Keyed Sum of Permutations: a simpler RP-based PRF

Ferdinand Sibleyras, Yosuke Todo

NTT Social Informatics Laboratories

Abstract. Idealized constructions in cryptography prove the security of a primitive based on the security of another primitive. The challenge of building a pseudorandom function (PRF) from a random permutation (RP) has only been recently tackled by Chen, Lambooij and Mennink [CRYPTO 2019] who proposed Sum of Even-Mansour (SoEM) with a provable beyond-birthday-bound security. In this work, we revisit the challenge of building a PRF from an RP. On the one hand, we describe Keyed Sum of Permutations (KSoP) that achieves the same provable security as SoEM while being strictly simpler since it avoids a key addition but still requires two independent keys and permutations. On the other hand, we show that it is impossible to further simplify the scheme by deriving the two keys with a simple linear key schedule as it allows a non-trivial birthday-bound key recovery attack. The birthday-bound attack is mostly information-theoretic, but it can be optimized to run faster than a brute-force attack.

Keywords: RP-to-PRF, SoEM, KSoP, beyond-birthday-bound, provable security

1 Introduction

1.1 Background

Idealized Constructions. This paper pursues the long line of symmetric cryptographic effort to analyze constructions that combine some primitives into another type of primitive. Such constructions notably include the Feistel network by Luby and Rackoff [24] that constructs a pseudorandom permutation (PRP) from pseudorandom functions (PRF); the key-alternating Feistel (KAF) network of Lampe and Seurin [23] that uses random functions (RF) to build a PRP; as well as the Even-Mansour construction [18] that constructs a PRP from a random permutation (RP).

Those constructions provide an information-theoretical analysis of the strategies employed to design ciphers. For instance, the Feistel network is an idealized DES [16] and the key-alternating cipher (KAC) [6], which is akin to an iteration of Even-Mansour, is an idealized AES [1].

Pseudorandom Functions. The previously cited constructions aim at building PRPs as they are interested in idealized constructions of block ciphers. However, the security of many modes of operation such as Galois Counter Mode (GCM) [21] can be improved with a PRF instead. The proofs of such modes typically start by applying the PRP/PRF switching lemma [4] allowing to consider the underlying primitive as an actual PRF. In a nutshell, the PRP/PRF switching lemma says that a PRP behaves like a PRF up to the birthday-bound, that is up to $\mathcal{O}(2^{n/2})$ queries with n the bit-size of the PRP. The main drawback of this composition is that proofs using such technique cannot show security beyond the birthday-bound. Hence, it is of interest to build a secured PRF with provable security beyond the birthday-bound.

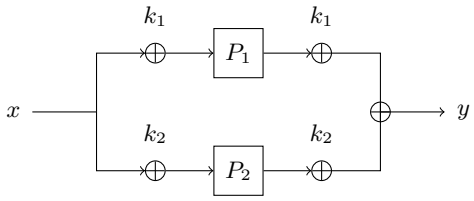

Fig. 1. Chen et al. [10] proposed to sum two single-keyed Even-Mansour with independent keys and permutations together to build a PRF: $\texttt{SoEM}(x) = P_1(x \oplus k_1) \oplus k_1 \oplus P_2(x \oplus k_2) \oplus k_2$.

From RP to PRF. Chen, Lambooij and Mennink [10] proposed the Sum of Even-Mansour construction (SoEM, Fig. 1), an RP-based PRF provably secure beyond the birthday-bound by summing two independent single-keyed Even-Mansour constructions. The single-keyed Even-Mansour construction is known to be secure only up to the birthday-bound [17], but Chen *et al.* gives a dedicated proof of SoEM that indeed shows SoEM is provably secure beyond the birthday-bond and up to $\mathcal{O}(2^{2n/3})$ queries. In the same work, Chen *et al.* also showed that the scheme *cannot* be simplified by having the two Even-Mansour constructions sharing the same key or the same permutation as it allows for a birthday-bound attack.

RP-based PRF construction is the topic of this paper. In particular, we look for possible simplifications of the scheme and study the effect on its provable security.

1.2 Results

There are two results, a positive and a negative one. We first show how SoEM can be simplified and its proof adapted to a new scheme we called Keyed Sum

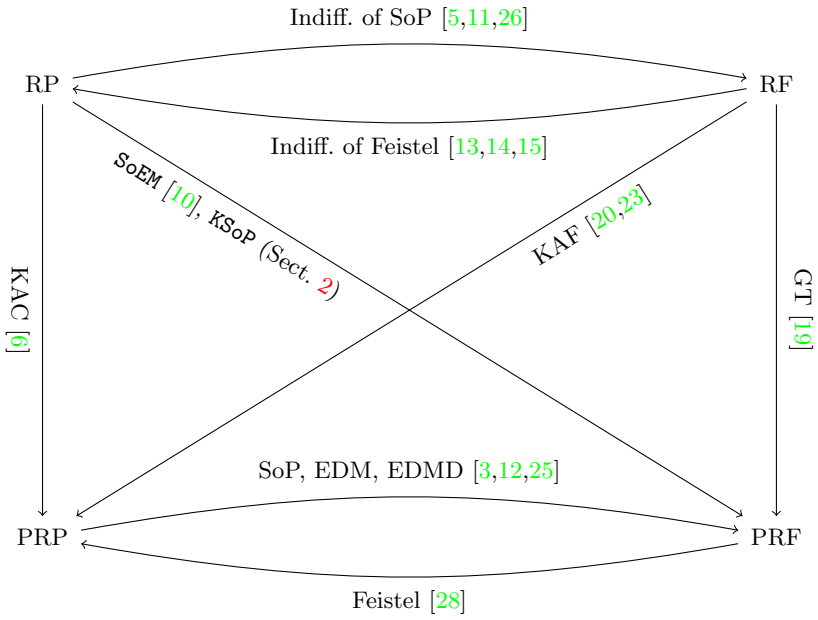

Fig. 2. Some examples of idealized constructions [10] between random/pseudorandom permutation/function (RP, PRP, RF, PRF).

of Permutations that drops the last key addition of SoEM:

$$\texttt{KSoP}(x) = P_1(x \oplus k_1) \oplus P_2(x \oplus k_2) \ .$$

Then, we show that the scheme *cannot* be simplified by having the two keys related by a simple linear key schedule as it leads to a non-trivial information-theoretic birthday-bound attack.

Keyed Sum of Permutations. In Section 2 we prove Theorem 1 stating that KSoP (Fig. 3) is secure beyond the birthday-bound and up to $\widetilde{\mathcal{O}}\big(2^{2n/3}\big)$ queries.

While KSoP can be described as SoEM without the last key addition, one can argue that the design strategy is opposite. Indeed, looking at Figure 2, SoEM strategy goes to first build two PRPs via Even-Mansour and then add their output to build a PRF:

$$\texttt{SoEM} : \text{RP} \xrightarrow{\texttt{EM}} \text{PRP} \xrightarrow{\texttt{SoP}} \text{PRF}$$

Indeed, it is known that Even-Mansour is a secure PRP and that the sum of two PRPs is a secure PRF. Although the direct combination of the proofs only guarantees a birthday-bound security, Chen *et al.* [10] gives a better dedicated proof.

On the other hand, by removing the last key addition, KSoP can no longer be described as the sum of two PRPs. Instead, KSoP's strategy goes the other way

around:

$$\texttt{KSoP} : \mathrm{RP} \xrightarrow{\texttt{SoP}} \mathrm{RF} \xrightarrow{\texttt{GT}} \mathrm{PRF}$$

It first sums two random permutations to build a random function; that is provably indifferentiable from a random function [5]; and then masks the inputs with two keys which is also known to be enough to build a secure PRF [19]. Again, this requires a dedicated analysis to prove the construction secure beyond the birthday-bound which is the topic of Section 2.1.

Linear Key-Schedule. Chen *et al.* [10] showed that SoEM is insecure when reusing the same key for both Even-Mansour construction. The attack they showed applies as such for KSoP. However, they did not study the effect of simple key-schedules on the security. For instance, it is known that the two-round Even-Mansour can be made secure using the same permutation, and using a linear key schedule (precisely a linear orthomorphism, like a finite field multiplication) is enough to guarantee a beyond birthday-bound security [8] even though the security collapses to birthday-bound in the absence of key-schedule.

