cryptography), one of the pursued goals is to limit the growth of noise as much as possible, which is equivalent to considering circuits with a limited multiplicative depth. This desirable property, also refered to as a high *homomorphic capacity*, is hard to obtain with block ciphers since the round iterations lead to an output with a large algebraic degree. However, the good point is that the noise is constant per block, which implies that noise does not add any limitation on the number of generated ciphertext blocks. On the other hand, the homomorphic capacity of stream ciphers is usually very high for the first ciphertext bits, but decreases as more bits are generated, imposing to re-initialise the cipher or to use techniques like bootstrapping.

The innovative design of Méaux et al. [12] succeeds in taking the best from both sides and enjoys a very good homomorphic capacity that remains constant with time. Their proposal is a family of stream ciphers named FLIP that is based on the filter generator construction, but drops the register update part

to avoid the algebraic degree increase. Instead, the register bits are permuted before entering the filter function, thus the name *filter permutator*.

Its operational principle is represented in Figure 1.

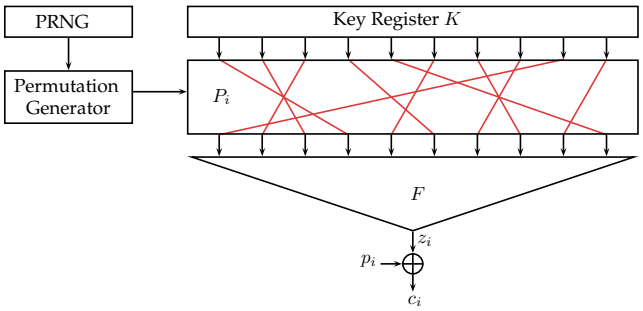

Fig. 1. General structure of the filter permutator construction used for the FLIP family of stream ciphers.

It is made up of three main components:

- a register storing the N-bit key K,
- a bit permutation generator, parametrised by a public PseudoRandom Number Generator (PRNG), producing at each clock i an N-bit permutation P_i,
- a filtering (Boolean) function F, generating the keystream bit z_i.

Once the PRNG is initialised using an IV, the master key K is loaded into the register and encryption starts: at each step i, the permutation generator produces a permutation P_i that shuffles the key bits right before they enter the filtering function F. F produces the keystream bit z_i which is XORed to the corresponding plaintext bit p_i and gives the ciphertext c_i.

To recover the plaintext, the same process is used to generate the bits z_i of the keystream that are simply XORed back with the c_i.

Every part of the scheme is public except the key.

2.2 The FLIP Family of Stream Ciphers

After an extensive analysis of the filter permutator construction with respect both to FHE constraints and resistance against most common stream ciphers attacks, the authors chose the concrete instantiation we now describe.

The PRNG used for the permutation generator is defined as a *forward secure* PRNG based on AES-128, and the permutation generator itself is a Knuth shuffle [9], which ensures that all the N-bit permutations have the same probability to be generated (provided that it is used with a random generator).

F is an N-variable Boolean function defined by the direct sum of three specific Boolean functions f_1, f_2 and f_3 that are defined in [13] and that we now recall.

In the following, n and k are positive integers and operations are considered over $\mathbb{F}_2$.

Definition 1 (L-type Function). *The n-th L-type function L_n is the n-variable linear function defined by:*

$$L_n(x_0, \cdots, x_{n-1}) = \sum_{i=0}^{n-1} x_i.$$

Definition 2 (Q-type Function). *The n-th Q-type function Q_n is defined by the $2n$-variable quadratic function:*

$$Q_n(x_0, \cdots, x_{2n-1}) = \sum_{i=0}^{n-1} x_{2i}x_{2i+1}$$

Definition 3 (T-type Function). *The k-th T-type function is the $\frac{k(k+1)}{2}$-variable Boolean function defined by:*

$$T_k(x_0, \cdots, x_{\frac{k(k+1)}{2}-1}) = \sum_{i=1}^{k} \prod_{j=0}^{i-1} x_{j+\sum_{\ell=0}^{i-1}\ell}.$$

For instance the 3rd T-type function is equal to:

$$T_3 = x_0 + x_1x_2 + x_3x_4x_5$$

Each of these types of functions has nice properties according to one or several security criteria (non-linearity, resiliency, algebraic immunity, ...).

The filtering function used in the FLIP family of stream ciphers uses a combination of 3 Boolean functions parameterised by the integers n_1, n_2 and n_3, chosen so that the resulting properties of F are good:

- $f_1(x_0, \cdots, x_{n_1-1}) = L_{n_1}$,
- $f_2(x_{n_1}, \cdots, x_{n_1+n_2-1}) = Q_{n_2/2}$,
- $f_3(x_{n_1+n_2}, \cdots, x_{n_1+n_2+n_3-1}) = T_k$ where k is such that $n_3 = \frac{k(k+1)}{2}$.

