J Cryptol (2016) 29:697–728
DOI: 10.1007/s00145-015-9207-3

Key Recovery Attacks on Iterated Even–Mansour Encryption Schemes

Itai Dinur*
Département d'Informatique, École Normale Supérieure, Paris, France
itai.dinur@ens.fr

Orr Dunkelman†
Computer Science Department, University of Haifa, Haifa, Israel

Nathan Keller‡
Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel

Orr Dunkelman · Nathan Keller · Adi Shamir
Computer Science Department, The Weizmann Institute, Rehovot, Israel

Communicated by Serge Vaudenay.

Received 26 December 2013
Online publication 18 April 2015

Abstract. Iterated Even–Mansour (EM) encryption schemes (also named "key-alternating ciphers") were extensively studied in recent years as an abstraction of commonly used block ciphers. A large amount of previous works on iterated EM concentrated on security in an *information-theoretic* model. A central question studied in these papers is: What is the minimal number of rounds for which the resulting cipher is indistinguishable from an ideal cipher? In this paper, we study a similar question in the *computational* model: What is the minimal number of rounds, assuring that no attack can recover the secret key faster than trivial attacks (such as exhaustive search)? We study this question for the two natural key scheduling variants that were considered in most previous papers: the *identical subkeys* variant and the *independent subkeys* variant. In the identical subkeys variant, we improve the best known attack by an additional round and show that $r = 3$ rounds are insufficient for assuring security, by devising a key recovery attack whose running time is about $n/\log(n)$ times faster than exhaustive search for an n-bit key. In the independent subkeys variant, we also extend the known results by one round and show that for $r = 2$, there exists a key recovery attack whose running time is

This paper in an extended version of [11], presented at ASIACRYPT 2013.

* Some of the work presented in this paper was done while the first author was a postdoctoral researcher at the Weizmann Institute, Israel.

† The second author was supported in part by the German-Israeli Foundation for Scientific Research and Development through Grant No. 2282-2222.6/2011.

‡ The third author was supported by the Alon Fellowship.

faster than the benchmark meet-in-the-middle attack. Despite their generic nature, we show that the attacks can be applied to improve the best known attacks on several concrete ciphers, including the full AES^2 (proposed at Eurocrypt 2012) and reduced-round LED-128 (proposed at CHES 2012).

Keywords. Cryptanalysis, Key recovery attacks, Iterated Even–Mansour, LED block cipher, AES^2 block cipher, Backdoors in cryptography.

1. Introduction

1.1. *Background*

The Even–Mansour cryptosystem was first proposed at Asiacrypt 1991 [15] in an attempt to obtain the simplest possible block cipher. It uses a single publicly known permutation P on n-bit values and two secret n-bit keys K_1 and K_2 and defines the encryption of the n-bit plaintext m as $E(m) = P(m \oplus K_1) \oplus K_2$. The decryption of an n-bit ciphertext c is similarly defined as $D(c) = P^{-1}(c \oplus K_2) \oplus K_1$. The construction can be naturally generalized into an r-round iterated EM encryption function (also called a key-alternating scheme in [1,7] and other papers), which is defined using r permutations $P_1, P_2, \ldots, P_r$ and $r + 1$ keys $K_1, K_2, \ldots K_{r+1}$ as

$$E(m) = P_r(\ldots P_2(P_1(m \oplus K_1) \oplus K_2) \ldots \oplus K_r) \oplus K_{r+1},$$

where decryption is defined in an analogous way.

For about 20 years, this scheme received little attention in the cryptographic literature, but since 2011 it became a very active research area. One possible explanation for this is the very simple key schedule of iterated EM schemes. This makes them an attractive choice for block ciphers with low resource consumption, which became a very important design goal in recent years, with the rise of lightweight cryptography. Some of these lightweight designs, such as LED-64 (presented at CHES 2011 [20]) and Zorro (presented at CHES 2013 [18]), have no key schedule at all, but instead, use several public permutations, separated by additions of the same key, namely an iterated EM with identical subkeys.

As several concrete instances of iterated EM schemes were proposed, many papers (e.g., [13,27,29,34]) devised various attacks on these primitives. Several of these attacks exploit specific properties of the internal permutations of the primitives and are thus restricted to specific instances. Other attacks target (almost) all primitives with a specific key schedule. These attacks are more general and work regardless of the choice of the internal permutations of the scheme (typically by assuming that the permutations are chosen uniformly at random).