In a paper studying the quantum security of SoEM and its variants, Shinagawa and Iwata [29] left the security of using a linear key schedule as an open question. Indeed, it is natural to consider the security of KSoP* with a linear key schedule Γ:

$$\texttt{KSoP*}(x) = P_1(x \oplus k) \oplus P_2(x \oplus \Gamma(k)) \ .$$

We answer the question of its classical security in Section 3 with an attack showing that the information-theoretic security of KSoP* is actually no better than birthday-bound. The cryptanalysis is more technical and differs substantially from the cryptanalyses of variants of SoEM with identical keys or identical permutations that are essentially looking for a collision. In particular, the time complexity of our birthday-bound attack is much higher, at $\widetilde{\mathcal{O}}(2^n)$. However, since in practice permutation queries can be assimilated to offline computations, we give in Section 3.3 a range of trade-offs to reduce the time and online query complexities at the expense of offline permutation queries. In particular, the time complexity can be lowered down to $\widetilde{\mathcal{O}}\big(2^{3n/4}\big)$.

We also verified the correctness of our attack experimentally[1].

Organization. We prove the security of KSoP in Section 2. A key-recovery attack when a linear key-schedule is used is described in Section 3 and concrete examples on how to run this attack are shown in Section 4.

1.3 Notations

We denote by $\{0,1\}^n$ the set of all n-bit strings. $a \xleftarrow{\$} \mathcal{A}$ means that a is randomly uniformly drawn from the set $\mathcal{A}$. $\mathcal{A}^*$ is shorthand for $\mathcal{A} \setminus \{0\}$.

A value $x \in \{0,1\}^n$ equivalently represents a *row* vector of n bits or a value in a finite field of characteristic 2 that is $x \in \mathrm{GF}(2^n)$. Similarly, an $m \times n$ bit-matrix X can be equivalently seen as a set of m values in $\mathrm{GF}(2^n)$.

[1] The source code is available in https://anonymous.4open.science/r/soem-335F/README.md.

The set spanned by all elements of an $m \times n$ matrix X is written $\mathrm{sp}\{X\} = \{\, eX : e \in \{\,0,1\,\}^m \,\}$. The set spanned by all elements of two matrices X and Y (equivalently, by all elements of $X \cup Y$), is written $\mathrm{sp}\{X, Y\}$. By convention, $\mathrm{sp}\{\emptyset\} = \{\,0\,\}$.

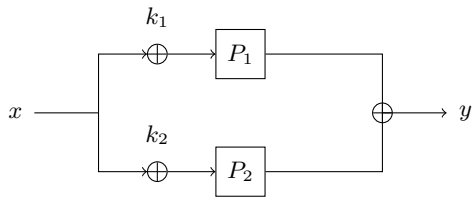

Fig. 3. The Keyed Sum of Permutation is a `SoEM` without the last key addition and computes $y = \texttt{KSoP}(x) = P_1(x \oplus k_1) \oplus P_2(x \oplus k_2)$.

2 Simplifying the Scheme

The `SoEM` scheme [10] can be further simplified by removing the last two key additions as in Figure 3. The results in a scheme that is strictly simpler than `SoEM` while retaining a beyond-birthday-bound PRF security.

Theorem 1. *Let the `KSoP` scheme, parametrized by two n-bit permutations P_1 and P_2 and two n-bit keys k_1 and k_2, be an oracle that for every input x returns $\texttt{KSoP}_{k_1,k_2}(x) := P_1(x \oplus k_1) \oplus P_1(x \oplus k_2)$. Let $\mathcal{P}$ and $\mathcal{F}$ be the set of all n-bit to n-bit permutations and functions, respectively. Then, for any distinguisher $\mathcal{D}$ interacting with three oracles, making p forward and backward queries to the first two oracles and q queries to its third oracle we have:*

$$\mathbf{Pr}\left[\mathcal{D}^{P_1,P_2,\texttt{KSoP}_{k_1,k_2}} \to 1\right] - \mathbf{Pr}\left[\mathcal{D}^{P_1,P_2,\$} \to 1\right] \leq \frac{4q(p+q)^2}{2^{2n}} + 2^{2-n} + 3\frac{qp^2}{2^{2n}} + 4\sqrt{n}\frac{p\sqrt{q}}{2^n}$$

with the randomness of $(P_1, P_2) \xleftarrow{\$} \mathcal{P}^2$, $\$ \xleftarrow{\$} \mathcal{F}$, $(k_1, k_2) \xleftarrow{\$} \{\,0,1\,\}^{2n}$ and the choices of $\mathcal{D}$.

2.1 Proof of `KSoP`

The proof of `KSoP` is mostly similar to the original proof of `SoEM` [10] and follows Patarin's H-Coefficient Technique. In fact, the only difference is in the analysis of the bad transcripts that will require the sum-capture Theorem [2,30]:

Theorem 2. *Let $\mathcal{B}_1$ and $\mathcal{B}_2 \subseteq \{0,1\}^n$ be arbitrary sets of p elements and $\mathcal{Y} \subseteq \{0,1\}^n$ a set of q elements drawn uniformly at random. Define the set of all triplets that sum to zero that is $T := \{(y, b_1, b_2) \in \mathcal{Y} \times \mathcal{B}_1 \times \mathcal{B}_2 : y \oplus b_1 \oplus b_2 = 0\}$ and $\rho := |T|$ the number of such triplets. Then for any value c we have:*

$$\mathbf{Pr}\left[\rho \geq \frac{qp^2}{2^n} + cp\sqrt{q}\right] \leq 2^{n+1} e^{-\frac{c^2}{2}} .$$

Patarin's H-Coefficient Technique. Consider a computationally unbounded deterministic adaptive adversary $\mathcal{D}$ for a distinguishing game between a real and an ideal world.

Let τ be the transcript all interactions made by $\mathcal{D}$ to its oracles. Let X_{re} and X_{id} be random variables denoting the transcripts in the real and ideal worlds, respectively. The probability that the transcript τ is realized in the real and ideal world is noted $\mathbf{Pr}[X_{\text{re}} = \tau]$ and $\mathbf{Pr}[X_{\text{id}} = \tau]$, respectively. Let Θ be the set of all attainable transcripts in the ideal world. The main theorem of the H-coefficient technique [27,9] is as follows.

Theorem 3 (H-coefficient technique). *Let $\mathcal{D}$ be an adversary that has access to either the real world oracles $\mathcal{O}_{\text{re}}$ or the ideal world oracles $\mathcal{O}_{\text{id}}$. Let $\Theta = \Theta_{\text{g}} \sqcup \Theta_{\text{b}}$ be some partition of the set of all attainable transcripts into* good *and* bad *transcripts. Suppose there exists $\epsilon_{\text{ratio}} \geq 0$ such that for any $\tau \in \Theta_{\text{g}}$,*

$$\frac{\mathbf{Pr}[X_{\text{re}} = \tau]}{\mathbf{Pr}[X_{\text{id}} = \tau]} \geq 1 - \epsilon_{\text{ratio}} ,$$

and there exists $\epsilon_{\text{bad}} \geq 0$ such that $\mathbf{Pr}[X_{\text{id}} \in \Theta_{\text{b}}] \leq \epsilon_{\text{bad}}$. Then,

$$\mathbf{Pr}\left[\mathcal{D}^{\mathcal{O}_{\text{re}}} \to 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\mathcal{O}_{\text{id}}} \to 1\right] \leq \epsilon_{\text{ratio}} + \epsilon_{\text{bad}} . \tag{1}$$

Game Setting. The adversary $\mathcal{D}$ interacts with 3 oracles. In both the real and ideal worlds, P_1 and P_2 are drawn randomly from the set of all n-bit permutations and $\mathcal{D}$ can perform forward and backward queries to them via two oracles. That is $\mathcal{D}$ can query a and add $(a, P_1(a))$ to its transcript or query b and add $(P_1^{-1}(b), b)$ to its transcript. Similarly for P_2.

In the real world, the third oracle first draws k_1 and k_2 at random and answers every query x by computing $E_{\texttt{KSoP}}(x) = P_1(x \oplus k_1) \oplus P_2(x \oplus k_2)$, that is the KSoP cipher, thus adding $(x, E_{\texttt{KSoP}}(x))$ to the transcript. In the ideal world, the third oracle answers every query x by a random value $y \xleftarrow{\$} \{0,1\}^n$ thus adding (x, y) to the transcript. The keys k_1 and k_2 are only drawn at the end of interactions.

We abuse notations and call the oracles of the real world P_1, P_2 and E while the ideal world oracles are P_1, P_2 and \$. Moreover, at the end of interactions, k_1 and k_2 are revealed to the adversary before its output decision.