F is defined as the direct sum of f_1, f_2 and f_3:

$$F(x_0, \cdots, x_{n_1+n_2+n_3-1}) = L_{n_1} + Q_{n_2/2} + T_k \quad \text{where} \quad n_1 + n_2 + n_3 = N$$

and thus inherits in some measure the good properties of f_1, f_2 and f_3 (see [13]).

The initial analysis performed by Méaux et al. and presented in the submitted version of the construction [13] takes into account the most common attacks on filter generators (which are very similar to filter permutators) and resulted in the selection of the parameters reported in Table 1.

Table 1. Parameters of the two concrete instantiations of FLIP with the corresponding complexities of Algebraic Attacks (AA), Fast Algebraic Attacks (FAA) Higher-Order Correlation attacks (HC).

FLIP (n_1, n_2, n_3)	key size (N)	Security	AI (F)	FAI (F)	res (F)	AA	FAA	HC
FLIP (47,40,105)	192	80	14	15	47	194	88	119
FLIP (87,82,231)	400	128	21	22	87	323	136	180

The security analysis detailed in [13] and more precisely the study of weak keys results in an additional limitation on the design which is that the key must be balanced (in the submitted version of the Eurocrypt paper [13] it is stated that: "*Since our N parameter will typically be significantly larger than the bit-security of our filter permutator instances, we suggest to restrict the key space to keys of Hamming weight $N/2$*").

Finally, note that the specification document does not give any limit on the number of keystream bits that can be generated under the same key.

3 Preliminary Remarks on the Vulnerabilities of the FLIP Family of Stream Ciphers

3.1 Attack Scenario and Computation Model

In the following we examine one of the most common attack scenarios considered for stream cipher analysis, which is the known-plaintext scenario: we suppose that we know a part of the plaintext together with the corresponding ciphertext, which implies that we know the value of some bits of the keystream z. Needless to say, our goal is to recover the secret key, which in the case of the FLIP family of stream ciphers is equivalent to recovering the internal state.

To express the performance of our attack, we use the three usual metrics which are time, data and memory complexities. Time complexity (hereafter denoted by C_T) expresses the quantity of operations that the attacker has to perform to execute the attack. In our case, we compute it in the same way as in the specification paper [13] so we count the number of basic operations. Data complexity (C_D) corresponds to the required number of keystream bits and finally memory complexity (C_M) measures the memory (in bits) needed during the attack.

3.2 The FLIP Family of Stream Ciphers and Guess-and-Determine Attacks

The attack we propose uses a variant of the guess-and-determine technique. This approach, which seems to have been named first in [5,8], has been extensively used to analyse stream ciphers, starting with the ones submitted to the NESSIE

project. The idea is to start by making a hypothesis on the value of some bits of the internal state or of the key (the 'Guess') and to use the information coming from keystream bits to deduce the unknown ones (to 'Determine' them). Most of the time the attack is completed thanks to algebraic techniques.

Two features of the FLIP family of stream ciphers seem to indicate that an attack using guess-and-determine techniques would be efficient: first its fixed internal state and second the definition of its filtering function. More precisely, the fact that the register is not updated implies that a guess of one key/internal state bit at any time would give an information of one bit at any other time. This is different from common stream ciphers for which the update function mixes the internal state bits together, implying that a one-bit information quickly vanishes after some (forward or backward) rounds.

The second feature that seems exploitable for a guess-and-determine attack is the definition of the filtering function F which contains very few monomials of high-degree. This is what we detail now.

3.3 Observations on the Boolean Function F

As reported before, the Boolean function F is made of the direct sum of 3 Boolean functions f_1, f_2 and f_3 which are respectively of L-, Q- and T-type. This definition implies that all the monomials of degree greater than or equal to 3 are present in f_3, which is given by the following formula:

$$\begin{aligned} f_3(x_{n_1+n_2}, \cdots, x_{n_1+n_2+n_3-1}) &= T_k(x_{n_1+n_2}, \cdots, x_{n_1+n_2+n_3-1}) \\ &= \sum_{i=1}^{k} \prod_{j=n_1+n_2}^{n_1+n_2+i-1} x_{j+\sum_{\ell=0}^{i-1}\ell} \end{aligned}$$

where k is the algebraic degree of f_3 and is such that $n_3 = \frac{k(k+1)}{2}$.

From this expression, we see that there are $k-2$ monomials of degree greater than or equal to 3 in F, in a total of $n_3 - 3$ variables. Given the multiplicative depth constraint, k has to be low[1], which implies that the T-type function has few monomials and therefore easy to cancel, as we show in the next section.

The core idea of our attack is to notice that since there are few high-degree monomials but a lot of null key bits, there is a high probability that the high-degree monomials of F are cancelled. In these cases, it also means that the keystream bits can be seen as expressions of degree less than or equal to 2 in the non-null key bits.

The attack we perform uses this specificity by doing a slight variant of the guess-and-determine technique: instead of making a hypothesis on the value of key/internal state bits, we guess the indices of some null key bits[2]. We deduce

[1] To give the order of magnitude, we recall here that the 2 concrete instantiations described in [13] use $k = 14$ and $k = 21$ for respective security of 80 and 128 bits.

