Noname manuscript No.
(will be inserted by the editor)

Multi-user Security Bound for Filter Permutators in the Random Oracle Model[⋆]

Benoît Cogliati · Titouan Tanguy

Received: date / Accepted: date

Abstract At EUROCRYPT 2016, Méaux *et al.* introduced a new design strategy for symmetric ciphers for Fully Homomorphic Encryption (FHE), which they dubbed filter permutators. Although less efficient than classical stream ciphers, when used in conjunction with an adequate FHE scheme, they allow constant and small noise growth when homomorphically evaluating decryption circuit. In this article, we present a security proof up to the birthday bound (with respect to the size of the IV and the size of the key space) for this new structure in the random oracle model and in the multi-user setting. In particular, this result justifies the theoretical soundness of filter permutators. We also provide a related-key attack against all instances of FLIP, a stream cipher based on this design.

Keywords Filter Permutators · FLIP · Related-Key attacks · Random Oracle Model · Provable Security

Mathematics Subject Classification (2010) 94A60

1 Introduction

Fully Homomorphic Encryption. Fully Homomorphic Encryption (FHE) allows a user to delegate computations to a Cloud service, without compromising the privacy of his data. Since Gentry's initial breakthrough [Gen09], researchers have been trying to improve the efficiency of FHE in order to allow more complex applications, some of which are listed in [NLV11]. Many bottlenecks still remain, as FHE algorithms require important computational and memory costs, both on the user and the server side. Moreover, almost every existing FHE scheme

[⋆] This is a post-peer-review, pre-copyedit version of an article published in Designs, Codes and Cryptography. The final authenticated version is available online at: http://dx.doi.org/10.1007/s10623-018-0555-5

University of Luxembourg, Luxembourg
E-mail: benoitcogliati@hotmail.fr E-mail: titouan.tanguy@gmail.com

have limited homomorphic capabilities: each operation increases the noise level of the involved ciphertexts, preventing decryption if too many operations are performed. Most FHE schemes offer some bootstrapping capabilities in order to decrease noise levels, however at great computational costs. This step can sometimes be avoided, for example in the case leveled Fully Homomorphic Encryption [BGV12, CNT12], at the expense of having to know in advance a (polynomial) upper bound on the multiplicative depth of the evaluated circuit.

Hybrid Framework. In order to alleviate the costs on the user side, a hybrid framework has been designed. A typical application scenario consists of a combination of five steps involving a FHE scheme H and a symmetric key algorithm S as summarized below:

1. *Initialization.* The user (Alice) runs both key generation algorithms for H and S and sends the Cloud (Bob) her homomorphic public key pk^H and the homomorphic ciphertext of her symmetric key $C^H(\mathsf{sk}_i^S)$;
2. *Storage.* Alice encrypts her data m_i using S and sends Bob the ciphertexts $C^S(m_i)$;
3. *Evaluation.* Bob homomorphically evaluates the decryption circuit of S on Alice's data and gets the ciphertexts $C^H(m_i)$;
4. *Computation.* Bob homomorphically evaluates some function f on Alice's encrypted data;
5. *Result.* Bob sends the compressed encrypted result of the computation $c^H(f(m_i))$, and Alice decrypts it.

In such a framework, the user is only required to evaluate the FHE scheme on the symmetric key bits and the results of Bob's computation, since the encryption of her data uses a typically much faster symmetric primitive. However, Bob now needs to evaluate a symmetric decryption circuit before being able to actually exploit Alice's data. This additional step will increase noise level in Alice's ciphertexts, effectively reducing homomorphic capability of the scheme.

In order to mitigate the impact of this new step, several works have tried to reduce the homomorphic cost of evaluating the symmetric decryption algorithm. Traditional block ciphers like AES were considered for this task [GHS12, CLT14], culminating with the best performance result: 120 AES decryptions in 4 minutes. Since traditional block ciphers offer constant but high noise growth, several new symmetric schemes have been purposefully built for FHE, most notably the block cipher LowMC [ARS+15] and the stream ciphers Trivium and Kreyvium [CCF+16]. These two constructions led to reduced multiplicative depths for the decryption algorithm, but were still limited by important drawbacks. More specifically, LowMC, as a block cipher, offers constant noise growth that is still large enough to significantly hamper homomorphic capability. As for Kreyvium, the error level actually grows with number of decryptions, which implies that, given a large enough data set, ciphertexts will become too noisy to be decrypted and will either require a costly bootstrapping operation or a re-initialization of the stream cipher.

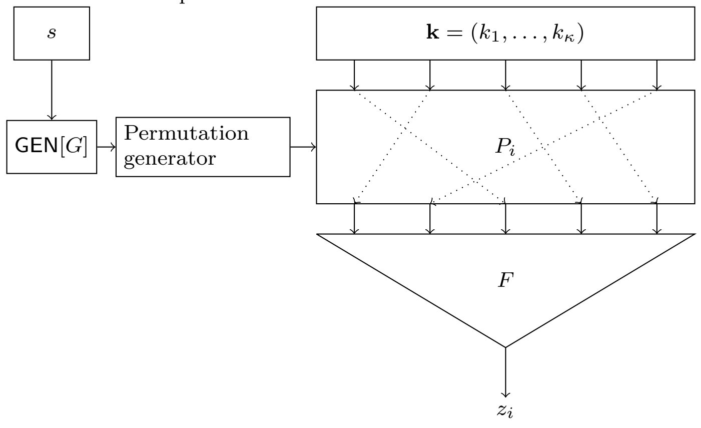
The FLIP Family of Stream Ciphers. At EUROCRYPT 2016, Méaux et al. [MJSC16] proposed a new family of stream ciphers dubbed FLIP which aims at overcoming the drawbacks of LowMC and Kreyvium. In particular, FLIP allows for constant and smaller noise growth by the use of a filter permutator, a new construction that is related to a filter generator. The main difference is that the key register is kept constant and is permuted before entering the filtering function. This effectively keeps the noise growth constant, as permutating the key register does not increase noise level of the ciphertexts. The multiplicative depth of the circuit is also limited by the low degree of the filtering function. An oversight in the security analysis of the first set of proposed parameters for FLIP was spotted by Duval *et al.* [DLR16] who designed a guess-and-determine attack against the first version of FLIP. This led to an update of the original article [MJSC16], which now proposes a security analysis of the filtering function with respect to guess-and-determine correlation attacks, higher-order correlation attacks, algebraic attacks and fast algebraic attacks. Unfortunately, this analysis only applies to the filtering function when it takes as input uniformly random words of the size of the key. However, due to the unusual structure of FLIP, the Hamming weight of the key register is constant. Thus evaluating the security of the filtering function requires a deeper analysis of the behavior of Boolean functions on restricted input sets which was started in [CMR17].

Our Contribution. Our contribution is twofold. First we realize the first analysis of the generic structure of filter permutators, upon which FLIP is built. A filter permutator generates a key stream by applying a filter to a key vector. A uniformly random IV s is used to initialize a stateful pseudorandom bit generator whose output is fed to a permutation generator. At each round, the components of the key vector $\mathbf{k}$ are permuted using fresh outputs of the permutation generator, and the filter is then applied to the new value of the key vector to generate an output. Figure 1 illustrates this structure. We analyze the security of filter permutators in the multi-user, known IV setting when the filter and the pseudorandom bit generator (PRG) are modeled as public random functions. In particular we prove that the advantage of a distinguisher against the security of a filter permutator following the design of FLIP is negligible as long as the number q_* of adversarial queries to the construction, the filter and the pseudorandom bit generator satisfies $q_* \ll \min\left(2^{n/2}/\kappa, 2^{b+1}/\kappa^2, \sqrt{|\mathcal{K}|}\right)$ where n is the size of the IV, b the output size of the stateful pseudorandom bit generator, $\mathcal{K}$ is the key space, κ the number of components of the key vectors. We also describe and provide security bounds for two specific choices of key spaces for filter permutators over a finite field $\mathbb{F}_p$:

– if the prime p is small, we can choose the key space of all κ-tuples containing an equal number of each element from $\mathbb{F}_p$;
– if the prime p is large, we simply set the key space to $(\mathbb{F}_p)^\kappa$.