Hence, the transcript τ contains three sets of pairs of n-bit values $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}$ that record the interactions to the three oracles P_1, P_2 and E (or \$) respectively,

as well as the keys k_1 and k_2. We further define the following sets:

$$\begin{aligned} \mathcal{A}_i &= \{\, a : (a,b) \in \mathcal{P}_i \,\} && i \in \{\,1,2\,\} \\ \mathcal{B}_i &= \{\, b : (a,b) \in \mathcal{P}_i \,\} && i \in \{\,1,2\,\} \\ \mathcal{X} &= \{\, x : (x,y) \in \mathcal{Q} \,\} \\ \mathcal{Y} &= \{\, y : (x,y) \in \mathcal{Q} \,\} \end{aligned}$$

Bad Events. Following the H-coefficient technique, the attainable ideal world transcripts are split into two categories: the *good* and *bad* transcripts. *Good* transcripts are all transcripts that aren't *bad* and *bad* transcripts are defined as transcripts where there exists three pairs $(a_1,b_1),(a_2,b_2),(x,y) \in \mathcal{P}_1 \times \mathcal{P}_2 \times \mathcal{Q}$ such that one of the following so-called bad events occurs:

1. $\begin{cases} x \oplus a_1 = k_1 \\ x \oplus a_2 = k_2 \end{cases}$
2. $\begin{cases} x \oplus a_1 = k_1 \\ y \oplus b_1 \oplus b_2 = 0 \end{cases}$
3. $\begin{cases} x \oplus a_2 = k_2 \\ y \oplus b_1 \oplus b_2 = 0 \end{cases}$

Notice that the definition of the bad events implies that for a good transcript, for all $x \in \mathcal{X}$ we have $x \oplus k_1 \notin \mathcal{A}_1$ or $x \oplus k_2 \notin \mathcal{A}_2$ which intuitively ensures that each real world encryption $E_{\mathtt{KSoP}}(x) = P_1(x \oplus k_1) \oplus P_2(x \oplus k_2)$ is randomized by at least one permutation call.

Bad Transcripts. Let $X_{\mathrm{re}}^{\mathtt{KSoP}}$ and $X_{\mathrm{id}}^{\mathtt{KSoP}}$ be random variables denoting the transcripts in the real and ideal worlds of the KSoP game. We now bound the probability that a transcript in the ideal world is bad as the sum of the probabilities of every bad event:

$$\mathbf{Pr}\big[X_{\mathrm{id}}^{\mathtt{KSoP}} \in \Theta_b\big] \leq \sum_{i=1}^{3} \mathbf{Pr}[\mathrm{bad}_i]$$

The first bad event is easily bounded with the randomness of k_1 and k_2 since they are drawn after interactions. Thus,

$$\mathbf{Pr}[\mathrm{bad}_1] \leq \frac{qp^2}{2^{2n}} \,.$$

The second and third bad events' analysis first requires to bound the number of triplets that satisfies the second condition that is $y \oplus b_1 \oplus b_2 = 0$. Let ρ be the number of such triplets; since all values y are random in the ideal world we can directly apply Theorem 2:

$$\mathbf{Pr}\left[\rho \geq \frac{qp^2}{2^n} + cp\sqrt{q}\right] \leq 2^{n+1} e^{-\frac{c^2}{2}} \,.$$

Then, we can use the randomness of k_1 (or k_2) to finish the analysis:

$$\begin{aligned}
&\mathbf{Pr}[\mathrm{bad}_2] = \frac{\rho}{2^n} \\
&\mathbf{Pr}[\mathrm{bad}_2] \leq \mathbf{Pr}\left[\rho \geq \frac{qp^2}{2^n} + cp\sqrt{q}\right] + \frac{\frac{qp^2}{2^n} + cp\sqrt{q}}{2^n} \\
&\mathbf{Pr}[\mathrm{bad}_2] \leq 2^{n+1}e^{-\frac{c^2}{2}} + \frac{qp^2}{2^{2n}} + \frac{cp\sqrt{q}}{2^n} \\
&\mathbf{Pr}[\mathrm{bad}_2] \leq 2^{n+1}e^{-2n} + \frac{qp^2}{2^{2n}} + 2\sqrt{n}\frac{p\sqrt{q}}{2^n} \qquad\qquad c \leftarrow 2\sqrt{n} \\
&\mathbf{Pr}[\mathrm{bad}_2] \leq 2^{1-n} + \frac{qp^2}{2^{2n}} + 2\sqrt{n}\frac{p\sqrt{q}}{2^n}\ .
\end{aligned}$$

The same bound applies to $\mathbf{Pr}[\mathrm{bad}_3]$.

Thus, we conclude that:

$$\mathbf{Pr}\left[X_{\mathrm{id}}^{\mathtt{KSoP}} \in \Theta_b\right] \leq \sum_{i=1}^{3} \mathbf{Pr}[\mathrm{bad}_i] \leq 2^{2-n} + 3\frac{qp^2}{2^{2n}} + 4\sqrt{n}\frac{p\sqrt{q}}{2^n}$$

Good Transcripts. We can use the analysis of the original `SoEM` [10] for the good transcripts of `KSoP`. Indeed, the only difference between the `SoEM` and `KSoP` game is that the real world encryption oracle of `SoEM` computes $E_{\mathtt{SoEM}}(x) = P_1(x \oplus k_1) \oplus P_2(x \oplus k_2) \oplus k_1 \oplus k_2 = E_{\mathtt{KSoP}}(x) \oplus k_1 \oplus k_2$ and the bad transcripts are defined as containing three pairs $(a_1, b_1), (a_2, b_2), (x, y) \in \mathcal{P}_1 \times \mathcal{P}_2 \times \mathcal{Q}$ such that one of the following occurs:

1. $\begin{cases} x \oplus a_1 = k_1 \\ x \oplus a_2 = k_2 \end{cases}$
2. $\begin{cases} x \oplus a_1 = k_1 \\ y \oplus b_1 \oplus b_2 = k_1 \oplus k_2 \end{cases}$
3. $\begin{cases} x \oplus a_2 = k_2 \\ y \oplus b_1 \oplus b_2 = k_1 \oplus k_2 \end{cases}$

Let $X_{\mathrm{re}}^{\mathtt{SoEM}}$ and $X_{\mathrm{id}}^{\mathtt{SoEM}}$ be random variables denoting the transcripts in the real and ideal worlds of the `SoEM` game. We will use the proof of Jha and Nandi [22] that shows that for all good transcript τ we have:

$$\frac{\mathbf{Pr}[X_{\mathrm{re}}^{\mathtt{SoEM}} = \tau]}{\mathbf{Pr}[X_{\mathrm{id}}^{\mathtt{SoEM}} = \tau]} \geq 1 - \frac{2qp^2 + 6q^2p + 5q^3}{2^{2n}}$$

Notice that for each good transcript $\tau = \{\, \mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}, k_1, k_2 \,\}$ in the `KSoP`game, we can build a good transcript $\tau' = \{\, \mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}', k_1, k_2 \,\}$ for the `SoEM` game such that $\mathcal{Q}' = \{\, (x, y \oplus k_1 \oplus k_2) : (x, y) \in \mathcal{Q} \,\}$. Such related good transcripts

τ, τ' are built using the same randomness, so it is clear that $\mathbf{Pr}[X_{\text{re}}^{\texttt{SoEM}} = \tau'] = \mathbf{Pr}[X_{\text{re}}^{\texttt{KSoP}} = \tau]$ and $\mathbf{Pr}[X_{\text{id}}^{\texttt{SoEM}} = \tau'] = \mathbf{Pr}[X_{\text{id}}^{\texttt{KSoP}} = \tau]$. Therefore

$$\frac{\mathbf{Pr}[X_{\text{re}}^{\texttt{KSoP}} = \tau]}{\mathbf{Pr}[X_{\text{id}}^{\texttt{KSoP}} = \tau]} = \frac{\mathbf{Pr}[X_{\text{re}}^{\texttt{SoEM}} = \tau']}{\mathbf{Pr}[X_{\text{id}}^{\texttt{SoEM}} = \tau']} \geq 1 - \frac{2qp^2 + 6q^2p + 5q^3}{2^{2n}}$$

Security of `KSoP`. By combining the analysis of the bad and good transcripts with the H-coefficient technique of Theorem 3 we obtain the following security bound for `KSoP`:

$$\begin{aligned}&\mathbf{Pr}\left[\mathcal{D}^{P_1,P_2,\texttt{KSoP}} \to 1\right] - \mathbf{Pr}\left[\mathcal{D}^{P_1,P_2,\$} \to 1\right] \\ &\leq \frac{5qp^2 + 6q^2p + 5q^3}{2^{2n}} + 2^{2-n} + 4\sqrt{n}\frac{p\sqrt{q}}{2^n}\ .\end{aligned}$$

Hence, `KSoP` is a PRF secure beyond the birthday-bound and up to $\widetilde{\mathcal{O}}\left(2^{\frac{2n}{3}}\right)$ queries.