[2] As we saw in Section 2, we are sure that there are $\frac{N}{2}$ null key bits.

from that the clocks when the keystream bits are an expression of low-degree in the other key bits and build a system from it. Finally we solve the system with linearisation techniques, which in the case of low-degree equations is of reasonable cost.

3.4 Probability of Cancelling all the High-Degree Monomials of F given that ℓ Input Variables are Null

To figure out the feasibility of such a procedure, we have to evaluate the probability that, given exactly ℓ positions of null bits in K, the expression of the keystream bit z_i is of degree less than or equal to 2 in the remaining key bits[3].

This probability is directly linked to the amount of data that is required to lead to the attack since it determines the amount of keystream bits that an attacker needs such that enough of them are exploitable to construct the system.

From the previous discussion, we know that there are exactly $k-2$ disjoint monomials of degree greater than or equal to 3 in the expression of $z_i = F(P_i(k_0, k_1, \cdots k_{N-1}))$. Then, if the attacker is only aware of $\ell < k-2$ zero positions, she won't be able to determine exploitable clocks, which forces $\ell \geq k-2$, i.e. at minimum one zero bit that could be positioned in each of the high-degree monomials.

First case: if $\ell = k-2$. The first possibility is to choose the number of null positions equal to the number of high-degree monomials that we want to cancel, i.e. $\ell = k-2$. In this case, exactly one null bit has to go into each monomial: for instance, if we are looking at a specific monomial of degree d: $x_0 x_1 \cdots x_{d-1}$, it is equivalent to choosing which of the variables is null, so there are d possibilities. From that, we can enumerate the set of valid configurations, which corresponds to choosing one index in each monomial, so since there are 3 possible choices for the monomial of degree 3, 4 possibilities for the one of degree 4 and so on up to the monomial of degree k, there is a total of $3 \times 4 \times 5 \cdots \times k = k!/2$ valid configurations. To obtain the probability, this amount has to be compared with the total number of possibilities for choosing the null positions, which is $\binom{N}{\ell}$ so we have:

$$\mathbb{P}_{\ell=k-2} = \frac{k!/2}{\binom{N}{\ell}}.$$

General case: if $\ell \geq k-2$. To increase the probability that a clock is exploitable, the attacker can guess more null key bit positions and choose $\ell \geq k-2$. A first way of computing this probability is:

$$\mathbb{P}_\ell = \frac{\sum_{i_1+i_2+\cdots+i_{k-2} \leq \ell} \binom{3}{i_1}\binom{4}{i_2}\cdots\binom{k}{i_{k-2}}\binom{N-m}{\ell-I}}{\binom{N}{\ell}}$$

where m is the number of variables that occur in the monomials of degree greater than or equal to 3 and $I = i_1 + i_2 + \cdots + i_{k-2}$.

[3]This is what we denote by an *exploitable equation* or *exploitable clock*.

Proof. Suppose that we are given ℓ null bit positions in K. We are interested in the probability that a random permutation P_i shuffles the key bits in a way that the evaluation of F does not contain any monomial of degree greater than or equal to 3. As previously, we count the number of valid configurations among the total number of permutations.

The idea is to list all the possible ways of positioning at least one null bit in each monomial: we set i_1 null bits in the monomial of degree 3, i_2 null bits in the monomial of degree 4, and so on up to i_{k-2} null bits in the monomial of highest degree (k). If we denote by $I = i_1 + i_2 + \cdots + i_{k-2}$ the number of null bits positioned in such a way, we are left with $\ell - I$ null bits to position in the other $N - m$ monomials. To obtain the probability, we have to divide this quantity by the number of ways to position ℓ guesses among N bits.

□

Another way of obtaining the probability is to compute the number of configurations that do not cancel the monomials of degree greater than or equal to 3, which is the complementary probability of the one we are looking for. The advantage is that this complementary can be easily expressed with the inclusion-exclusion principle. Let us denote A_J the event that our guess doesn't cancel the monomials of degrees included in the set J, i.e.

$$A_J \text{ is the event: } \{\forall j \in J,\, M_j \neq 0\}$$

where M_j is the unique monomial of degree j in T_k.

$\mathbb{P}(A_J)$ is the probability of setting the ℓ bits among the monomials whose degrees are not in J so is equal to:

$$\mathbb{P}(A_J) = \frac{\binom{N - \sum\limits_{j \notin J} j}{\ell}}{\binom{N}{\ell}}$$

Then we can express the probability that our guess yields a polynomial of degree higher than or equal to 3 by:

$$\mathbb{P}\Big(\bigcup_{d \in \{3,\cdots,k\}} A_{\{d\}}\Big) = \sum_{s=1}^{k-2} \left((-1)^s \sum_{\substack{J \subseteq \{3,\cdots,k\} \\ |J| = s}} \mathbb{P}(A_J) \right)$$

which can be expressed as

$$\mathbb{P}\Big(\bigcup_{d \in \{3,\cdots,k\}} A_{\{d\}}\Big) = \frac{\sum\limits_{s=1}^{k-2} \left((-1)^s \sum\limits_{\substack{J \subseteq \{3,\cdots,k\} \\ |J| = s}} \binom{N - \sum\limits_{j \notin J} j}{\ell} \right)}{\binom{N}{\ell}}$$