An additional line of work in the analysis of iterated EM schemes takes an information-theoretic approach and is mostly aimed at formally proving the security of these schemes by bounding the information that is available to the attacker. The security analysis of iterated EM schemes can thus be roughly divided into two types:

– *Computational* In this analysis model, the focus is on algorithms that break a given iterated EM scheme (usually by recovering the secret key) and measure their complexity by counting the number of operations they perform. When we analyze the security of a given iterated EM scheme in this model, we are thus interested in what is the most computationally efficient algorithm that can break it.

– *Information-Theoretic* In this analysis model, the focus is on the amount of information that is required in order to distinguish a given iterated EM scheme from some ideal construction (such as a random permutation). When we analyze the security of a given iterated EM scheme in this model, we are thus interested in counting the number of queries to its internal permutations that are required in order to distinguish the scheme from the ideal construction.

The main motivation behind the information-theoretic model is that (similarly to many problems in computer science) it is very difficult to prove lower bounds on the complexity of algorithms in the computational model, while proving lower bounds in the information-theoretic model is generally much easier. As lower bounds in the information-theoretic model can often serve as bounds in the computational model, these bounds may also contribute to understanding security in the computational model. Indeed, in the paper introducing the EM scheme [15], Even and Mansour proved an information-theoretic bound that any attack on the scheme with n-bit keys must satisfy $DT = \Omega(2^n)$ (where D is the data complexity and T is the number of queries to the permutation, which serves as a lower bound on the time complexity of the algorithm). Shortly after the introduction of the scheme, Daemen [9] presented an attack matching the bound $DT = O(2^n)$ (where T is the actual time complexity of the algorithm), though in the chosen-plaintext model. Twenty years later, Dunkelman et al. [14] showed that the same bound holds also in the known-plaintext model, thus fully determining the security of EM in the key recovery model. Furthermore, Dunkelman et al. showed that the single-key variant of EM, defined as $E(m) = P(m \oplus K) \oplus K$, provides exactly the same security level.

The information-theoretic analysis of iterated EM schemes was initiated At Eurocrypt 2012 by Bogdanov et al. [7]. This study was pursued in a series of papers, including [1, 8, 24, 25, 35], and is still developing. However, we emphasize that for small values of $r > 1$, there is a significant gap between the lower bounds obtained in the information-theoretic model, which is relevant in the computational model, and the complexity of the corresponding best known algorithms. For example, for an iterated EM scheme with identical subkeys and $r = 2$, the information-theoretic lower bound (the natural extension of the bound for $r = 1$) is $2^{2n/3}$ [7], whereas the complexity of the best known algorithm so far is only about $2^n/n$ [29]. Thus, in the information-theoretic model, small values of r seem less interesting, and the main investigated question is what is the minimal r, assuring that the encryption scheme cannot be distinguished from an ideal cipher in complexity of $\Omega(2^n)$. Such a value of r can also be meaningful in the computational model, although the true minimal value of r that ensures that the scheme is secure in the computational model is perhaps significantly lower.

1.2. *Our Contribution*

In this paper, we analyze iterated EM schemes in the computational model. As the analysis strongly depends on the key scheduling algorithm used to derive the subkeys $K_1, \ldots, K_{r+1}$ from the secret key K, and there are many possible key schedules, we concentrate on the two most natural variants:

- *The independent subkeys variant*, in which the subkeys K_1, $K_2, \dots, K_{r+1}$ are fully independent, such that the effective key size is $(r+1)n$ bits. This variant is arguable the most natural generalization of the original EM scheme, and it was studied in most of the theoretic papers on iterated EM constructions [7,8,24,35].
- *The identical subkeys variant*, in which $K_1 = K_2 = \cdots = K_{r+1} = K$. This variant is the natural generalization of the single-key variant of EM defined in [14], and it was studied in [1,25].