3 (In)Security of Linear Key Schedule

A typical way to further simplify a scheme is to reduce the number of keys. Chen *et al.* [10] already showed that `SoEM` is not secure when using the same key $k_1 = k_2$, but they did not consider the case of a simple key schedule.

For instance, it is known that the 2-round Even-Mansour is not safe beyond birthday-bound when using the same key thrice, but it can become so with a simple linear key-schedule, as simple as a doubling in a Galois field GF(2^n). Shinagawa and Iwata [29] asked whether we could do the same for `SoEM` as using a linear key schedule seems to effectively thwart the simple collision attack in the case of $k_1 = k_2$ [10].

In this section, we show that a linear key schedule is not enough to guarantee a PRF security beyond the birthday-bound of `KSoP`. These attacks are easily extendable to the original `SoEM` with the same linear key schedule. We show examples of the described attack on concrete instances in Section 4.

3.1 Generic Strategy

Let $\Gamma(\cdot)$ be a linear key schedule and define the `KSoP*` scheme as `KSoP` where $k_1 = k$ and $k_2 = \Gamma(k)$, as in Figure 4. We can alternatively write the linear function Γ as a matrix multiplication $\Gamma(k) = k\Gamma$ where k is a row vector and Γ is now an $n \times n$ bit matrix.

The proof of `KSoP` of Section 2.1 does not apply to `KSoP*` as it fails to bound the first bad event that is to find a triplet of queries (a_1, b_1) to P_1, (a_2, b_2) to P_2, and (x, y) to `KSoP*` such that:

1. $\begin{cases} x \oplus a_1 = k \\ x \oplus a_2 = k\Gamma \end{cases}$

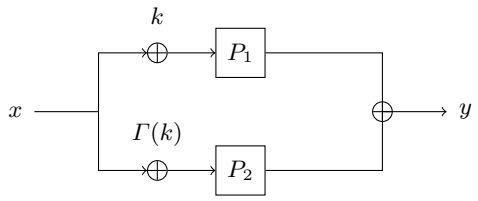

Fig. 4. KSoP with a linear key schedule $\Gamma(\cdot)$ computes $y = \texttt{KSoP*}(x) = P_1(x \oplus k) \oplus P_2(x \oplus \Gamma(k))$.

which is equivalent to (I is the n by n identity matrix):

$$\begin{cases} x \oplus a_1 = k \\ x(I \oplus \Gamma) \oplus a_1\Gamma \oplus a_2 = 0 \end{cases} \tag{2}$$

where only k is random, everything else is chosen by the adversary. In particular, the adversary can choose the sets $\mathcal{X}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ that are the values he will query to its oracles.

Key Recovery. The attack strategy starts by defining three $n/2$ by n bit matrices X, A_1, A_2 that will guide our query sets:

$$\mathcal{X} = \text{sp}\{X\} = \left\{ eX : e \in \{0,1\}^{n/2} \right\}$$
$$\mathcal{A}_i = \text{sp}\{A_i\} = \left\{ eA_i : e \in \{0,1\}^{n/2} \right\} \qquad i \in \{1,2\}$$

where e is an $n/2$-bit row vector. In other words, $\mathcal{X}$ is the set of all linear combinations of the $n/2$ row vectors forming the matrix X. Algorithm 1 shows the generic attack procedure. The algorithm is quite non-trivial, but the objective of each step becomes clear as we prove that it actually recovers the key.

Success Analysis. Notice that if there is a triplet $(x, a_1, a_2) \in \mathcal{X} \times \mathcal{A}_1 \times \mathcal{A}_2$ that satisfies Equations 2 then it will pass the test of Step 10 and the key recovery will succeed.

Now we claim that for all couple $(x, a_1) \in \mathcal{X} \times \mathcal{A}_1$, there is a value $a_2 \in \mathcal{A}_2$ such that $x(I \oplus \Gamma) \oplus a_1\Gamma \oplus a_2 = 0$. By construction, $A_1 = X(I \oplus \Gamma^{-1})$ and $A_2 = X(I \oplus \Gamma)$ therefore:

$$x(I \oplus \Gamma) \in \text{sp}\{X(I \oplus \Gamma)\}$$
$$a_1\Gamma \in \text{sp}\{A_1\Gamma\} = \text{sp}\{X(I \oplus \Gamma^{-1})\Gamma\} = \text{sp}\{X(I \oplus \Gamma)\}$$
$$a_2 \in \text{sp}\{A_2\} = \text{sp}\{X(I \oplus \Gamma)\}$$

Since $x(I \oplus \Gamma) \oplus a_1\Gamma \in \text{sp}\{X(I \oplus \Gamma)\}$ there is necessarily a matching value $a_2 \in \text{sp}\{X(I \oplus \Gamma)\}$ that will satisfy the equation $x(I \oplus \Gamma) \oplus a_1\Gamma = a_2$.

Algorithm 1 Generic Key Recovery

1: **input:** P_1, P_2, E .
2: **output:** $k : E(x) = P_1(x \oplus k) \oplus P_2(x \oplus k\Gamma)$.
3: **procedure** KEYEXTRACTOR(P_1, P_2, E)
4: Find $n/2$ by n bit matrix X such that $\mathrm{sp}\{X\Gamma, X\} = \{0,1\}^n$.
▷ See Section 3.2.
5: $A_1 \leftarrow X(I \oplus \Gamma^{-1})$
6: $A_2 \leftarrow X(I \oplus \Gamma)$

7: $L_X \leftarrow \{ e_1(x) = x(I \oplus \Gamma) \parallel E(x) : x \in \mathrm{sp}\{X\} \}$
8: $L_{A_1} \leftarrow \{ e_2(a_1) = a_1\Gamma \parallel P_1(a_1) : a_1 \in \mathrm{sp}\{A_1\} \}$
9: $L_{A_2} \leftarrow \{ e_3(a_2) = a_2 \parallel P_2(a_2) : a_2 \in \mathrm{sp}\{A_2\} \}$

10: $\Phi \leftarrow \{ (e_1(x), e_2(a_1), e_3(a_2)) \in L_X \times L_{A_1} \times L_{A_2} : e_1 \oplus e_2 \oplus e_3 = 0 \}$ ▷ 3-XOR
11: **for all** $(e_1(x), e_2(a_1), e_3(a_2)) \in \Phi$ **do**
12: $\hat{k} \leftarrow x \oplus a_1$
13: **if** $\hat{k}$ is the key **then** ▷ Test with a few stored online queries
14: **return** $\hat{k}$
15: **end if**
16: **end for**
17: **end procedure**

Let us assume that we can do Step 4, we discuss this step in the next Section 3.2. Next, we claim that for any value $k \in \{0,1\}^n$ there is a couple $(x, a_1) \in \mathcal{X} \times \mathcal{A}_1$ such that $x \oplus a_1 = k$. In other word, $\mathrm{sp}\{X, A_1\} = \{0,1\}^n$. This is again coming from the fact that we computed X and A_1 such that:

$$\{0,1\}^n = \mathrm{sp}\{X\Gamma, X\} = \mathrm{sp}\{X\Gamma \oplus X, X\} = \mathrm{sp}\{X(I \oplus \Gamma), X\} = \mathrm{sp}\{A_1, X\}$$

Combining the two observations shows that there will be a triplet satisfying Equations 2 of Step 10 thus the algorithm will succeed. The number of false positives is expected to be 1 on average as we showed that 2^n triplets will collide on the first n-bit half and the second half is an n-bit filter. It is easy to deal with false positives by testing the guessed key and continuing until the key recovery is successful.

Complexity analysis. The online and offline query complexities depend on the dimension of X found in Step 4. To span n dimension, X and $X\Gamma$ must contain at least n elements thus X must contain at least $n/2$ elements of $\{0,1\}^n$. We show in Section 3.2 that we can always find such a minimal X that makes for a total online and offline query complexities of $q = p = 2^{\lceil n/2 \rceil}$, that is the birthday-bound.

On the other hand, the time and memory complexities are determined by how we tackle Step 10, that is how we filter the triplets to find the correct one. Unfortunately, even if we treat it as a 3-XOR problem (Definition 1) with three lists of size $2^{n/2}$, the best algorithms [7] run in $\widetilde{\mathcal{O}}(2^n)$ time and $\mathcal{O}\left(2^{n/2}\right)$ memory.

Definition 1. *The 3-XOR problem. Let $\ell > 0$ an integer. Given three lists $L_1, L_2, L_3 \subseteq \{0,1\}^\ell$ find a triplet $(e_1, e_2, e_3) \in L_1 \times L_2 \times L_3$ such that:*

$$e_1 \oplus e_2 \oplus e_3 = 0 \ .$$

Discussion. Brute-forcing uses a handful of online queries and almost no memory but requires $\mathcal{O}(2^n)$ offline queries that are computations of the underlying permutations. Therefore, our present key recovery attack requires more online queries and memory and, most importantly, $\mathcal{O}(2^n)$ computations as well. However, this birthday-bound attack shows that one cannot prove `KSoP*` information-theoretically secure beyond the birthday-bound.