From which we get the expression of the probability that we are looking for:

$$\mathbb{P}_\ell = \mathbb{P}\Big(\bigcap_{d \in \{3,\cdots,k\}} \overline{A_{\{d\}}}\Big) = 1 - \mathbb{P}\Big(\bigcup_{d \in \{3,\cdots,k\}} A_{\{d\}}\Big)$$

The evaluation of these formulas gives the results reported in Tables 3, 4 and 5 in Appendix, and we will see in the next section that they are good enough to mount an attack. For instance, if we attack the small version[4] of FLIP and do the minimal number of guesses (i.e. $\ell = 12$) we will have a probability of having an exploitable equation of $\mathbb{P}_{\ell=12} = 2^{-26.335}$. For the other version[5] and a minimal number of guesses we have $\mathbb{P}_{\ell=19} = 2^{-42.382}$.

4 Our Attack

4.1 Description

Setting. Since we consider a known-plaintext scenario, we suppose that we are given C_D keystream bits that we denote by z_i, $i = 0, \cdots, C_D - 1$. Additionally, the associated permutations P_i are public so we have expressions of the keystream bits as function of the unknown key bits $k_0, \cdots, k_{N-1}$:

$$z_i = F(P_i(k_0, k_1, \cdots, k_{N-1})) \quad \forall i \geq 0$$

Our attack takes advantage of the two vulnerabilities detailed in the previous section to boil down the key recovery problem to the solving of a linearised system.

First step: initial guess. The first step consists in making a hypothesis on the positions of ℓ null key bits, where $\ell \geq k - 2$. Assuming that these bits are null gives us a simplified expression of z_i in only $N - \ell$ unknowns. Since the key K is balanced, the probability of our guess being right is[6]:

$$\mathbb{P}_{rg} = \frac{\binom{\frac{N}{2}}{\ell}}{\binom{N}{\ell}}$$

Second step: extraction of low-degree equations. The objective of step 2 is to collect equations of low-degree in the unknown key bits. To do so, we look at the expressions of the available z_i and pick up all the equations for which the null key bits cancel the monomials of degree greater than or equal to 3. As seen in previous section, this event is of probability $\mathbb{P}_\ell$.

Third step: solving the system. One of the easiest ways of solving the quadratic system is to use linearisation techniques, which consist in converting the system into a linear one by introducing a new variable for each non-linear monomial that appears. In our specific case, the only non-linear expressions we have to deal with are the monomials of degree 2. Since F takes as input N variables but we guessed ℓ of them, we are left with $N - \ell$ unknown variables,

[4] FLIP (47,40,105)

[5] FLIP (87,82,231)

[6] This probability is slightly smaller than in the case of a random key ($2^{-\ell}$), but the advantage is that as long as we guess $\ell \leq \frac{N}{2}$ we are sure that at least one guess will be correct while it could fail for a random key that does not have enough null bits.

which in the worst case scenario form $\binom{N-\ell}{2}$ monomials of degree two. This implies that once linearised, our converted system will contain

$$v_\ell = N - \ell + \binom{N-\ell}{2}$$

variables.

Assuming that the equations are random, the number of equations that are necessary to give a unique solution (or show a contradiction) is roughly equal to the number of unknowns[7]. This implies that the necessary amount of keystream bits that the attacker needs is the product of the number of variables and the inverse of the probability that a z_i is exploitable:

$$C_D = v_\ell \times \frac{1}{\mathbb{P}_\ell}$$

The time complexity is determined by the time to solve the system[8] multiplied by the number of times we have to repeat the guess of ℓ null bit positions before finding a correct one:

$$C_T = v_\ell^3 \times \frac{1}{\mathbb{P}_{rg}}$$

The final memory complexity is dominated by the memory necessary to store the system, so is roughly equal to:

$$C_M = v_\ell^2$$

Tables 3, 4 and 5 in Appendix give the possible trade-offs between time and data complexity for the two versions of FLIP. As we can see, increasing the number of initial guesses ℓ allows to reduce the amount of data necessary to conduct the attack at the cost of an increased time complexity.

4.2 Discussion and Possible Improvements

Data Complexity Reduction. The data complexity can be further improved if, rather than choosing the guesses at random, the attacker chooses them according to the observed permutations. With the PRNG seed being public, at any point in time, she knows all the upcoming permutations so she can deduce a guess that cancels the triangular part for many of the upcoming permutations.

Possibility of Precomputations. Most of the computational cost of the attack lies in the linear system solving. Notice that this linear system depends only on the permutation and the guess, which are all known to the attacker, who can therefore compute the system inversion for several guesses without any

[7] This will be confirmed by our experiments detailed in Section 6.