The main question we study is similar to the main question studied in [1,25] in the ideal cipher model, namely what is the minimal number r such that r-round iterated EM is indifferentiable from an ideal cipher. The question we study can be informally stated as follows:

Question 1 *What is the minimal number r such that r-round iterated EM provides "full security" with respect to key recovery attacks?*

The precise meaning of "full security" in this context depends on the key scheduling algorithm. In the identical subkeys iterated EM variant, the "trivial" attack is exhaustive key search, requiring $O(2^n)$ time for an n-bit key. Hence, we say that the scheme provides full security if any attack on it requires $\Omega(2^n)$ time, data or memory. In the independent subkeys variant, the currently best known generic attack is an obvious meet-in-the-middle attack, which can break the scheme in $O(2^{\lfloor r/2 \rfloor n})$ time, and hence, the scheme can be considered secure if any attack on it requires $\Omega(2^{\lfloor r/2 \rfloor n})$ time, data or memory. We say that a cipher *provides m-bit security* if any attack on it requires $\Omega(2^m)$ time, data or memory. In these terms, the question above can be formulated as

Question 1 **(Reformulated)** *What is the minimal r such that r-round iterated EM with n-bit subkeys provides n-bit security (in the identical subkeys variant), or $\lfloor r/2 \rfloor n$-bit security (in the independent subkeys variant)?*

We emphasize that we can only give lower bounds on the value of r by presenting efficient attacks. In order to upper bound r, we need to prove a lower bound on the computation complexity of all algorithms that try to break the corresponding scheme. However, as previously mentioned, this is considered a very difficult task, as we lack such proof techniques in the computational model.

Our main results are the following:

Conclusion 1. *The minimal number of rounds, r_1, such that an r_1-round iterated EM with identical n-bit subkeys provides n-bit security is $r_1 \geq 4$.*

We support this claim by devising a key recovery attack on 3-round EM with identical subkeys, which is $n/\log n$ times faster than exhaustive key search. This result improves by a full round the best previously known result [29] that can break 2-round EM (with roughly the same complexity as our 3-round attack).

Conclusion 2. *The minimal number of rounds, r_2, such that an r_2-round iterated EM with independent n-bit subkeys provides n-bit security is $r_2 \geq 3$.*

We support this claim by devising a key recovery attack on 2-round EM with independent subkeys whose overall complexity is $o(2^n)$. This is the first result on this variant which succeeds to go further than the twenty-year-old attacks of [9] on the original EM scheme.

After examining the two main iterated EM variants, we present applications of our techniques to three other EM variants that may be naturally used by concrete ciphers.

– *Iterated EM with Random Involutions*: Involutions are functions that satisfy $f(f(x)) = x$ for all x. They are used as building blocks in several block ciphers (such as KHAZAD [4], Anubis [3] and NOEKEON [10]). Involutions were already considered for EM schemes with identical subkeys in [14], which showed that choosing the permutation to be a random involution does not reduce its resistance to key recovery attacks. We show that contrary to the EM case, in iterated EM schemes with identical subkeys, choosing the P_i's to be random involutions reduces the security considerably. Indeed, while the attacks on 2-round and 3-round iterated EM with an n-bit key require about $O(2^{n-\log n})$ time, in the involutional case the complexity reduces drastically to $O(2^{n/2})$ and $O(2^{3n/4})$, respectively. A rather surprising implication of this result is that one can *reduce* the security of a 2-round iterated EM scheme by pre-pending an additional round whose permutation is a (possibly strong) involution (e.g., a keyless Feistel construction with no round constants). This reduction in security can be considered as a way to insert a backdoor into iterated EM constructions.
– *Iterated EM with Alternating Subkeys*: One of the possible ways to instantiate iterated EM with block size of n bits and key size of $2n$ bits (which is quite common, e.g., AES-256 and IDEA) is to divide the master key K into two n-bit subkeys K_1, K_2 and use them alternately. This design is used in the block cipher LED-128 [20]. We adapt our attack on 3-round EM with identical subkeys to an attack on 8-round EM with alternating subkeys, thus showing that the minimal number of rounds required in alternating subkeys EM for providing $2n$-bit security is at least 9. The best previous result on this scheme was an attack on 6 rounds [29] (resulting from an adaptation of an attack on 2-round EM with identical subkeys).
– *Iterated EM with Linearly Derived Subkeys*: In numerous block ciphers, the round subkeys are not identical, but are all derived from the master key in a linear way. Examples include DES, IDEA and SHACAL-1. We show that our attacks on iterated EM with identical subkeys can be adapted to work with the same complexity on iterated EM in which the subkeys are linearly derived from the master key.