This raises the question of any computational beyond-birthday-bound security of `KSoP*`. While proving computational security is hard, we can actually optimize the attack to reduce the time complexity down to $\widetilde{\mathcal{O}}(2^{3n/4})$ which represents an exponential speed-up with respect to brute-force. We show in Section 3.3 trade-offs for reducing the time and online query complexities at the cost of more offline queries.

3.2 Looking For The Minimal Matrix

All discussion regarding Algorithm 1 assumes that Step 4 returns a minimally sized X such that $\mathrm{sp}\{X\Gamma, X\} = \{0,1\}^n$. In this section, we show how it can systematically be done in the case of finite field multiplication which can then be adapted to other linear key schedule.

Algorithm 2 Set gathering

```
1: input: n by n bit matrix Γ .
2: output: Multiple sets such that all elements of all sets together span {0,1}^n .
3: procedure SetGathering(Γ)
4:     Φ ← ∅
5:     while ∃u ∈ {0,1}^n : u ∉ sp{Φ} do
6:         S ← { u }
7:         while uΓ ∉ sp{S} do            ▷ Effectively builds S_u as in Theorem 4.
8:             u ← uΓ
9:             S ← S ∪ { u }
10:        end while
11:        Φ ← Φ ∪ { S }                  ▷ Φ is a set of sets.
12:    end while
13:    return Φ
14: end procedure
```

Theorem 4. *Let Γ be an invertible n by n matrix in $GF(2)$. For any $u \in \{0,1\}^n \setminus 0$, define h as the smallest natural number such that $u\Gamma^h \in \mathrm{sp}\{S_u\}$ with $S_u := \{ u\Gamma^i : i \in [0, h-1] \}$.*

Then we have:

- $v \in \mathrm{sp}\{S_u\} \Leftrightarrow v\Gamma \in \mathrm{sp}\{S_u\}$.

Proof of Theorem 4. Since h is minimal, $u\Gamma^{h-1} \notin \mathrm{sp}\{\{\ u\Gamma^i : i \in [0, h-2]\ \}\}$, therefore $u\Gamma^h \notin \mathrm{sp}\{S_u \setminus \{\ u\ \}\}$. Hence, $u\Gamma^h$ can be written as a linear combination depending on u, so we can remove u and add $u\Gamma^h$ to the set S without affecting the span: $\mathrm{sp}\{S_u\} = \mathrm{sp}\{\{\ u\Gamma^i : i \in [1, h]\ \}\}$.

If $v \in \mathrm{sp}\{S_u\}$ then $v\Gamma \in \mathrm{sp}\{\{\ u\Gamma^i : i \in [1, h]\ \}\} = \mathrm{sp}\{S_u\}$. The forward implication is done.

By induction, we deduce that $\mathrm{sp}\{S_u\} = \mathrm{sp}\{\{\ u\Gamma^i : i \in [j, j+h-1]\ \}\}$ for any $j \geq 0$. Moreover, Γ being invertible means that its application is a permutation over the finite set $\{\,0,1\,\}^n$. Therefore, there exists a $\rho \geq 0$ such that $u\Gamma^\rho = u$ (note that ρ depends on both u and Γ) thus $u\Gamma^i = u\Gamma^{i+\rho}$ for any i. In particular, $\mathrm{sp}\{S_u\} = \mathrm{sp}\{\{\ u\Gamma^i : i \in [\rho-1, \rho+h-2]\ \}\} = \mathrm{sp}\{\{\ u\Gamma^i : i \in [-1, h-2]\ \}\}$.

If $v\Gamma \in \mathrm{sp}\{S_u\}$ then $v \in \mathrm{sp}\{\{\ u\Gamma^i : i \in [-1, h-2]\ \}\} = \mathrm{sp}\{S_u\}$. The backward implication is done.

Set Gathering. The goal of Algorithm 2 is to build sets $\{\ u, u\Gamma, u\Gamma^2, ..., u\Gamma^{\ell-1}\ \}$ that, taken together, span the whole space $\{\,0,1\,\}^n$. For instance, if the set $S_u = \{\ u, u\Gamma, u\Gamma^2, ..., u\Gamma^{n-1}\ \}$ spans $\{\,0,1\,\}^n$, then we can build the row of X with the elements $\{\ u, u\Gamma^2, ..., u\Gamma^{n-2}\ \}$, and we have that $\mathrm{sp}\{X, X\Gamma\} = \mathrm{sp}\{S_u\} = \{\,0,1\,\}^n$.

We show that this algorithm is particularly efficient when Γ represents some Galois field multiplication by a value $\gamma \in \mathrm{GF}(2^n) \setminus \{\,0,1\,\}$. In that case, following Theorem 5, Algorithm 2 will output exactly n/h sets containing h elements from some $h \geq 2$.

Theorem 5. *Let $\gamma \in GF(2^n)^*$. Define h as the smallest natural number such that $\gamma^h \in \mathrm{sp}\{S\}$ with $S := \{\ \gamma^i : i \in [0, h-1]\ \}$. For any $u \in GF(2^n)^*$, define $S_u := \{\ u\gamma^i : i \in [0, h-1]\ \}$.*

Then we have:

- $1 \in \mathrm{sp}\{S_u\} \Leftrightarrow u \in \mathrm{sp}\{S\}$.
- $v \in \mathrm{sp}\{S\}^* \Leftrightarrow v^{-1} \in \mathrm{sp}\{S\}^*$.
- $\forall i \geq 1, \forall \{\ u_1, u_2, ..., u_i\ \} \subseteq GF(2^n)^*, \forall v \notin \mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\}$: $\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\} \cap \mathrm{sp}\{S_v\} = \{\ 0\ \}$.
- *h divides n.*

Proof of Theorem 5. We can equivalently write $\mathrm{sp}\{S\}$ as the set of all polynomials of γ and $\mathrm{sp}\{S_u\}$ the set of all polynomials of γ multiplied by u. Therefore, for any $k \in \mathrm{sp}\{S\}$ and any $u \in \mathrm{GF}(2^n)^*$ we have $uk \in \mathrm{sp}\{S_u\}$.

If $1 \in \mathrm{sp}\{S_u\}$ for some $u \in \mathrm{GF}(2^n)^*$ then so does $\gamma, \gamma^2, ... \in \mathrm{sp}\{S_u\}$ (Theorem 4) that is $S \subseteq \mathrm{sp}\{S_u\}$; since S contains h linearly independent values $\mathrm{sp}\{S\} = \mathrm{sp}\{S_u\}$ thus $u \in \mathrm{sp}\{S\}$.

Assume that $v^{-1} \in \mathrm{sp}\{S\}$ then $vv^{-1} = 1 \in \mathrm{sp}\{S_v\}$ implying $v \in \mathrm{sp}\{S\}$. Same for the converse.

For some $i \geq 1$, let $\{\ u_1, u_2, ..., u_i\ \} \subseteq \mathrm{GF}(2^n)^*$ and $v \notin \mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\}$. Assume there exists a value $a \neq 0$ such that $a \in \mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\}$ and $a \in \mathrm{sp}\{S_v\}$. We can thus write $a = v\beta = u_1\alpha_1 + u_2\alpha_2 + ... + u_i\alpha_i$ for some α_j and β values in $\mathrm{sp}\{S\}$. Note that $\beta \in \mathrm{sp}\{S\}^*$ therefore $\beta^{-1} \in \mathrm{sp}\{S\}^*$ and so does

$\alpha_j\beta^{-1} \in \mathrm{sp}\{S\}$ for all j. Hence, $v = u_1\alpha_1\beta^{-1} + u_2\alpha_2\beta^{-1} + ... + u_i\alpha_i\beta^{-1} \in \mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\}$ which is a contradiction. Such a value a cannot exist thus $\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\} \cap \mathrm{sp}\{S_v\} = \{\, 0 \,\}$.

By construction, Algorithm 2, with Γ the multiplication by γ, will output $k \geq 1$ sets $\{\, S_{u_1}, S_{u_2}, ..., S_{u_k} \,\}$ built such that $\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_k}\} = \{\, 0, 1 \,\}^n$ thus $|\,\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_k}\}| = 2^n$. Since $|\,\mathrm{sp}\{S_{u_1}\}| = 2^h$ and at each step we choose $u_{i+1} \notin \mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\}$ therefore $\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_i}\} \cap \mathrm{sp}\{S_{u_{i+1}}\} = \{\, 0 \,\}$ we deduce by induction that $|\,\mathrm{sp}\{S_{u_1}, S_{u_2}, ..., S_{u_k}\}| = 2^{kh}$. Therefore, $kh = n$ for some $k \geq 1$ that is h divides n.