[8] Which is v_ℓ^3 for a basic Gaussian elimination or $v_\ell^{2.8}$ with Strassen's algorithm. We will use the first one for simplicity.

knowledge of the keystream. Once she receives the keystream bits, she plugs them into her precomputations to obtain the results. The drawback of this technique is its increase in memory complexity.

Seed Independence. Our attack has the property of being unaffected by a re-initialisation of the system. What we mean here is that a change of the PRNG seed in the middle of the attack will not force the attacker to restart her attack: she can combine the previously obtained equations with the one obtained under the new seed.

Security. The security level of the FLIP family of stream ciphers is at most proportional to $\sqrt{N}$ bits, where N is the key size.

Proof. The time complexity of our attack is

$$C_T = v_\ell^3 \times \frac{1}{\mathbb{P}_{rg}}$$

As $\ell \ll N$, one can say that $\mathbb{P}_{rg}$ is roughly equivalent to $2^{-\ell}$. Also, as $v_\ell = N - \ell + \binom{N-\ell}{2}$, we can give an approximation of C_T which is

$$C_T \sim N^6 \times 2^\ell$$

Additionally, the number of guesses we need to perform our attack is the number of monomials of degree greater than or equal to 3 in T_{n_3}. Thus $n_3 = (\ell+2)(\ell+3)/2$, so $\ell \sim \sqrt{n_3}$, from which we get:

$$\log C_T \sim \alpha\sqrt{N}$$

□

Figure 2 represents the evolution of the time complexity of our attack as function of the key size when we consider instances of FLIP of the form FLIP (n_1, n_2, n_3) where $N = n_1 + n_2 + n_3 = 2n_3$ (which is consistent with the parameters proposed in [13]). ℓ is chosen as the minimal number of guesses needed to perform the attack, i.e. $\ell = k - 2$.

Attempt to Cancel the Quadratic Part. Our attack consists in guessing key bits to cancel the triangular part of the filtering function: another possibility would be to cancel the monomials of degree 2 in order to reduce the resistance of the scheme against correlation attacks. We considered this option but our studies showed that the complexity of such an attack would be too high. We also thought of cancelling both quadratic and triangular parts, thus leaving only linear relations, but the data complexity of such an attack makes it less practical.

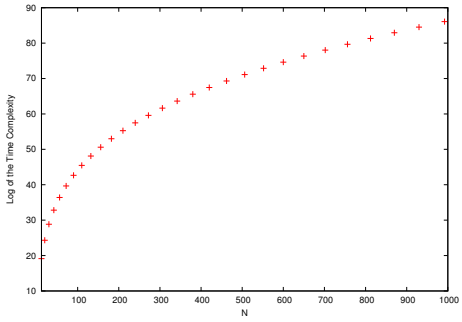

Fig. 2. Evolution of the Time Complexity as function of the Key Size N.

5 Description of the Algorithm

The main computation part of our attack is a linear system solving over $\mathbb{F}_2$. If the solving detects a contradiction, we deduce that our guess is wrong and we start again with another guess. Otherwise, the guess was right and the solving yields the key. The intuition is that we don't always need a full-rank system to detect a contradiction. We can therefore improve the attack by treating every equation as they come, rather than waiting for a full-rank system.

A pseudocode description of the attack using this improvement is given in Algorithm 1. In this algorithm, an equation will be represented as a $(v_\ell + 1)$-bit word containing 1 where a variable is present in the equation and 0 otherwise. The least significant bit of this representation contains the value of the keystream bit of the equation. We also memorise if equation i is present in the system through the vector $Exists$. If $Exists[i] = 1$, then equation i is in the system.

6 Verification of the Attack on a Toy Version

To support our findings, we implemented our attack on a toy version of the cipher. We reduced the key size to $N = 64$ bits and adapted accordingly the values of the parameters to $n_1 = 14$, $n_2 = 14$ and $n_3 = 36$ (the proportions between the size of the parameters are kept). The filtering function F has algebraic degree 8 and is defined as follows:

$$F(x_0, \cdots x_{63}) = f_1(x_0, \cdots x_{13}) + f_2(x_{14}, \cdots x_{27}) + f_3(x_{28} \cdots x_{63})$$

where:

$$\begin{aligned} f_1(x_0, \cdots, x_{13}) &= L_{14}(x_0, \cdots, x_{13}) = x_0 + x_1 + \cdots + x_{13} \\ f_2(x_{14}, \cdots, x_{27}) &= Q_7(x_{14}, \cdots, x_{27}) = x_{14}x_{15} + x_{16}x_{17} + \cdots + x_{26}x_{27} \\ f_3(x_{28}, \cdots, x_{63}) &= T_8(x_{28}, \cdots, x_{63}) = x_{28} + x_{29}x_{30} + x_{31}x_{32}x_{33} + \cdots + x_{56}x_{57} \cdots x_{63} \end{aligned}$$