These constructions generalize the high-level structure of FLIP to every finite field, and could prove interesting in applications. Indeed, some FHE schemes

Fig. 1 Structure of a filter permutator



are built over some finite field of odd characteristic. In this case, it would be preferable to rely on a symmetric-key algorithm whose output is in the same field to keep the amount of required computation to a minimum. We leave it as an interesting open problem to define and analyze filters for odd characteristic.

Second we describe a chosen-IV related-key attack on the FLIP family of stream ciphers. This attack relies on the combination of an algebraic attack combined with related internal states (see e.g [ALP05] for other attacks following this strategy) and allows key recovery with overwhelming probability using at most 75 known bits of key stream and one related-key query per key bit. Since filter permutators are most likely used in a hybrid FHE framework, such a related key attack seems especially devastating when used by a malicious server. However, its impact can be mitigated by the fact that a malicious server alone would not be able to directly apply it, unless it is also able to break the FHE scheme or it relies on the behavior of the user. In the former case, the server would not need to attack FLIP, while in the latter case, it could already use software fault injection attacks [CGG16] to recover the key bits.

Discussion on the model. In this paper, we model the filter and PRG as random oracles (i.e. public uniformly random functions). This strong abstraction deviates from the design of a real filter permutator (e.g. FLIP), which implies limitations on the usefulness of our result. Indeed, our bounds will only provide security guarantees against generic adversaries[1], while most cryptanalysis will focus on specific weaknesses of the filter. However, our results indicate that filter permutators, as a generic structure, are sound. This suggests that, even if the FLIP ciphers were proven to be insecure, it may still be possible to design a secure instantiation of filter permutators by strengthening the filter function.

[1] In a sense, this model amounts to restrict the adversarial knowledge of the PRG and filter to the (black-box) queries it issued.

Organization. In section 2 we present general notation and security definitions. Section 3 is devoted to the study of the generic filter permutator structure. In section 4 we discuss security parameters for the structure of the FLIP family of stream ciphers and describe our related-key attack.

2 Preliminaries

2.1 Notation and General Definitions

General Notation. In all the following, we fix an integer $n \geq 1$. The set of all permutations of $\{0,1\}^n$ will be denoted $\mathrm{Perm}(n)$. Given a non-empty set X, we denote by $x \leftarrow_{\$} X$ the draw of an element x from X uniformly at random and by $\mathrm{Perm}(X)$ the set of all permutations of X.

Let $m \geq 1$ be an integer and let E be any non-empty set. We sometimes write $[m]$ for the set $\{1,\ldots,m\}$. For any permutation $P \in \mathrm{Perm}([m])$ and any vector $\mathbf{z} = (z_1,\ldots,z_m) \in E^m$, we denote by $P(\mathbf{z})$ the permuted vector $(z_{P(1)},\ldots,z_{P(m)})$. For any vector $\mathbf{z} \in E^m$, we denote by C_{E^m} the set of all $P \in \mathrm{Perm}([m])$ such that $P(\mathbf{z}) = \mathbf{z}$. We also denote by $\overline{C_{E^m}}$ the average over E^m of C_{E^m}. A subset $E' \subset E^m$ will be said *stable* if, for every vector $\mathbf{z} \in E'$ and any permutation $P \in \mathrm{Perm}([m])$, $P(\mathbf{z}) \in E'$.

Pseudorandom Bit Generators. A standard *pseudorandom bit generator* (PRG) is a function $G : \{0,1\}^s \rightarrow \{0,1\}^{b+s}$ that takes as input a s-bit seed and returns a longer $(b+s)$-bit string. We recall the classical (computational) security definition for PRGs from [BY01]. Let $\mathcal{D}$ be a distinguishing algorithm that given a $b+s$ bit string returns a bit. We let

$$\mathbf{Adv}_G^{\mathrm{PRG}}(\mathcal{D}) = \left|\Pr\left[D^G \rightarrow 1\right] - \Pr\left[D^{\$} \rightarrow 1\right]\right|,$$
$$\mathbf{Adv}_G^{\mathrm{PRG}}(t) = \max_{\mathcal{D}}\left(\mathbf{Adv}_G^{\mathrm{PRG}}(\mathcal{D})\right),$$

where $\Pr\left[D^G \rightarrow 1\right]$ (resp. $\Pr\left[D^{\$} \rightarrow 1\right]$) denotes the probability that D outputs 1 when it receives string $G(x)$ for a uniformly random $x \in \{0,1\}^s$ (resp. a uniformly random string from $\{0,1\}^{b+s}$) and the maximum is taken over all adversaries running in time at most t.

One can for example create two pseudorandom generators G_1 and G_2 respectively based on the AES_{128} and AES_{256} block cipher as follows:

$$\begin{aligned}
G_1 \;:\; & \{0,1\}^{128} \rightarrow \{0,1\}^{256} \\
& \qquad x \mapsto \mathsf{AES}_{128}(x,0)||\mathsf{AES}_{128}(x,1), \\
G_2 \;:\; & \{0,1\}^{256} \rightarrow \{0,1\}^{512} \\
& \qquad x \mapsto \mathsf{AES}_{256}(x,0)||\mathsf{AES}_{256}(x,1)||\mathsf{AES}_{256}(x,2)||\mathsf{AES}_{256}(x,3).
\end{aligned}$$

Then, for any non-negative integer t, one has

$$\mathbf{Adv}_{G_1}^{\mathrm{PRG}}(t) \leq \mathrm{Adv}_{\mathsf{AES}_{128}}^{\mathrm{PRF}}(2,t),$$

$$\mathbf{Adv}_{G_2}^{\mathrm{PRG}}(t) \leq \mathrm{Adv}_{\mathsf{AES}_{256}}^{\mathrm{PRF}}(4, t),$$

where $\mathrm{Adv}_{\mathsf{AES}_k}^{\mathrm{PRF}}(q, t)$ denotes the maximum advantage of an adversary trying to distinguish AES_k from a uniformly random function in time at most t and using at most q queries.

These notions are usual when considering security in the standard, computational model. In our case, the PRG will be used to derive a sequence of pseudorandom outputs based on a *public* value. Hence, we will not be able to directly use such a security definition. However, since we are aiming at a purely information-theoretic security proof in the random oracle model, we are going to also model the PRG by a public uniformly random function.