Selecting Elements. Let us describe how to build the matrix X from the sets of set $\Phi = \{\, S_{u_1}, S_{u_2}, ..., S_{u_k} \,\}$. Note that when we consider finite field multiplication, all the sets will be of the same size. Nevertheless, we describe how to pick elements of X in the general case with sets of multiple sizes with at least 2 elements.

After gathering the sets $\{\, S_{u_1}, S_{u_2}, ..., S_{u_k} \,\}$, we first deal with sets of even size $S_u = \{\, u, u\Gamma, ..., u\Gamma^{2\ell-1} \,\}$, and we simply select one out of two elements that is $X \supseteq \{\, u, u\Gamma^2, ..., u\Gamma^{2\ell-2} \,\}$ which implies that $X\Gamma \supseteq \{\, u\Gamma, u\Gamma^3, ..., u\Gamma^{2\ell-1} \,\}$. All elements of S_u are present when combining X and $X\Gamma$.

When the size of the set is odd and greater than 1, we cannot split it in two, but there is a trick to deal with two such sets of odd sizes and keep X to a minimum. Let $S_u = \{\, u, u\Gamma, ..., u\Gamma^{2\ell} \,\}$ and $S_v = \left\{\, v, v\Gamma, ..., v\Gamma^{2\ell'} \,\right\}$ two sets of odd sizes. We keep elements $X \supseteq \{\, u \oplus v \,\} \cup \{\, u\Gamma^2, u\Gamma^4, ..., u\Gamma^{2\ell} \,\} \cup \left\{\, v\Gamma, v\Gamma^3, ..., v\Gamma^{2\ell'-1} \,\right\}$ implying $X\Gamma \supseteq \{\, u\Gamma \oplus v\Gamma \,\} \cup \{\, u\Gamma^3, u\Gamma^5, ..., u\Gamma^{2\ell+1} \,\} \cup \left\{\, v\Gamma^2, v\Gamma^4, ..., v\Gamma^{2\ell'} \,\right\}$. Considering the span of all these elements combined, we see that $u\Gamma \oplus v\Gamma$ and $v\Gamma$ spans $u\Gamma$, so the span contains $\{\, u\Gamma, u\Gamma^2, ..., u\Gamma^{2\ell+1} \,\}$ which precisely spans $\mathrm{sp}\{S_u\}$ and in particular the element u; therefore v is spanned thanks to $u \oplus v$ which completes the elements of S_v. We conclude that $\mathrm{sp}\{X, X\Gamma\} \supseteq \mathrm{sp}\{S_u, S_v\}$ and X contains $1 + \ell + \ell'$ elements which is exactly half of $2\ell + 1 + 2\ell' + 1$.

Conclusion. When Γ is equivalent to a Galois field multiplication everything works flawlessly: Algorithm 2 will output equally sized sets with at least two elements that we can split evenly to build X with the minimum of $\lceil n/2 \rceil$ values.

While it is not easy to generalize the approach for all linear key schedules Γ, other choices than a $\mathrm{GF}(2^n)$ multiplication hardly seems to pose any issue. For instance, the presence of fixed points, $x\Gamma = x$, may make Algorithm 2 outputs sets with a single element or even make it impossible to build a minimally sized X. However, key schedules with many fixed points are not known to offer much security especially as a birthday-bound attack is already known when there is no key-schedule, that is when Γ is the identity. Alternatively, we choose the values u of Algorithm 2 to avoid fixed points: choosing u with a single active bit will avoid all fixed points of circular-shift key schedules and output equally sized sets.

3.3 Optimizing the Online and Time Complexities

The attack described in Section 3.1 is great as an information-theoretically tight key-recovery attack, but it still requires $\widetilde{\mathcal{O}}(2^n)$ computations which may make this cryptanalysis not practical at all even for relatively short n. We can actually describe a range of possible trade-offs by combining a few tricks on the way we build the matrices X, A_1, A_2 and the lists for the 3-XOR problem filtering the triplets. Concretely, for two parameters α and β such that $2\alpha+\beta \leq n/2$, the key recovery attack requires $2^{n/2-\beta}$ online queries, $2^{n/2+\alpha+\beta}$ offline queries, $2^{n-\alpha}$ time and $2^{n/2+\beta}$ memory.

Coupling Related Solutions. Let us first describe an optimization that increases the number of offline queries to $2^{n/2+\alpha}$ and lowers the time complexity to $\widetilde{\mathcal{O}}(2^{n-\alpha})$ for some $0 \leq \alpha \leq n/4$. In practice, we may consider offline queries as mere computations of a fully described permutation. Therefore, if one considers that computing a permutation takes 1 unit of time then, taking $\alpha = n/4$, this optimized attack requires $2^{n/2}$ chosen plaintexts (still birthday-bound), $2^{n/2}$ memory and takes $\widetilde{\mathcal{O}}\big(2^{3n/4}\big)$ computations.

The optimization mainly exploits the fact that all solutions are strongly related. Indeed, we are looking for a solution (x, a_1, a_2) such that:

$$\begin{cases} x \oplus a_1 = k \\ x \oplus a_2 = k\Gamma \end{cases} \tag{3}$$

so it is clear that if (x, a_1, a_2) satisfies Equation (3) then $\{\ (x \oplus c, a_1 \oplus c, a_2 \oplus c) : c \in \{\,0,1\,\}^n\ \}$ is the set of all solutions to Equation (3).

The idea then is to couple a set of solutions and look for this set. That is, take α linearly independent values in a set C and consider for some (x, a_1, a_2) (not necessarily a solution) the set $\{\ x \oplus c : c \in \mathrm{sp}\{C\}\ \} \times \{\ a_1 \oplus c : c \in \mathrm{sp}\{C\}\ \} \times \{\ a_2 \oplus c : c \in \mathrm{sp}\{C\}\ \}$: if it contains a solution to Equation (3) then it necessarily contains 2^α related solutions, and we have:

$$\left(\bigoplus_{c \in \mathrm{sp}\{C\}} E(x \oplus c)\right) \oplus \left(\bigoplus_{c \in \mathrm{sp}\{C\}} P_1(a_1 \oplus c)\right) \oplus \left(\bigoplus_{c \in \mathrm{sp}\{C\}} P_2(a_2 \oplus c)\right) = 0$$

This is Algorithm 3 with parameter $\beta = 0$. Notice that Step 13 of Algorithm 3 always finds a set of solutions for the same reason that Algorithm 1 always finds a solution. Indeed, as we just showed, even if there is no solution $(x, a_1, a_2) \in \mathrm{sp}\{X'\} \times \mathrm{sp}\{A_1\} \times \mathrm{sp}\{A_2\}$ it suffices that a solution $(x \oplus c, a_1, a_2)$ exists for some $c \in \mathrm{sp}\{C\}$ to pass the test of Step 13 and return the right key directly derived from a_1 and a_2.

Reducing Online Complexity. The second trick is a simple way to reduce the online query complexity at the direct expanse of the offline query complexity. The idea is to "transfer" a dimension from $\mathrm{sp}\{X\}$ to both $\mathrm{sp}\{A_1\}$ and $\mathrm{sp}\{A_2\}$. Concretely, this means removing a value v (that is a row) from X and adding it to A_1 and A_2. It is clear that if $(x, a_1, a_2) \in \mathrm{sp}\{X\} \times \mathrm{sp}\{A_1\} \times \mathrm{sp}\{A_2\}$ is a

solution to Equation (3) then so is $(x \oplus v, a_1 \oplus v, a_2 \oplus v)$ and since one of x or $x \oplus v$ belongs to $\mathrm{sp}\{X\} \setminus \{ v \}$ we have that $\mathrm{sp}\{X\} \setminus \{ v \} \times \mathrm{sp}\{A_1, v\} \times \mathrm{sp}\{A_2, v\}$ also contains a unique solution. We can thus "transfer" multiple rows, β rows for instance, of X, and keep the property that we have a unique solution. This is Algorithm 3 with parameter $\alpha = 0$. This modification directly reduces the number of online queries to $2^{n/2-\beta}$ while increasing the number of offline queries to both permutations, as well as the memory to $2^{n/2+\beta}$.