Algorithm 1 FLIP Key recovery

Input: Keystream, PRNG seed
Output: Key

SYSTEM ← Vector of v_ℓ null words
$Exists$ ← Vector of v_ℓ null bits
$KeyNotFoundYet$ ← **true**
while $KeyNotFoundYet$ **do**
    $G \leftarrow NewRandomGuess$
    $NoContradiction$ ← **true**
    $N_{eq} \leftarrow 0$
    while $NoContradiction$ **and** $N_{eq} \leq v_\ell$ **do**
        $E \leftarrow NewEquation$
        $NewIndex \leftarrow -1$
        $i \leftarrow \text{MSB}(E)$
        while $i \leq v_\ell$ **do**
            if $Exists[i]$ **then**
                $E \leftarrow E \oplus \text{SYSTEM}[i]$
                $i \leftarrow \text{MSB}(E)$
            else
                if $NewIndex = -1$ **then**
                $NewIndex \leftarrow i$
                end if
                $i \leftarrow$ Index of the next bit with value 1 starting from index $i+1$ and going towards the LSB
            end if
        end while
        for $j = 1$ to $NewIndex$ **do**
            if $Exists[i]$ **and** $\text{SYSTEM}[i][NewIndex] = 1$ **then**
                $\text{SYSTEM}[i] \leftarrow \text{SYSTEM}[i] \oplus E$
            end if
        end for
        if $E = 1$ **then**
            $NoContradiction$ ← **false**
        else
            if $E \neq 0$ **then**
                $\text{SYSTEM}[NewIndex] \leftarrow E$
                $N_{eq} \leftarrow N_{eq} + 1$ {If $E = 0$, the equation is linearly dependent from the first ones but brings no contradiction, we then don't increment N_{eq}}
            end if
        end if
    end while
    if $NoContradiction$ **then**
        Get x_i and $x_i x_j$ and Test if there is a contradiction
        if There is no contradiction **then**
            $KeyNotFoundYet$ ← **false**
            $Key = (x_i)_{1 \leq i \leq n}$
        end if
    end if
end while
return Key

According to our analysis, the parameters of the attacks are the ones described in Table 6: for instance if we decide to make a hypothesis on $\ell = 8$ null indices, the probability that our guess is correct is

$$\mathbb{P}_{rg} = 2^{-8.717}.$$

The probability that a permutation is exploitable is equal to:

$$\mathbb{P}_{\ell} = 2^{-7.814}$$

and the linearised system depends on $v_\ell = 1596$ variables. We expect that $C_D = 2^{18.454}$ bits are necessary to conduct the analysis and that the attack requires $C_T = 2^{40.638}$ basic operations.

We implemented our own version of this toy instance of FLIP on which we performed our attack with $\ell = 8$ guesses. The statistics we obtain are given in Table 2.

Although the equations have a very specific structure, we noticed that they behave like random equations in the following sense: the first linearly dependent equation is only found after generating 1590 equations, which fits with the theory in the case of random equations [10]. However, treating the equations as they come allows us to discard right away any equation that is linearly dependent from the others. This way, we can stop collecting equations as soon as we have as many equations in our system as are variables [9].

Table 2. Comparison of the experimental results with theory: attack on the toy version FLIP (14,14,36) with a hypothesis on $\ell = 8$ null bit positions (average on 1000 tests, launched on an Intel(R) Xeon(R) CPU W3670 at 3.20GHz (12MB cache), and with 8GB of RAM)

	Guesses	Data Generated	Ratio Exploited	Elementary Op.	Time (sec)
Practice	437.1	$2^{18.455}$	$2^{-7.813}$	$2^{38.588}$	280.93
Theory	$\mathbb{P}_{rg}^{-1} = 420.8$	$C_D = 2^{18.454}$	$\mathbb{P}_{\ell} = 2^{-7.814}$	$C_T = 2^{40.638}$	$\varnothing$

As we can see in Table 2, experimental results fit pretty well with the theory.

7 Conclusion

In this paper we presented a cryptanalysis of the FLIP family of stream ciphers. Our attack makes use of the weaknesses of the FLIP structure against guess-and-determine attacks to reduce the degree of the filtering function, after what an

[9] The experiments show that we discard about 500 equations before we get 1596 independent equations.

algebraic attack suffices to recover the key. We obtained theoretical estimations of the complexity of the attack and an implementation of the attack on a toy version shows that this complexity holds in practice. This attack can be performed in 2^{54} basic operations (resp. 2^{68}), compared to the claimed security of 2^{80} (resp. 2^{128}), and we discussed trade-offs and improvements that can lower this complexity even more. We also underlined that a simple increase of the key size is not an efficient countermeasure as the complexity of the attack doesn't increase much with the key size.

Finally, in view of fixing this attack, one should keep in mind the inherent weakness of the filter permutator construction against guess-and-determine attacks due to its constant register. The biggest issue of the FLIP family of stream ciphers is that its filtering function increases the fragility against guess-and-determine attacks. To strengthen the security of the filter permutator, a possible direction would be to refine its filtering function, for instance by using more high-degree monomials.