Stateful Generators. We also adapt the notion of stateful generator from [BY01] to our context. A *stateful pseudorandom bit generator* (sPRG)

$$\mathsf{GEN}[G] = (\mathsf{GEN}[G].\mathsf{init}, \mathsf{GEN}[G].\mathsf{next}, b, n)$$

is specified by a pair of algorithms and a pair of positive integers. The *initialization* algorithm $\mathsf{GEN}[G].\mathsf{init}$ takes as input a n-bit string and uses it to initialize the state of the generator. The *next step* algorithm, given the current state, returns a pair consisting of a b-bit output block and the next state of the generator. This generation process is application controlled. Whenever the algorithm relying on $\mathsf{GEN}[G]$ needs a new block of pseudorandom bits, it simply calls the algorithm $\mathsf{GEN}[G].\mathsf{next}$ to get the next output. It is a well-known fact (see [BY01, Theorem 2.3]) that, in the standard model, a secure pseudorandom bit generator $G : \{0,1\}^n \to \{0,1\}^{n+b}$ can be turned into a secure stateful pseudorandom bit generator $\mathsf{GEN}[G]$ as follows. The algorithm $\mathsf{GEN}[G].\mathsf{init}$ takes as input a n-bit string and returns the same string. The algorithm $\mathsf{GEN}[G].\mathsf{next}$ takes as input a n-bit string s, computes $G(s)$ and returns the first b bits (which we denote $G^{\mathsf{out}}(s)$) as output block and the last n bits (which we denote $G^{\mathsf{st}}(s)$) as the new state. The authors of [MJSC16] suggest this construction with the aforementioned PRG based on the AES blockcipher. That is why we are also going to rely on this construction, albeit using an underlying idealized PRG (that we model as a random oracle).

3 Security Analysis of Filter Permutators

3.1 High-level structure

For the remaining of the section, we fix three integers $b, n, \kappa \geq 1$, two non-empty sets A, B, a non-empty stable set $\mathcal{K} \subset A^\kappa$ and a Stateful Pseudorandom bit Generator $\mathsf{GEN}[G] = (\mathsf{GEN}[G].\mathsf{init}, \mathsf{GEN}[G].\mathsf{next}, n, b)$ based on an underlying Pseudorandom bit Generator $G : \{0,1\}^n \to \{0,1\}^{n+b}$. Then, for any filtering function $F : A^\kappa \longmapsto B$, we define the *Filter Permutator* $\mathsf{FP}_{[\mathsf{GEN}[G],F]}$. Given a key $\mathbf{k} = (k_1, \ldots, k_\kappa) \in \mathcal{K}$ and a public IV $s \in \{0,1\}^n$, it generates a key stream $\mathbf{z} = (z_1, \ldots, z_{q_*}) = \mathsf{FP}_{[\mathsf{GEN}[G],F]}(\mathbf{k}, s, q_*)$ as follows:

1. the IV s is used to initialize the state of the sPRG with $\mathsf{GEN}[G].\mathsf{init}$;
2. for each $i = 1, \ldots, q_*$,
   (a) the next κ outputs of $\mathsf{GEN}[G].\mathsf{next}$ are given as input to a permutation generator based on the Fisher-Yates shuffle, as per Algorithm 1, which shuffles the κ key variables by generating a permutation P_i;
   (b) the filtering function F is then applied to the permuted key to generate z_i.

Figure 1 illustrates this high-level structure.

Algorithm 1 Fisher-Yates shuffle

```
1: procedure SHUFFLE($a$)                          ▷ The array to be shuffled
2:     for i=$\kappa$; i>1; i-=1 do
3:         $j \leftarrow (\mathsf{GEN}[G].\mathsf{next}()\ \mathsf{mod}\ i) + 1$
4:         Exchange $a[j]$ and $a[i]$
5:     return $a$                                  ▷ the shuffled array
```

3.2 Security Notions and Statement of the Result

Our main result is a bound on the multi-user security of filter permutators, when the filtering function F and the PRG are modeled as uniformly random public functions. In particular, adversaries are computationally unbounded. They are only limited in the number of queries they can issue to their different oracles. This model amounts to giving an upper bound on the adversarial knowledge about the underlying filter and PRG.

Let us fix a number l of users. An Adversary $\mathcal{D}$ is then modeled as an algorithm which interacts with $l + 2$ oracles that we denote generically $(F, G, F_1, \ldots, F_l)$ where syntactically F (resp. G) are functions with input space A^κ (resp. $\{0,1\}^n$) and output space B (resp. $\{0,1\}^{n+b}$), and $F_1, \ldots, F_l$ take as input an integer j and output a tuple $(s, \mathbf{z}) \in \{0,1\}^n \times B^j$. The goal of $\mathcal{D}$ is to distinguish two worlds: the so-called *real world* where

- $F : A^\kappa \to B$ and $G : \{0,1\}^n \to \{0,1\}^{n+b}$ are uniformly random functions,
- for $i = 1, \ldots, l$ and any integer j,

$$F_i(j) = \mathsf{FP}^{\$}_{[\mathsf{GEN}[G],F,\mathbf{k}_i]}(j) \overset{\text{def}}{=} \big(s, \mathsf{FP}_{[\mathsf{GEN}[G],F]}(\mathbf{k}_i, s, j)\big) \in \{0,1\}^n \times B^j,$$

  where s is a fresh uniformly random IV, and the keys $\mathbf{k}_i$ for $i = 1, \ldots, l$ are drawn uniformly and independently at random in $\mathcal{K}$, independently from F;

and the so-called *ideal world* where

- $F : A^\kappa \to B$ and $G : \{0,1\}^n \to \{0,1\}^{n+b}$ are uniformly random functions,
- for $i = 1, \ldots, l$ and any integer j, $F_i(j) = (s, \mathbf{z})$ is drawn uniformly at random in $\{0,1\}^n \times B^j$, independently from F and every preceding query.

We will refer to F as the filtering oracle, to G as the PRG oracle and to F_i as the i-th construction oracle, for $i = 1, \ldots, l$.

The distinguishing advantage of a distinguisher $\mathcal{D}$ is defined as

$$\mathbf{Adv}(\mathcal{D}) \overset{\text{def}}{=} \left| \Pr\left[\mathcal{D}^{F,G,\mathsf{FP}^{\$}_{[\mathsf{GEN}[G],F,\mathbf{k}_1]},\ldots,\mathsf{FP}^{\$}_{[\mathsf{GEN}[G],F,\mathbf{k}_l]}} \to 1\right] - \Pr\left[\mathcal{D}^{F,G,F_1,\ldots,F_l} \to 1\right] \right|,$$

where the first probability is taken over the random choice of the keys $\mathbf{k}_1, \ldots, \mathbf{k}_l$, F, G and the IVs, and the second probability is taken over the random choice of F, G, and the random draw of the outputs of $F_1, \ldots, F_l$. In all the following, we consider computationally unbounded distinguishers, and we can thus assume that they are deterministic.

For non-negative integers q_F, q_G, q_*, we define

$$\mathbf{Adv}^{\mathrm{mu-kIV}}_{\mathsf{FP}^{\$}_{[\mathsf{GEN}[G]]}}(q_*, q_F, q_G) = \max_{\mathcal{D}} \mathbf{Adv}(\mathcal{D}),$$

where the maximum is taken over all distinguishers making exactly q_F queries to the filtering oracle, q_G queries to the PRG oracle and getting exactly q_* outputs from the construction oracles[2]. We also assume that they never make pointless queries. In this context, it means that the distinguisher never repeats a query to its filtering oracle or to its PRG oracle, since a construction query always results in a fresh key stream.

Theorem 1 *For any non-negative integers q_F, q_G, q_*, one has*

$$\mathbf{Adv}^{\mathrm{mu-kIV}}_{\mathsf{FP}^{\$}_{[\mathsf{GEN}[G]]}}(q_*, q_F, q_G) \leq \frac{(\kappa-1)q_*\left(q_G + (\kappa-1)q_*\right)}{2^n} + q_*^2 \cdot \frac{\overline{C_{\mathcal{K}}}}{\kappa!} + \frac{q_*\kappa^2}{2^{b+1}} + \frac{(q_F + q_*)q_*}{|\mathcal{K}|}.$$

The proof of this result relies on Patarin's H-coefficients technique [Pat08]. We will first give a brief overview of this technique in section 3.3 and proceed with the proof of Theorem 1 in section 3.4.