Algorithm 3 Optimized Key Recovery

1: **input:** $(\alpha, \beta : 2\alpha + \beta \leq n/2), \Gamma, P_1, P_2, E$.
2: **output:** $k : E(x) = P_1(x \oplus k) \oplus P_2(x \oplus k\Gamma)$.
3: **procedure** KeyExtractor(P_1, P_2, E)
4: Find $n/2$ by n bit matrix X such that $\mathrm{sp}\{X\Gamma, X\} = \{0,1\}^n$. ▷ See Section 3.2.
5: $A_1 \leftarrow X(I \oplus \Gamma^{-1})$
6: $A_2 \leftarrow X(I \oplus \Gamma)$
7: $\begin{bmatrix} C_{\alpha \times n} \\ D_{\beta \times n} \\ X'_{(n/2-\alpha-\beta) \times n} \end{bmatrix} \leftarrow X$ ▷ Split the values of X between C, D and X'.
8: $A'_1 \leftarrow \begin{bmatrix} D \\ A_1 \end{bmatrix}$ ▷ A'_1 and A'_2 are $(n/2+\beta) \times n$ matrices.
9: $A'_2 \leftarrow \begin{bmatrix} D \\ A_2 \end{bmatrix}$
10: $L_{X'} \leftarrow \left\{ e_1(x) = \bigoplus_{c \in \mathrm{sp}\{C\}} E(x \oplus c) : x \in \mathrm{sp}\{X'\} \right\}$
11: $L_{A'_1} \leftarrow \left\{ e_2(a_1) = \bigoplus_{c \in \mathrm{sp}\{C\}} P_1(a_1 \oplus c) : a_1 \in \mathrm{sp}\{A'_1\} \right\}$
12: $L_{A'_2} \leftarrow \left\{ e_3(a_2) = \bigoplus_{c \in \mathrm{sp}\{C\}} P_2(a_2 \oplus c) : a_2 \in \mathrm{sp}\{A'_2\} \right\}$
13: $\Phi \leftarrow \left\{ (e_1(x), e_2(a_1), e_3(a_2)) \in L'_X \times L_{A'_1} \times L_{A'_2} : e_1 \oplus e_2 \oplus e_3 = 0 \right\}$ ▷ 3-XOR
14: **for all** $(e_1(x), e_2(a_1), e_3(a_2)) \in \Phi$ **do**
15: $\hat{k} \leftarrow (a_1 \oplus a_2)(I \oplus \Gamma)^{-1}$
16: **if** $\hat{k}$ is the key **then** ▷ Test with a few stored online queries
17: **return** $\hat{k}$
18: **end if**
19: **end for**
20: **end procedure**

Complexity Analysis. The two mentioned tricks can trivially be combined and the resulting key recovery attack is described in Algorithm 3. The main difference with Algorithm 1 is that now the lists are unbalanced: $L_{X'}$ contains $2^{n/2-\alpha-\beta}$ elements while $L_{A'_1}$ and $L_{A'_2}$ contains $2^{n/2+\beta}$ elements. First, notice that the number of false positives passing through the filter of Step 13 is negligible with regard to the total complexity as we test $2^{1.5n-\alpha+\beta}$ triplets with an n-bit filter, so we expect only about $2^{n/2-\alpha+\beta}$ false positives. Moreover, as in Algorithm 1,

the bottleneck regarding the time complexity is in the filtering process itself. Again, the best algorithms solving the 3-XOR problem have a time complexity comparable (ignoring log factors) to combining the two shortest lists and solve a classical collision problem. Hence, for instance combining $L_{X'}$ and $L_{A'_1}$, we get a time complexity of $\widetilde{\mathcal{O}}(2^{n-\alpha})$.

By construction of the lists, Algorithm 3 requires $2^{n/2-\beta}$ online queries and $2^{n/2+\alpha+\beta}$ offline queries. As the time complexity cannot be lower than the query complexity (a query has to be read at least), the total time complexity is thus $\widetilde{\mathcal{O}}\big(\max(2^{n-\alpha}, 2^{n/2+\alpha+\beta})\big)$. The complexity profile seems to strictly worsen as the offline queries dominate the time complexity. Therefore, we only look at positive α, β parameters such that $2\alpha + \beta \leq n/2$ where the total time complexity is indeed $\widetilde{\mathcal{O}}(2^{n-\alpha})$.

The extreme case $\alpha = n/4$, $\beta = 0$ optimizes the most the time complexity which becomes $\widetilde{\mathcal{O}}\big(2^{3n/4}\big)$ while the other extreme $\alpha = 0$, $\beta = n/2$ only requires a handful of known plaintexts but is actually equivalent to the trivial brute-force approach.

4 Cryptanalysis Examples

In this section, we show two examples that help readers understand our attack. Independently of the actual key, it is the set-up that we show with concrete settings. After describing the queries required for the attack we show that there will be a successful triplet for any key.

4.1 Attack in GF(2^8) with $\gamma = 2$

Let us consider the finite field used in AES [1] which is GF(2^8) with feedback polynomial $x^8 = x^4 + x^3 + x + 1$. A multiplication by $\gamma = 2$ in that field is equivalent to a bit-matrix multiplication by Γ defined as:

$$\Gamma = \begin{bmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&1&0&1&1&0&0&0 \end{bmatrix}$$

Set Gathering. Now if we run the Set gathering Algorithm 2 starting with the element $u = [10000000]$ it's easy to see that we get a collection Φ containing a single set that contains all vectors with a single active bit that is the identity matrix $\Phi = \{ I_8 \}$. Then we simply build X by picking one element of Φ out of

two and compute $A_1 = X(I_8 \oplus \Gamma^{-1})$ and $A_2 = X(I_8 \oplus \Gamma)$:

$$X = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0 \end{bmatrix} \quad A_1 = \begin{bmatrix} 0&0&1&1&0&0&0&1\\ 0&1&1&0&0&0&0&0\\ 0&0&0&1&1&0&0&0\\ 0&0&0&0&0&1&1&0 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1&1&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&1&1 \end{bmatrix}$$

Key Recovery. Therefore, the online, P_1 and P_2 queries are of the form $x = e_x X$, $a_1 = e_1 A_1$ and $a_2 = e_2 A_2$, respectively for all $e_x, e_1, e_2 \in \{0,1\}^4$. In fact, we can explicitly write the solution for any 8-bit key $k = [k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7]$:

$$\begin{aligned} e_x &= [k_0,\ k_1 + k_2 + k_7,\ k_3 + k_4 + k_7,\ k_5 + k_6] \\ e_1 &= [k_7,\ k_1,\ k_3 + k_7,\ k_5] \\ e_2 &= [k_0 + k_7,\ k_2 + k_7,\ k_4,\ k_6] \end{aligned}$$

We have indeed $e_x X \oplus e_1 A_1 = k$ and $e_x X \oplus e_2 A_2 = k\Gamma = 2k$.

4.2 Attack in GF(2^9) with $\gamma = 2^{73}$

Let us now demonstrate how we can set up the attack in the finite field GF(2^9) with feedback polynomial $x^9 = x^8 + x^5 + x^4 + 1$. The key schedule is a multiplication by $\gamma = 2^{73}$ which is equivalent to a bit-matrix multiplication by Γ defined as:

$$\Gamma = \begin{bmatrix} 1&0&1&1&1&1&1&0&1\\ 1&1&0&1&0&0&1&1&1\\ 1&1&1&0&0&1&0&1&0\\ 0&1&1&1&0&0&1&0&1\\ 1&0&1&1&0&1&0&1&1\\ 1&1&0&1&0&1&1&0&0\\ 0&1&1&0&1&0&1&1&0\\ 0&0&1&1&0&1&0&1&1\\ 1&0&0&1&0&1&1&0&0 \end{bmatrix}$$

This setting is interesting because of the following corollary:

Corollary 1. *In GF(2^9), for all primitive elements α, either $\alpha^{73} = \alpha^{146} \oplus \alpha^{292}$ or $\alpha^{73} = \alpha^{219} \oplus \alpha^{292}$.*

Proof of Corollary 1. Since $2^9 - 1 = 511 = 73 \times 7$ we deduce that $\alpha^{511} = 1$ and thus by repeatedly multiplying by α^{73} we obtain $1 \overset{\times\alpha^{73}}{\to} \alpha^{73} \overset{\times\alpha^{73}}{\to} \alpha^{146} \overset{\times\alpha^{73}}{\to} \alpha^{219} \overset{\times\alpha^{73}}{\to} \alpha^{292} \overset{\times\alpha^{73}}{\to} \alpha^{365} \overset{\times\alpha^{73}}{\to} \alpha^{438} \overset{\times\alpha^{73}}{\to} 1$ a chain with 7 different values. From Theorem 5, those values span a dimension h that divides $n = 9$. As it contains 7 distinct elements it cannot span 9 dimensions nor can it span only 1 dimension, so it has to span 3 dimension; that is α^{292} can be written as a sum involving $\{\alpha^{73}, \alpha^{146}, \alpha^{219}\}$.