References

1. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 430–454. Springer (2015)
2. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: How to Compress Homomorphic Ciphertexts. In: Fast Software Encryption FSE 2016 (to appear), also available at `http://eprint.iacr.org/2015/113`
3. Coron, J., Lepoint, T., Tibouchi, M.: Scale-Invariant Fully Homomorphic Encryption over the Integers. In: Krawczyk, H. (ed.) Public-Key Cryptography - PKC 2014. Lecture Notes in Computer Science, vol. 8383, pp. 311–328. Springer (2014)
4. Doröz, Y., Hu, Y., Sunar, B.: Homomorphic AES Evaluation using NTRU. IACR Cryptology ePrint Archive 2014, 39 (2014), `http://eprint.iacr.org/2014/039`
5. Ekdahl, P., Johansson, T.: SNOW - A new stream cipher (2000)
6. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009. pp. 169–178. ACM (2009)
7. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 850–867. Springer (2012)
8. Hawkes, P., Rose, G.G.: Exploiting Multiples of the Connection Polynomial in Word-Oriented Stream Ciphers. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976, pp. 303–316. Springer (2000)
9. Knuth, D.E.: The Art of Computer Programming, Volume II: Seminumerical Algorithms. Addison-Wesley (1969)
10. Lidl, R., Niederreiter, H.: Finite fields. Cambridge university press (1983)
11. Méaux, P.: Symmetric Encryption Scheme adapted to Fully Homomorphic Encryption Scheme. In: Journées Codage et Cryptographie - JC2 2015 - 12ème édition des Journées Codage et Cryptographie du GT C2, 5 au 9 octobre 2015, La Londe-les-Maures, France (2015), `http://imath.univ-tln.fr/C2/`
12. Méaux, P., Journault, A., Standaert, F., Carlet, C.: Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 311–343. Springer (2016), full version available at `http://eprint.iacr.org/2016/254`
13. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. personal communication (October 2015)

A Possible Trade-Offs

A.1 FLIP (47,40,105)

Table 3. Log of the complexities of the attacks as function of the number of initial guesses (ℓ) for the instantiation FLIP (47,40,105)

ℓ	$\mathbb{P}_\ell$	v_ℓ	$\mathbb{P}_{rg}$	C_D	C_T	C_M
12	-26.335	13.992	-12.528	40.326	54.503	27.983
13	-23.049	13.976	-13.627	37.025	55.554	27.951
14	-20.653	13.960	-14.736	34.613	56.615	27.919
15	-18.738	13.943	-15.854	32.682	57.684	27.887
16	-17.141	13.927	-16.982	31.069	58.763	27.854
17	-15.775	13.911	-18.120	29.686	59.852	27.821
18	-14.585	13.894	-19.267	28.480	60.950	27.788
19	-13.536	13.878	-20.425	27.414	62.057	27.755
20	-12.601	13.861	-21.592	26.462	63.175	27.722
21	-11.762	13.844	-22.771	25.606	64.303	27.688
22	-11.004	13.827	-23.960	24.831	65.442	27.654
23	-10.315	13.810	-25.160	24.125	66.591	27.621
24	-9.686	13.793	-26.371	23.479	67.750	27.586
25	-9.110	13.776	-27.593	22.886	68.921	27.552
26	-8.580	13.759	-28.827	22.339	70.103	27.517
27	-8.092	13.741	-30.073	21.833	71.297	27.483
28	-7.640	13.724	-31.331	21.364	72.502	27.448
29	-7.221	13.706	-32.601	20.927	73.720	27.413
30	-6.832	13.689	-33.883	20.520	74.949	27.377
31	-6.469	13.671	-35.179	20.140	76.191	27.342
32	-6.131	13.653	-36.487	19.784	77.446	27.306
33	-5.816	13.635	-37.809	19.450	78.714	27.270
34	**-5.520**	**13.617**	**-39.145**	**19.137**	**79.995**	**27.233**
> 34	...	...	...	...	> 80	...
35	-5.243	13.598	-40.495	18.842	81.290	27.197

A.2 FLIP (87,82,231)

Table 4. Log of the complexities of the attacks as function of the number of initial guesses (ℓ) for the instantiation FLIP (87,82,231)