3.3 The H-Coefficients Technique

We summarize the interaction of $\mathcal{D}$ with its oracles in what will be referred to as the queries transcript $(\mathcal{Q}_C, \mathcal{Q}_F, \mathcal{Q}_G, \mathcal{Q}'_G)$ of the attack, where $\mathcal{Q}_C$ records the queries to the constructions oracles, $\mathcal{Q}_F$ records the queries to the filtering oracle and $\mathcal{Q}_G$ records the queries to the PRG oracle. More precisely, $\mathcal{Q}_C$ is a multiset containing all tuples

$$(i, q, s, \mathbf{z}) \in \{1, \ldots, l\} \times \{1, \ldots, q_*\} \times \{0,1\}^n \times B^q$$

such that $\mathcal{D}$ queried q outputs to the i-th construction oracle and received IV s and outputs $\mathbf{z}$, $\mathcal{Q}_F$ contains all the pairs

$$(u, v) \in A^{\kappa} \times B$$

[2] Note that IV blocks do not count as output blocks from the construction.

such that $\mathcal{D}$ made the query u to its filtering oracle and received answer v and $\mathcal{Q}_G$ is the *ordered* set of all the pairs

$$(x, y) \in \{0,1\}^n \times \{0,1\}^{b+n}$$

such that $\mathcal{D}$ made the query x to its PRG oracle and received answer y. Note that one has

$$|\mathcal{Q}_F| = q_F, \qquad \sum_{(i,q,s,\mathbf{z})\in\mathcal{Q}_C} q = q_*,$$

since we assume that $\mathcal{D}$ never makes pointless queries.

In order to simplify the description of bad transcripts, we will also reveal additional information to the adversary:

– at the same time the adversary receives an answer from a construction query, we will offer him additional information corresponding to each state of the PRG during the permutation generation step in the real world, along with the corresponding evaluation of G (in the ideal world, we will just mimic the real world); this will be summarized in an additional multiset $\mathcal{Q}'_G$ which will contain pairs $(s, G(s))$ for each state s of the PRG $\mathsf{GEN}[G]$,[3]
– at the end of its interaction with its oracles, but before it output its answer, the actual keys $(\mathbf{k}_i)_{1\leq i\leq l}$ are revealed if we are in the real world, or uniformly random dummy keys $(\mathbf{k}_i)_{1\leq i\leq l}$ drawn independently from the queries transcript are given if we are in the ideal world.

A transcript τ is thus defined as $\tau = (\mathcal{Q}_C, \mathcal{Q}_F, \mathcal{Q}_G, \mathcal{Q}'_G, \overline{\mathbf{k}})$ with $\overline{\mathbf{k}} = (\mathbf{k}_i)_{1\leq i\leq l}$ and a transcript will be *attainable* with respect to a fixed distinguisher $\mathcal{D}$ if and only if its associated queries transcript is attainable. We also denote Θ the set of attainable transcripts. As usual, we denote T_{re}, resp. T_{id} the probability distribution of the transcript τ induced by the real world, resp. the ideal world, and by extension we use the same notation for random variables distributed according to each distribution. The main lemma of the H-coefficient technique is the following one.

Lemma 1 ([Pat08]) *Fix a distinguisher $\mathcal{D}$. Let $\Theta = \Theta_{good} \bigcup \Theta_{bad}$ be a partition of the attainable transcripts. Assume that there exists ϵ_1 such that for any $\tau \in \Theta_{good}$ one has*

$$\frac{Pr[T_{re} = \tau]}{Pr[T_{id} = \tau]} \geq 1 - \epsilon_1$$

and that there exists ϵ_2 such that

$$Pr[T_{id} \in \Theta_{bad}] \leq \epsilon_2$$

Then $\mathbf{Adv}(\mathcal{D}) \leq \epsilon_1 + \epsilon_2$.

This result is classical and proofs can be found in e.g. [CS14, CLL+14].

[3] Note that, in the ideal world, construction oracles outputs do not depend on these values. Also note that states of the stateful PRG can repeat and collide with PRG queries from the adversary. That is why we record them in a separate multiset. Obviously, such collisions are unwanted and will be our first bad event.

3.4 Proof of Theorem 1

In this section, we prove Theorem 1 using the H-coefficients technique.

Let q_F, q_G, q_* be three non-negative integers and let $\mathcal{D}$ be any (q_F, q_G, q_*)-distinguisher against the multi-user security of $\mathsf{FP}_{\mathsf{GEN}[G],F}$ in the known IV setting, where F and G are uniformly random public functions. We will denote δ_j, for $j = 1, \ldots, l$, the total number of queries the adversary made to its j-th construction oracle and $\mathcal{Q}_{C,j} = \{(i_1, q_1, s_1, \mathbf{z}_1), \ldots, (i_{\delta_j}, q_{\delta_j}, s_{\delta_j}, \mathbf{z}_{\delta_j})\}$ the subset of $\mathcal{Q}_C$ consisting in all the construction queries to the j-th oracle (note that this is a multiset which is ordered with an arbitrary ordering). We also denote $P_{s,i}$ the i-th permutation generated by Algorithm 1, given the outputs of $\mathsf{GEN}[G]$ initiated with IV s.

As usual, our first step will be the definition of a set of bad transcripts. Then we will lower bound the probability to obtain a bad transcript in the ideal world in Lemma 2, and prove that good transcripts occur with the same probability in both the real and the ideal world in Lemma 3. Theorem 1 will follow by combining Lemma 1 with Lemmas 2 and 3 below.

Definition 1 We say that an attainable transcript $\tau = (\mathcal{Q}_C, \mathcal{Q}_F, \mathcal{Q}_G, \mathcal{Q}'_G, \overline{\mathbf{k}})$ is bad if one of the following conditions if fulfilled:

(C-1) there exists a query $(x, y) \in \mathcal{Q}'_G$ such that:

$$\begin{cases} \text{there exists a query } (x', y') \in \mathcal{Q}_G \text{ such that } x = x' \\ \qquad \text{or} \\ \text{there exists another query } (x'', y'') \in \mathcal{Q}'_G \text{ such that } x = x''; \end{cases}$$

(C-2) there exists $1 \le i \le l$ and two distinct tuples $(j_1, j_2) \in \{1, \ldots, \delta_i\} \times \{1, \ldots, q_{j_1}\}$, $(j'_1, j'_2) \in \{1, \ldots, \delta_i\} \times \{1, \ldots, q_{j'_1}\}$ such that $P_{s_{j_1}, j_2}(\mathbf{k}_i) = P_{s_{j'_1}, j'_2}(\mathbf{k}_i)$;

(C-3) there exists $1 \le i < i' \le l$ and two tuples $((i, q, s, \mathbf{z}), j) \in \mathcal{Q}_{C,i} \times \{1, \ldots, q\}$, $((i', q', s', \mathbf{z}'), j') \in \mathcal{Q}_{C,i'} \times \{1, \ldots, q'\}$ such that $P_{s,j}(\mathbf{k}_i) = P_{s',j'}(\mathbf{k}_{i'})$;

(C-4) there exists $(i, q, s, \mathbf{z}) \in \mathcal{Q}_C$, $1 \le j \le q$ and $(u, v) \in \mathcal{Q}_F$ such that $P_{s,j}(\mathbf{k}_i) = u$.

We denote Θ_{bad} the set of bad transcripts, and $\Theta_{good} = \Theta \setminus \Theta_{bad}$.