α^{73} has to appear in the sum to span 3 dimensions; $\alpha^{73} = \alpha^{292}$ is impossible because there are distinct; $\alpha^{73} = \alpha^{146} \oplus \alpha^{219} \oplus \alpha^{292}$ implies that $\alpha^{365} = \alpha^{146} \oplus \alpha^{219} \oplus \alpha^{292} = \alpha^{73}$ which is again impossible. Therefore, only two possibilities are left to Corollary 1.

Set Gathering. Since 2 is a primitive element of this field, according to Corollary 1 we know that the Set gathering Algorithm 2 will output a collection of sets containing three values. In fact in our case we have $2^{73} = 2^{219} \oplus 2^{292}$ (eq. $\Gamma^4 = \Gamma^3 \oplus \Gamma$). Choosing successively the starting points u with Hamming weight 1 we get a Φ containing three sets of three elements:

$$\Phi = \begin{cases} \Big[[100000000]\ [101111101]\ [101101111]\Big] \\ \Big[[010000000]\ [110100111]\ [110101110]\Big] \\ \Big[[001000000]\ [111001010]\ [011010111]\Big] \end{cases}$$

All 9 elements of Φ indeed span $\{0,1\}^9$.

In order to get a minimal working set X, we do the trick of adding to X the sum of the first elements of the two first sets followed with the third and second of the first and second sets, respectively. Lastly we add two elements of the last set to X which makes for a minimal matrix with 5 values, and we compute $A_1 = X(I_8 \oplus \Gamma^{-1})$ and $A_2 = X(I_8 \oplus \Gamma)$:

$$X = \begin{bmatrix} 1&1&0&0&0&0&0&0&0 \\ 1&0&1&1&0&1&1&1&1 \\ 1&1&0&1&0&0&1&1&1 \\ 0&0&1&0&0&0&0&0&0 \\ 0&1&1&0&1&0&1&1&1 \end{bmatrix} \quad A_1 = \begin{bmatrix} 1&1&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&0 \\ 1&0&0&1&0&0&1&1&1 \\ 1&0&1&0&1&1&1&0&1 \\ 1&0&0&0&1&1&1&0&1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1&0&1&0&1&1&0&1&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&1 \\ 1&1&0&0&0&1&0&1&0 \\ 0&0&1&0&0&0&0&0&0 \end{bmatrix}$$

Key Recovery. In the odd case such as $n = 9$, after building a minimal X and computing the corresponding A_1 and A_2, there always exists a value in X that we can drop without affecting the span $\mathrm{sp}\{X, A_1\}$ to reduce the online queries. In our case, let $X' = X \setminus \{\ [0\,0\,1\,0\,0\,0\,0\,0\,0]\ \}$; note that $\mathrm{sp}\{X', A_1\} = \{0,1\}^9$ as the removed element is actually equal to the sum of the 4th and 5th elements of A_1.

Therefore, the online, P_1 and P_2 queries are of the form $x = e_x X'$, $a_1 = e_1 A_1$ and $a_2 = e_2 A_2$, respectively for all $e_x \in \{0,1\}^4$ and $e_1, e_2 \in \{0,1\}^5$. In fact, we explicitly write the solution for any 9-bit key $k = [k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8]$:

$$\begin{aligned} e_x = [&k_0 + k_4 + k_6 + k_7 + k_8,\ k_3 + k_4 + k_5 + k_6 + k_7 + k_8, \\ &k_0 + k_1 + k_6,\ k_4 + k_6 + k_7] \\ e_1 = [&k_6 + k_8,\ k_3 + k_4 + k_8,\ k_0 + k_1 + k_4 + k_5 + k_7 + k_8, \\ &k_2 + k_3 + k_5 + k_8,\ k_2 + k_4 + k_5 + k_7 + k_8] \\ e_2 = [&k_0 + k_4 + k_7,\ k_5 + k_6 + k_7,\ k_4 + k_5 + k_6 + k_7 + k_8, \\ &k_2 + k_3 + k_5 + k_8,\ k_2 + k_5 + k_6 + k_8] \end{aligned}$$

We have indeed $e_x X' \oplus e_1 A_1 = k$ and $e_x X' \oplus e_2 A_2 = k\Gamma = 2^{73}k$.

References

1. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce (Nov 2001)
2. Babai, L.: The fourier transform and equations over finite abelian groups. Lecture Notes, version 1(1) (1989)
3. Bellare, M., Krovetz, T., Rogaway, P.: Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 266–280. Springer, Heidelberg (May / Jun 1998)
4. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (May / Jun 2006)
5. Bhattacharya, S., Nandi, M.: Full indifferentiable security of the xor of two or more random permutations using the χ^2 method. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 387–412. Springer, Heidelberg (Apr / May 2018)
6. Bogdanov, A., Knudsen, L.R., Leander, G., Standaert, F.X., Steinberger, J.P., Tischhauser, E.: Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations - (extended abstract). In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 45–62. Springer, Heidelberg (Apr 2012)
7. Bouillaguet, C., Delaplace, C., Fouque, P.A.: Revisiting and improving algorithms for the 3xor problem. IACR Trans. Symm. Cryptol. 2018(1), 254–276 (2018)
8. Chen, S., Lampe, R., Lee, J., Seurin, Y., Steinberger, J.P.: Minimizing the two-round Even-Mansour cipher. Journal of Cryptology 31(4), 1064–1119 (Oct 2018)
9. Chen, S., Steinberger, J.P.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (May 2014)
10. Chen, Y.L., Lambooij, E., Mennink, B.: How to build pseudorandom functions from public random permutations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 266–293. Springer, Heidelberg (Aug 2019)
11. Cogliati, B., Lampe, R., Patarin, J.: The indistinguishability of the XOR of k permutations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 285–302. Springer, Heidelberg (Mar 2015)
12. Cogliati, B., Seurin, Y.: EWCDM: An efficient, beyond-birthday secure, nonce-misuse resistant MAC. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 121–149. Springer, Heidelberg (Aug 2016)
13. Coron, J.S., Holenstein, T., Künzler, R., Patarin, J., Seurin, Y., Tessaro, S.: How to build an ideal cipher: The indifferentiability of the Feistel construction. Journal of Cryptology 29(1), 61–114 (Jan 2016)
14. Dachman-Soled, D., Katz, J., Thiruvengadam, A.: 10-round Feistel is indifferentiable from an ideal cipher. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 649–678. Springer, Heidelberg (May 2016)
15. Dai, Y., Steinberger, J.P.: Indifferentiability of 8-round Feistel networks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 95–120. Springer, Heidelberg (Aug 2016)
16. Data encryption standard. National Bureau of Standards, NBS FIPS PUB 46, U.S. Department of Commerce (Jan 1977)

17. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: The Even-Mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (Apr 2012)
18. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT'91. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (Nov 1993)
19. Gaži, P., Tessaro, S.: Secret-key cryptography from ideal primitives: A systematic overview. In: 2015 IEEE Information Theory Workshop (ITW). pp. 1–5 (2015)
20. Gentry, C., Ramzan, Z.: Eliminating random permutation oracles in the Even-Mansour cipher. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 32–47. Springer, Heidelberg (Dec 2004)
21. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 31–49. Springer, Heidelberg (Aug 2012)
22. Jha, A., Nandi, M.: A survey on applications of h-technique: Revisiting security analysis of prp and prf. Entropy 24(4) (2022)
23. Lampe, R., Seurin, Y.: Security analysis of key-alternating Feistel ciphers. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 243–264. Springer, Heidelberg (Mar 2015)
24. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17(2) (1988)
25. Mennink, B., Neves, S.: Encrypted Davies-Meyer and its dual: Towards optimal security using mirror theory. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 556–583. Springer, Heidelberg (Aug 2017)
26. Mennink, B., Preneel, B.: On the XOR of multiple random permutations. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 15. LNCS, vol. 9092, pp. 619–634. Springer, Heidelberg (Jun 2015)
27. Patarin, J.: The "coefficients H" technique (invited talk). In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (Aug 2009)
28. Pieprzyk, J.: How to construct pseudorandom permutations from single pseudorandom functions. In: Damgård, I. (ed.) EUROCRYPT'90. LNCS, vol. 473, pp. 140–150. Springer, Heidelberg (May 1991)
29. Shinagawa, K., Iwata, T.: Quantum attacks on sum of even-mansour pseudorandom functions. Information Processing Letters 173, 106172 (2022)
30. Steinberger, J.P.: The sum-capture problem for abelian groups. arXiv preprint arXiv:1309.5582 (2013)