ℓ	$\mathbb{P}_\ell$	v_ℓ	$\mathbb{P}_{rg}$	C_D	C_T	C_M
19	-42.382	16.151	-19.647	58.533	68.100	32.302
20	-38.522	16.144	-20.721	54.666	69.151	32.287
21	-35.589	16.136	-21.799	51.725	70.206	32.272
22	-33.169	16.128	-22.881	49.298	71.266	32.257
23	-31.097	16.121	-23.967	47.218	72.329	32.241
24	-29.282	16.113	-25.058	45.395	73.397	32.226
25	-27.667	16.105	-26.153	43.772	74.469	32.211
26	-26.214	16.098	-27.253	42.311	75.546	32.195
27	-24.895	16.090	-28.357	40.985	76.627	32.180
28	-23.691	16.082	-29.465	39.773	77.712	32.164
29	-22.584	16.074	-30.578	38.658	78.802	32.149
30	-21.562	16.067	-31.696	37.629	79.896	32.133
31	-20.615	16.059	-32.818	36.674	80.994	32.118
32	-19.734	16.051	-33.944	35.785	82.097	32.102
33	-18.912	16.043	-35.075	34.955	83.205	32.086
34	-18.142	16.035	-36.211	34.178	84.317	32.071
35	-17.421	16.027	-37.352	33.448	85.434	32.055
36	-16.743	16.020	-38.497	32.762	86.556	32.039
37	-16.104	16.012	-39.648	32.116	87.683	32.023
38	-15.502	16.004	-40.803	31.505	88.814	32.007
39	-14.932	15.996	-41.963	30.928	89.950	31.991
40	-14.393	15.988	-43.128	30.381	91.091	31.975
41	-13.883	15.980	-44.298	29.862	92.237	31.959
42	-13.398	15.972	-45.473	29.370	93.388	31.943
43	-12.937	15.964	-46.653	28.901	94.543	31.927
44	-12.499	15.956	-47.838	28.455	95.704	31.911
45	-12.082	15.947	-49.028	28.029	96.870	31.895
46	-11.684	15.939	-50.224	27.624	98.042	31.879
47	-11.305	15.931	-51.425	27.236	99.218	31.862
48	-10.942	15.923	-52.631	26.865	100.400	31.846
49	-10.596	15.915	-53.842	26.511	101.586	31.830

Table 5. Log of the complexities of the attacks as function of the number of initial guesses (ℓ) for the instantiation FLIP (87,82,231)

ℓ	$\mathbb{P}_\ell$	v_ℓ	$\mathbb{P}_{rg}$	C_D	C_T	C_M
50	-10.265	15.907	-55.059	26.171	102.779	31.813
51	-9.948	15.898	-56.282	25.846	103.976	31.797
52	-9.644	15.890	-57.509	25.534	105.180	31.780
53	-9.353	15.882	-58.743	25.235	106.388	31.763
54	-9.074	15.873	-59.982	24.947	107.602	31.747
55	-8.806	15.865	-61.227	24.671	108.822	31.730
56	-8.548	15.857	-62.477	24.405	110.048	31.713
57	-8.301	15.848	-63.734	24.149	111.279	31.697
58	-8.063	15.840	-64.996	23.903	112.516	31.680
59	-7.835	15.831	-66.264	23.666	113.758	31.663
60	-7.614	15.823	-67.538	23.437	115.007	31.646
61	-7.402	15.815	-68.818	23.217	116.262	31.629
62	-7.198	15.806	-70.104	23.004	117.522	31.612
63	-7.001	15.797	-71.397	22.799	118.789	31.595
64	-6.812	15.789	-72.695	22.601	120.062	31.578
65	-6.629	15.780	-74.000	22.409	121.341	31.561
66	-6.452	15.772	-75.311	22.224	122.627	31.543
67	-6.281	15.763	-76.629	22.044	123.918	31.526
68	-6.116	15.754	-77.953	21.871	125.216	31.509
69	-5.957	15.746	-79.284	21.703	126.521	31.491
70	**-5.803**	**15.737**	**-80.621**	**21.540**	**127.832**	**31.474**
> 70	...	...	...	...	> 128	...
71	-5.655	15.728	-81.965	21.383	129.150	31.457

B Complexities of the Attack on the Toy Version of FLIP

Table 6. Log of the complexities of the attacks as function of the number of initial guesses (ℓ) for the toy version FLIP (14,14,36)

ℓ	$\mathbb{P}_\ell$	v_ℓ	$\mathbb{P}_{rg}$	C_D	C_T	C_M
6	-11.861	10.741	-6.370	22.601	38.592	21.481
7	-9.436	10.691	-7.528	20.126	39.601	21.382
8	-7.814	10.640	-8.717	18.454	40.638	21.280
9	-6.602	10.589	-9.939	17.191	41.706	21.177
10	-5.649	10.536	-11.197	16.185	42.806	21.072
11	-4.876	10.483	-12.493	15.359	43.941	20.966
12	-4.235	10.428	-13.828	14.663	45.113	20.857
13	-3.696	10.373	-15.207	14.069	46.325	20.746
14	-3.237	10.316	-16.631	13.553	47.580	20.633
15	-2.843	10.259	-18.105	13.102	48.881	20.517
16	-2.503	10.200	-19.632	12.703	50.231	20.399
17	-2.208	10.140	-21.217	12.347	51.636	20.279
18	-1.950	10.078	-22.865	12.028	53.100	20.156
19	-1.723	10.015	-24.581	11.739	54.628	20.031
20	-1.524	9.951	-26.373	11.476	56.227	19.903
21	-1.349	9.886	-28.247	11.235	57.904	19.771
22	-1.194	9.819	-30.214	11.013	59.670	19.637
23	-1.057	9.750	-32.284	10.807	61.534	19.500
24	**-0.935**	**9.679**	**-34.472**	**10.615**	**63.510**	**19.359**
> 24	...	...	...	...	> 64	...
25	-0.827	9.607	-36.794	10.435	65.616	19.215