Remark 1 In the ideal world, the construction oracles give answers that are independent from G and from F. This can allow an attacker to distinguish between the two worlds in the case where the permutation generation step results in a collision between inputs of F in the real world, since in that case the output of the construction oracle is already fixed by a previous query. Conditions (C-2), (C-3) and (C-4) are here to ensure that no such collision occurs. In the worst case, a state collision between two queries to the same construction oracle could result in the same keystream in the real world, and to uniformly random and independent outputs in the ideal world. Such a case is taken care of in condition (C-1). This also implies that the birthday bound (with respect to the key and the IV size) is the best security bound we can possibly achieve for filter permutators.

We first upper bound the probability to get a bad transcript in the ideal world.

Lemma 2 *One has*

$$Pr[T_{id} \in \Theta_{bad}] \leq \frac{(\kappa-1)q_* \left(q_G + (\kappa-1)q_*\right)}{2^n} + q_*^2 \cdot \frac{\overline{C_{\mathcal{K}}}}{\kappa!} + \frac{q_*\kappa^2}{2^{b+1}} + \frac{(q_F + q_*)q_*}{|\mathcal{K}|}.$$

Proof We denote Θ_i the set of attainable transcripts that satisfy condition (C-i). We are going to lower bound the probability of obtaining a bad transcript as follows:

$$\begin{aligned}\Pr\left[T_{\text{id}} \in \Theta_{bad}\right] \leq & \Pr\left[T_{\text{id}} \in \Theta_1\right] + \Pr\left[T_{\text{id}} \in \Theta_2 | T_{\text{id}} \notin \Theta_1\right] \\ & + \Pr\left[T_{\text{id}} \in \Theta_3\right] + \Pr\left[T_{\text{id}} \in \Theta_4\right].\end{aligned}$$

Condition (C-1). Let $(x, y) \in \mathcal{Q}'_G$. We are going to compute the probability that it is the *first* query such that there exists a query $(\alpha, \beta) \in \mathcal{Q}_G$ such that $x = \alpha$ or there exists a *previous* query $(\alpha, \beta) \in \mathcal{Q}'_G$ such that $x = \alpha$. First note that, since we assume that the adversary never makes pointless queries, the first event can only occur if the query $(\alpha, \beta) \in \mathcal{Q}_G$ occurred before the query (x, y). Now recall that the value x can either be a freshly drawn IV, or can consist in the last n bits of the output of the previous query. Moreover, we have $|\mathcal{Q}'_G| = (\kappa - 1)q_*$. In the first case, the probability that either type of collision occur is smaller than $(q_G + (\kappa - 1)q_*)/2^n$. In the second case, since (x, y) is the first query from $\mathcal{Q}'_G$ which is involved in a collision, the output of the previous query is perfectly random and the probability that a collision occurs is also smaller than $(q_G + q_*(\kappa - 1))/2^n$. Summing over every query from $\mathcal{Q}'_G$ gives

$$\Pr\left[T_{\text{id}} \in \Theta_1\right] \leq \frac{(\kappa-1)q_* \left(q_G + (\kappa-1)q_*\right)}{2^n}.$$

Condition (C-2). Let us denote $v_{1+j+(i-1)(\kappa-1)}$ the $j-th$ output of $\mathsf{GEN}[G]$ that was used to generate the i-th permutation, for $1 \leq i \leq q_*$, $0 \leq j \leq \kappa - 2$. Let us denote U the event

$$v_{1+j+(i-1)(\kappa-1)} < (\kappa - j)\lfloor 2^b/(\kappa - j)\rfloor$$

for every $1 \leq i \leq q_*$, $0 \leq j \leq \kappa - 2$. Then one has

$$\begin{aligned}\Pr\left[T_{\text{id}} \in \Theta_2 | T_{\text{id}} \notin \Theta_1\right] = & \Pr\left[(T_{\text{id}} \in \Theta_2) \wedge U | T_{\text{id}} \notin \Theta_1\right] \\ & + \Pr\left[(T_{\text{id}} \in \Theta_2) \wedge \overline{U} | T_{\text{id}} \notin \Theta_1\right] \qquad (1) \\ \leq & \Pr\left[T_{\text{id}} \in \Theta_2 | U \wedge (T_{\text{id}} \notin \Theta_1)\right] + \Pr\left[\overline{U} | T_{\text{id}} \notin \Theta_1\right]. \qquad (2)\end{aligned}$$

Conditioned on the event $T_{\text{id}} \notin \Theta_1$, no collision between the states of the PRG occurs, and no state of the PRG collides with a query to the oracle G. This event only involves the values of the IV and of the last n bits of the queries from $\mathcal{Q}'_G$. This means that, conditioned on $T_{\mathrm{id}} \notin \Theta_1$, the first b bits of the outputs of the queries from $\mathcal{Q}'_G$, which correspond to the outputs of $\mathsf{GEN}[G]$, are uniformly random and independent messages from $\{0,1\}^b$. Thus one has

$$\begin{aligned}
\Pr\left[\overline{U} | T_{\mathrm{id}} \notin \Theta_1\right] \leq & \sum_{i=1}^{q_*} \sum_{j=0}^{\kappa-2} \Pr\left[v_{j+1+(i-1)(\kappa-1)} \geq (\kappa-j)\lfloor 2^b/(\kappa-j) \rfloor\right] \\
\leq & \sum_{i=1}^{q} \sum_{j=0}^{\kappa-2} \frac{\kappa-j-1}{2^b} = \sum_{i=1}^{q} \sum_{j=1}^{\kappa-1} \frac{j}{2^b} \\
\leq & \frac{q_* \kappa^2}{2^{b+1}}.
\end{aligned} \tag{3}$$

Fact 1. Conditioned on the event $U \wedge (T_{\mathrm{id}} \notin \Theta_1)$, the values $v_{j+1+(i-1)(\kappa-1)}$ are independent and $(v_{1+j+(i-1)(\kappa-1)} \pmod{\kappa - j}) + 1$ will be uniformly random in $\{1, \dots, \kappa - j\}$ for every $1 \leq i \leq q_*$, $0 \leq j \leq \kappa - 2$.

A proof of this fact can be found in the full version of this paper. This implies that, when we condition on the event $(T_{\mathrm{id}} \notin \Theta_1) \wedge U$, the permutations generated by Algorithm 1 are uniformly random and independent permutations of $\{1, \dots, \kappa\}$. Let us now fix $i \in \{1, \dots, l\}$ and two distinct tuples $(j_1, j_2) \in \{1, \dots, \delta_i\} \times \{1, \dots, q_{j_1}\}$, $(j'_1, j'_2) \in \{1, \dots, \delta_i\} \times \{1, \dots, q_{j'_1}\}$. Then, one has

$$\begin{aligned}
&\Pr\left[P_{s_{j_1}, j_2}(\mathbf{k}_i) = P_{s_{j'_1}, j'_2}(\mathbf{k}_i) | U \wedge (T_{\mathrm{id}} \notin \Theta_1)\right] \\
&\quad = \Pr\left[P, P' \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, P(\mathbf{k}_i) = P'(\mathbf{k}_i)\right] \\
&\quad = \sum_{P'' \in \mathrm{Perm}([\kappa])} \Pr\left[P, P' \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, (P(\mathbf{k}_i) = P''(\mathbf{k}_i)) \wedge (P' = P'')\right] \\
&\quad = \sum_{P'' \in \mathrm{Perm}([\kappa])} \Pr\left[P \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, (P(\mathbf{k}_i) = P''(\mathbf{k}_i))\right] \\
&\qquad \times \Pr\left[P' \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, P' = P''\right] \\
&\quad = \frac{1}{\kappa!} \sum_{P'' \in \mathrm{Perm}([\kappa])} \Pr\left[P \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, (P''^{-1}(P(\mathbf{k}_i)) = \mathbf{k}_i)\right] \\
&\quad = \Pr\left[P \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, P(\mathbf{k}_i) = \mathbf{k}_i\right] \\
&\quad = \sum_{\mathbf{a} \in \mathcal{K}} \Pr\left[P \leftarrow_\$ \mathrm{Perm}([\kappa]) \,:\, (P(\mathbf{a}) = \mathbf{a}) \wedge (\mathbf{k}_i = \mathbf{a})\right] \\
&\quad = \sum_{\mathbf{a} \in \mathcal{K}} \frac{C_{\mathcal{K}}(\mathbf{a})}{\kappa! |\mathcal{K}|} = \frac{\overline{C_{\mathcal{K}}}}{\kappa!}.
\end{aligned}$$

Hence, by summing over every possible output values, we get

$$\Pr\left[T_{\mathrm{id}} \in \Theta_2 | U \wedge (T_{\mathrm{id}} \notin \Theta_1)\right] \leq \frac{(q_*)^2 \overline{C_{\mathcal{K}}}}{\kappa!}. \tag{4}$$

Combining Eqs (1), (3) and (4) yields

$$\Pr\left[T_{\text{id}} \in \Theta_2 | T_{\text{id}} \notin \Theta_1\right] \leq \frac{q_* \kappa^2}{2^{b+1}} + \frac{(q_*)^2 \overline{C_{\mathcal{K}}}}{\kappa!}.$$

Condition (C-3). Let $(i, q, s, \mathbf{z})$, $(i', q', s', \mathbf{z}')$ where $i \neq i'$ be two queries to distinct construction oracles and let $j \in \{1, \ldots, q\}$, $j' \in \{1, \ldots, q'\}$. Recall that, in the ideal world, the l-tuple of keys is drawn uniformly at random from $\mathcal{K}^l$ independently from the queries transcript. The probability, over the random draw of $\mathbf{k}_i$ and $\mathbf{k}'_i$ that $P_{s,j}(\mathbf{k}_i) = P_{s',j'}(\mathbf{k}_{i'})$ is exactly $1/|\mathcal{K}|$. By summing over every possible pair of output values, we get

$$\Pr\left[T_{\text{id}} \in \Theta_3\right] \leq \frac{q_*^2}{|\mathcal{K}|}.$$

Condition (C-4). Similarly, let $(\mathcal{Q}_C, \mathcal{Q}_F, \mathcal{Q}_G, \mathcal{Q}'_G)$ be any attainable queries transcript. Let $(i, q, s, \mathbf{z}) \in \mathcal{Q}_C$, $j \in \{1, \ldots, q\}$ and $(u, v) \in \mathcal{Q}_F$. The probability, over the random draw of $\mathbf{k}_i$, that $P_{s,j}(\mathbf{k}_i) = u$ is exactly $\frac{1}{|\mathcal{K}|}$. By summing over every such pair of queries, we get

$$\Pr\left[T_{\text{id}} \in \Theta_3\right] \leq \frac{q_F q_*}{|\mathcal{K}|}.$$

We then analyze good transcripts.

Lemma 3 *For any good transcript τ, one has*

$$\frac{Pr[T_{re} = \tau]}{Pr[T_{id} = \tau]} = 1.$$

Proof Let $\tau = (\mathcal{Q}_C, \mathcal{Q}_F, \mathcal{Q}_G, \mathcal{Q}'_G, \overline{\mathbf{k}})$ be a good transcript. Let us denote q the number of constructions queries. Note that, since τ is a good transcript, the total number of *distinct* queries to the function G (from $\mathcal{Q}_G$ and $\mathcal{Q}'_G$) is exactly $q_G + (\kappa - 1)q_*$ since no collision occurs.

In the ideal world, the outputs of F are uniformly random and independent from the uniformly random draw of the outputs of the construction oracles, G and the keys. Thus one has

$$Pr[T_{id} = \tau] = \frac{1}{|B|^{q_*}} \cdot \frac{1}{|B|^{q_F}} \cdot \frac{1}{|\mathcal{K}|^l} \cdot \frac{1}{2^{qn+(b+n)(q_G+(\kappa-1)q_*)}}.$$

In the real world, since τ is a good transcript, we know that the inputs of the function F are always pairwise distinct, its output are then perfectly random. Since the IVs and the keys are also drawn uniformly at random, and independently from each other and from F and G, we have

$$Pr[T_{re} = \tau] = \frac{1}{|B|^{q_*+q_F}} \cdot \frac{1}{|\mathcal{K}|^l} \cdot \frac{1}{2^{qn+(b+n)(q_G+(\kappa-1)q_*)}}.$$

So we indeed have, for a good transcript τ:

$$\frac{Pr[T_{re} = \tau]}{Pr[T_{id} = \tau]} = 1.$$

3.5 Choice of Security Parameters

FHE schemes usually rely on arithmetic over finite fields. In order to maximize the advantages of a filter permutator (i.e. constant and small growth noise), it seems necessary to build a stream cipher over the same finite field. In this section, we study two possible choices of parameters and derive the corresponding security bound, depending on the relative size of κ and the finite field. These two constructions can actually be seen as generalizations of the high-level structure of the FLIP family of stream ciphers to arbitrary finite fields.

A Construction Without Weak Keys. We follow and generalize the design strategy of FLIP when relying on a finite field of small cardinality p. In this case, it is possible to choose a multiple of p for κ, and to propose a construction without weak keys[4] as follows.

Let $\mathbb{F}_p$ be a finite field. We denote $\mathsf{FP}_{\mathsf{GEN}[G],p,\kappa,F}$ the filter permutator using a pseudorandom bit generator G (with its associated stateful generator $\mathsf{GEN}[G]$) and such that the filter is a function F from $(\mathbb{F}_p)^\kappa$ to $\mathbb{F}_p$. For any multiple m of p, we specify the key space as the set $\mathcal{K}_{m,p}$ of all vectors $\mathbf{k} \in (\mathbb{F}_p)^q$ such that each $x \in \mathbb{F}_p$ is represented exactly m/p times. The following corollary states a security bound for this construction when the filter and the PRG are modeled as a uniformly random public functions.

Corollary 1 *Assume that κ is a multiple of p. For any non-negative integers q_F, q_G, q_*, one has*

$$\mathbf{Adv}^{\text{mu-kIV}}_{\mathsf{FP}^{\$}_{[\mathsf{GEN}[G],\mathcal{K}_{\kappa,p}]}}(q_*, q_F, q_G) \leq \frac{(\kappa-1)q_*\left(q_G + (\kappa-1)q_*\right)}{2^n} + q_*(q_F + 2q_*) \cdot \frac{[(\kappa/p)!]^p}{\kappa!} + \frac{q_*\kappa^2}{2^{b+1}}.$$

Proof It is easy to see that

$$|\mathcal{K}_{\kappa,p}| = \prod_{i=0}^{p-2} \binom{\kappa - i\kappa/p}{\kappa/p} = \prod_{i=0}^{p-2} \frac{(\kappa - i\kappa/p)!}{(\kappa/p)!(\kappa-(i+1)\kappa/p)!} = \frac{\kappa!}{[(\kappa/p)!]^p}.$$

Recall that $\overline{C_{\mathcal{K}}} = \sum_{\mathbf{a}\in\mathcal{K}} \frac{C_{\mathcal{K}}(\mathbf{a})}{|\mathcal{K}|}$. Let us fix a key $\mathbf{a} \in \mathcal{K}$. By definition, exactly κ/p of its coordinates are equal to i for every $i \in \mathbb{F}_p$. Thus the vector $\mathbf{a}$ is a fixed point of a permutation in $\mathrm{Perm}([\kappa])$ if and only if, for every $i \in \mathbb{F}_p$, each one of the κ/p coordinates that is equal to i is sent to another coordinate that is equal to i. This means that $\mathbf{a}$ is a fixed point for exactly $[(\kappa/p)!]^p$ permutations in $\mathrm{Perm}([\kappa])$. Then one has $\overline{C_{\mathcal{K}}} = [(\kappa/p)!]^p$.

[4] A key $\mathbf{k} \in \mathcal{K}$ will be weak when $C_{\mathcal{K}}(\mathbf{k})$ is much higher than the average $\overline{C_{\mathcal{K}}}$, since in this case collisions between inputs of the filter will occur with a higher probability.

A Construction For Large Prime Numbers. In the case of a large prime number p, using a multiple of p for κ becomes impractical. In this case we simply specify the key space $\mathcal{K}$ as $(\mathbb{F}_p)^\kappa$. Then one has the following result.

Corollary 2 *For any non-negative integers q_F, q_G, q_* and any $c \in \{0, \dots, \kappa - 1\}$, one has*

$$\mathbf{Adv}^{\mathrm{mu-kIV}}_{\mathsf{FP}^{\$}_{[\mathsf{GEN}[G],(\mathbb{F}_p)^\kappa]}}(q_*, q_F, q_G) \leq \frac{(\kappa-1)q_*\left(q_G + (\kappa-1)q_*\right)}{2^n} + \frac{q_*^2}{(c+1)!} + \frac{q_*^2}{p^{\lfloor\frac{\kappa-c}{2}\rfloor}} + \frac{q_*\kappa^2}{2^{b+1}} + \frac{(q_F+q_*)q_*}{p^\kappa}.$$

Proof One has

$$\begin{aligned}\frac{\overline{C_{\mathcal{K}}}}{\kappa!} &= \frac{|\{(P,\mathbf{a}) \in \mathrm{Perm}([\kappa]) \times (\mathbb{F}_p)^\kappa\}|}{p^\kappa \kappa!} \\ &= \Pr\left[(P,\mathbf{a}) \leftarrow_\$ \mathrm{Perm}([\kappa]) \times (\mathbb{F}_p)^\kappa \ : \ P(\mathbf{a}) = \mathbf{a}\right].\end{aligned}$$

Let $c \in \{0, \dots, \kappa - 1\}$ be an integer. The event $P(\mathbf{a}) = \mathbf{a}$ can be rewritten as the following disjunction:

(E1) the number of fixed points of P is smaller than c and $P(\mathbf{a}) = \mathbf{a}$, or
(E2) the number of fixed points of P is strictly greater than c and $P(\mathbf{a}) = \mathbf{a}$.

We first consider Event (E1). Conditioned on the fact that the number of fixed points of P is smaller than c, the event $P(\mathbf{a}) = \mathbf{a}$ will be equivalent to at least $\lfloor\frac{\kappa-c}{2}\rfloor$ equations of the form $a_{P(i)} = a_i$, where $i \in \{1, \dots, \kappa\}$ and $P(i) \neq i$. Indeed, if $P(i) = i$, the equation is automatically fulfilled. For indexes $i \in \{1, \dots, \kappa\}$ such that $P(i) \neq i$, some equations might be redundant. The whole number of equations will depend on the number and size of the orbits of P, and will be minimal when P is only constituted of cycles of length 2, in which case there are exactly $\lfloor\frac{\kappa-c}{2}\rfloor$ independent equations to be satisfied. Thus one has

$$\Pr[(\mathrm{E1})] \leq \frac{1}{p^{\lfloor\frac{\kappa-c}{2}\rfloor}}.$$

We now consider event (E2). This event is included in the event

(E'2) the number of fixed points of P is strictly greater than c.

The number of such permutations can be lower bounded as follows. First we need to choose $c+1$ indices that will be fixed points. There are $\binom{\kappa}{c+1}$ possible choices. Then, for every remaining indices, there are $(\kappa - c - 1)!$ possibilities. Thus one has

$$\Pr[(\mathrm{E2})] \leq \Pr[(\mathrm{E'2})] \leq \frac{\binom{\kappa}{c+1} \cdot (\kappa - c - 1)!}{\kappa!} \leq \frac{1}{(c+1)!}.$$

4 A Note on the FLIP family of Stream Ciphers

4.1 Description of FLIP

The FLIP family of stream ciphers follow the filter permutator structure. It has first been specified in [MJSC16]. One has $A = B = \{0,1\}$, G is a forward secure PRG based on the AES-128 [BY01], κ is an even non-negative integer and $\mathcal{K}$ is the subset of $\{0,1\}^\kappa$ containing every word of Hamming weight $\kappa/2$. The filter F is a κ-variable Boolean function defined by the direct sum of three Boolean functions f_1, f_2 and f_3 of respectively n_1, n_2 and n_3 variables, such that:

– $f_1(x_0, \ldots, x_{n_1-1}) = \bigoplus_{i=0}^{n_1-1} x_i$,
– $f_2(x_{n_1}, \ldots, x_{n_1+n_2-1}) = \bigoplus_{i=0}^{n_2/2-1} x_{n_1+2i}x_{n_1+2i+1}$,
– $f_3(x_{n_1+n_2}, \ldots, x_{n_1+n_2+n_3-1})$ is the sum of nb triangular functions T_k, each one operating on different and independent variables, where

$$T_k(y_0, \ldots, y_{m-1}) = \bigoplus_{i=1}^{k} \prod_{j=0}^{i-1} x_{j+(i-1)i/2},$$

$m = k(k+1)/2$ and $n_3 = \mathsf{nb} \cdot k(k+1)/2$.

In [MJSC16], an analysis of F is provided. In particular, its resistance against several types of attacks is proved, in the case where it is applied to uniformly random elements of $\{0,1\}^\kappa$. Although the authors stress that this does not directly translate into security arguments for FLIP instances, the authors assume that the constant Hamming weight of the inputs of the filter will not give rise to a significantly better attack and provide a table which is summarized in Figure 2. The authors also indicate that keys should not be used to encrypt more than 2^{64} bits of data.

Instance	N	λ
FLIP(42,128,8,9)	530	81
FLIP(46,136,4,15)	662	80
FLIP(82,224,8,16)	1394	134
FLIP(86,238,5,23)	1704	128

Fig. 2 FLIP$(n_1, n_2, \mathsf{nb}, k)$ instances, with the associated number of variables N and the expected security parameter λ from [MJSC16].

Corollary 1, when applied to the finite field $\mathbb{F}_2$ and using $\kappa = N$ gives the following bound in the random oracle model for the filter and the PRG:

$$\mathbf{Adv}^{\mathrm{mu-kIV}}_{\mathsf{FLIP}^{\$}_{[\mathsf{AES}_{128}, \mathcal{K}_{N,2}]}}(q_*, q_F, q_G) \leq \frac{(N-1)q_* \left(q_G + (N-1)q_*\right)}{2^{128}}$$

$$+\frac{q_*(q_F+2q_*)}{|\mathcal{K}_{N,2}|}+\frac{q_*N^2}{2^{129}},$$

$$\mathbf{Adv}^{\mathrm{mu-kIV}}_{\mathsf{FLIP}^{\$}_{[\mathsf{AES}_{256},\mathcal{K}_{N,2}]}}(q_*,q_F,q_G)\le\frac{(N-1)q_*\,(q_G+(N-1)q_*)}{2^{256}}$$

$$+\frac{q_*(q_F+2q_*)}{|\mathcal{K}_{N,2}|}+\frac{q_*N^2}{2^{257}}.$$

This result suggests that FLIP might be secure against generic multi-user known IV attacks (i.e. attacks that treat the filter and the PRG as a black box) as long as one has $\max\{q_F,q_G,q_*\}\ll\min\left(2^{64}/N,\sqrt{|\mathcal{K}_{N,2}|}\right)$ for the first variant, and as long as one has $\max\{q_F,q_G,q_*\}\ll\min\left(2^{128}/N,\sqrt{|\mathcal{K}_{N,2}|}\right)$ for the second one.

In the next section, we present a related-key attack against FLIP and discuss its practical impact.

4.2 A Related-Key attack

4.2.1 Attack Scenario and Computation Model

We suppose that we know a part of the plaintext with the corresponding ciphertext, which implies that we know some of the bits of the key stream. Moreover, we place ourselves in the context of a XOR-related key attack:

– the adversary is allowed to XOR any constant of its choice to the key;
– the adversary is allowed to feed its oracle with an IV that has already been used (potentially with a modified key register).

In order to simplify the analysis of our attack, we assume that the underlying stateful permutation generator outputs uniformly random and independent values when fed with different IVs.

Regarding the complexity analysis of our attack, given its triviality, we only take into account its query complexity.

First we describe and analyze our attack. Then we discuss the relevance of our attack scenario in section 4.2.3

4.2.2 Description of the Attack

Recall that the filter permutator's permutation generator relies on a public IV. This means that an attacker can compute the sequence of key permutations that will be used to feed the filter function. Thus, at each round of the construction, because the filtering function is known, the attacker can know the exact polynomial equation linking an output to the key variables. In particular, for each output bit, it is trivial to identify which monomial each key variable is involved in.

Assuming the attacker wants to know the value of k_j, he can then find an output bit z_i where k_j will be involved in a quadratic term. Given the fact that each key variable can only be involved in a single monomial, there exists a polynomial R such that, for a particular $j' \neq j$:

$$z_i = k_j k_{j'} \oplus R(k_1, \ldots, k_{j-1}, k_{j+1}, \ldots, k_{j'-1}, k_{j'+1}, \ldots, k_N),$$

From our hypothesis the attacker had the ability to XOR a constant to the key, and query the corresponding key stream for the same value of the IV. The same output bit z_i' using the same IV and the key value $(k_1, \ldots, k_{j'-1}, k_{j'} \oplus 1, k_{j'+1}, \ldots, k_N)$ becomes

$$\begin{aligned} z_i' &= k_j(k_{j'} \oplus 1) \oplus R(k_1, \ldots, k_{j-1}, k_{j+1}, \ldots, k_{j'-1}, k_{j'+1}, \ldots, k_N) \\ &= k_j \oplus k_j k_{j'} \oplus R(k_1, \ldots, k_{j-1}, k_{j+1}, \ldots, k_{j'-1}, k_{j'+1}, \ldots, k_N) \\ &= k_j \oplus z_i. \end{aligned}$$

It is thus easy to recover key bit k_j, using a single related-key query. Note that, by doing so, it might be possible to recover more than one key bit using another keystream bit. This can be done for each key bit that is involved in a quadratic term in at least one known keystream bit. We are now going to lower bound the number of required known keystream bits for this attack to allow complete key recovery with overwhelming probability.

Let q be the number of known keystream bits, N the total number of variables and n_1', n_2', n_3' respectively be the number of variables involved in linear, quadratic, and of degree greater than 3 monomials. Let $i \in \{1, \ldots, N\}$. Then the probability, over the uniformly random and independent draw of the outputs of the permutation generator, that k_i is *never* involved in a quadratic term, is exactly

$$\left(\frac{n_1' + n_3'}{N}\right)^q.$$

Our attack will fail to completely recover the key if there exists $i \in \{1, \ldots, N\}$, which happens with probability at most

$$N\left(\frac{n_1' + n_3'}{N}\right)^q = N\left(\frac{N - 2\mathsf{nb} - n_2}{N}\right)^q$$

for any instance of $\mathsf{FLIP}(n_1, n_2, \mathsf{nb}, n)$ and $N = n_1 + n_2 + \mathsf{nb} \cdot k(k+1)/2$.

Overall, in order to completely recover the key of an instance of $\mathsf{FLIP}(n_1, n_2, \mathsf{nb}, n)$ with a probability greater than $1 - \varepsilon$, we need at least $\frac{\log(\varepsilon/N)}{\log(1-(2\mathsf{nb}+n_2)/N)}$ known key stream bits and at most N related key queries, where $N = n_1 + n_2 + \mathsf{nb} \cdot k(k+1)/2$. For example, if we consider the safest parameters suggested in [MJSC16], where $N = 1704$, $n_2 = 238$ and $\mathsf{nb} = 8$, our attack will succeed with a probability greater than 0.99 using 75 known keystream bits and at most 1704 related key queries.

4.2.3 Discussion

Our attack scenario requires a powerful adversary. On a theoretical point of view, it makes sense to consider such a model in order to achieve a better understanding of FLIP. The particular weakness we exploit is the fact that each key variable is only involved in exactly one monomial. Adding a linear layer before the application of the filter does not seem to mitigate the impact of this attack. A potential (though expensive) solution could be to use a more complex filter, in which each variable is always involved in several monomials of various degree. However, such a countermeasure would come at the expense of a greater computational complexity.

On a practical point of view, since filter permutators are most likely used in a hybrid FHE framework, such a related key attack seems especially threatening when used by a malicious server. In that context, the server would then be in possession of the encryption of the key bits for FLIP. It is thus trivial for it to XOR any constant of its choice to the key, and to generate the associated keystream for an already used IV. However, in order to recover the value of the targeted key bit, an attacker would either have to be able to test the equality of two ciphertexts under the FHE scheme, or rely on some interaction with the user. In the former case, it essentially implies the ability of breaking the FHE scheme, thus negating the need of this attack. In the latter case, the malicious server could already use software fault injection attacks [CGG16] in order to efficiently recover plaintext or secret key bits.

Acknowledgments

This work has been partially supported by the European Union's H2020 Programme under grant agreement number ICT-644209. We would also like to thank Jean-Sébastien Coron for his feedback about early versions of this work.

References

ALP05. Frederik Armknecht, Joseph Lano, and Bart Preneel. Extending the resynchronization attack. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, pages 19–38, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

ARS$^+$15. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

BY01. Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. Cryptology ePrint Archive, Report 2001/035, 2001. http://eprint.iacr.org/2001/035.

CCF$^+$16. Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 313–333, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

CGG16. Ilaria Chillotti, Nicolas Gama, and Louis Goubin. Attacking fhe-based applications by software fault injections. Cryptology ePrint Archive, Report 2016/1164, 2016. http://eprint.iacr.org/2016/1164.

CLL$^+$14. Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the Two-Round Even-Mansour Cipher. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 (Proceedings, Part I)*, volume 8616 of *LNCS*, pages 39–56. Springer, 2014. Full version available at http://eprint.iacr.org/2014/443.

CLT14. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 311–328, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

CMR17. Claude Carlet, Pierrick Méaux, and Yann Rotella. Boolean functions with restricted input and their robustness; application to the flip cipher. Cryptology ePrint Archive, Report 2017/097, 2017. http://eprint.iacr.org/2017/097.

CNT12. Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'12, pages 446–464, Berlin, Heidelberg, 2012. Springer-Verlag.

CS14. Shan Chen and John Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014. Full version available at http://eprint.iacr.org/2013/222.

DLR16. Sébastien Duval, Virginie Lallemand, and Yann Rotella. Cryptanalysis of the flip family of stream ciphers. In *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*, pages 457–475, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 850–867, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

MJSC16. Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 311–343, 2016.

NLV11. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

Pat08. Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography - SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, 2008.