Although our results are generic and do not depend on the structure of the permutations P_i (with the exception of involutional schemes), they provide the best known attacks against several concrete block ciphers. These ciphers include both variants of LED (LED-64 and LED-128), proposed by Guo et al. at CHES 2011 [20] and AES^2, presented at Eurocrypt 2012 by Bogdanov et al [7]. In addition, we present an application of our techniques to 8 rounds (out of 12) of the block cipher Crypton (a former AES candidate), which has relatively weak round functions, making it somewhat different from the other ciphers we consider. While 8-round Crypton can be attacked with other techniques (as shown in [22,28]), it is of a particular interest, as it demonstrates that our generic techniques can be competitive with highly specialized attacks even on block ciphers with weak round functions (for which one may expect specialized techniques to reach more rounds than our generic techniques). Additionally, our attack is generic and can work even if the round function of Crypton is replaced by a very strong function.

We also mention the block cipher Zorro (presented at CHES 2013 by Gérard et al. [18]), whose preliminary version was a 3-round EM with identical subkeys. Due to our attack (communicated to the designers), the designers decided to increase its number of rounds to 6, in exchange for simplifying the permutations used in the rounds (see [19, page12]).[1]

1.3. *Our Techniques*

The origin of our techniques is the first paper which analyzed EM, by Daemen [9] in 1991. Daemen observed that in single-key EM, an attacker can use the fact that the XOR of the unknown input and output of the permutation P is equal to the known XOR of the plaintext and the ciphertext. This observation can be used to break single-key EM significantly faster than exhaustive search.

At FSE 2013, Nikolić et al. [29] extended the basic observation considerably. They considered the graph of the function $P'(x) = x \oplus P(x)$ [2] and showed that vertices with a large in-degree in this graph can be exploited to bypass an additional round of EM. Thus, [29] described the first key recovery attack on 2-round EM with identical subkeys, which is faster than exhaustive search (by a small factor).

In this paper, we develop these techniques one step further and show that the graphs of the functions P_1' and P_3' (corresponding to the permutations P_1 and P_3) can be deployed simultaneously, resulting in the first attack on 3-round EM with identical subkeys. However, this enhancement by itself increases the time complexity of the 2-round attack further, and it becomes very close to exhaustive key search. Nevertheless, a surprising feature of our attack on 3-round EM is that it has about the same time complexity as the 2-round attack. This feature is due to a novel filtering technique (which is related to the splice-and-cut technique [2]) based on tailor-made linear subspaces that offers an efficient method to quickly dispose of data which is useless for our attack.

Another novel technique that we develop in this paper allows us to adapt the differential-based attack of [27] (which was originally applied to 2-round iterated EM with identical subkeys) to the more complex case of 2-round iterated EM with independent subkeys. While the attack of [27] makes use of plaintext pairs with a fixed difference, we notice that in its original form, it is not faster than the standard meet-in-the-middle attack on this scheme. In our attack, we work on nonstandard structures of plaintext triplets which offer a more efficient method to filter out wrong key guesses.

[1] We note that the main motivation for most of the results described above is to understand the theoretic security of the schemes we consider. As in many previous cryptanalytic attacks, it is very likely that practitioners will prefer to use a straightforward exhaustive search due to its simplicity, lower memory complexity and parallelizability. In fact, the same distinction between theory and practice applies to numerous other computational tasks such as matrix multiplication, in which the best theoretic algorithms are not likely to be used by practitioners.

[2] In [29], the permutation P is actually the full encryption function, and thus, x is a plaintext and $P(x)$ is its corresponding ciphertext.

1.4. *Organization of the Paper*

The paper is organized as follows. In Sect. 2, we overview previous attacks on iterated EM constructions and their relation to our work, while in Sect. 3, we describe our complexity model. In Sect. 4, we present our attacks on iterated EM with identical subkeys. Our attacks on iterated EM with independent subkeys are presented in Sect. 5. In Sect. 6, we consider other variants of iterated EM, as described above. The applications of our techniques to the block ciphers LED-64, LED-128, AES^2 and Crypton are presented in Sect. 7. Finally, we conclude the paper in Sect. 8.

2. Related Work

In order to understand the context of our results on iterated EM constructions, we elaborate further on the previous works in the computational model defined in the Introduction and emphasize their relation to our work.[3] As previously mentioned, the security of the original EM scheme (i.e., with $r = 1$) is completely determined, and thus, we concentrate on iterated EM schemes (i.e., with $r > 1$).

In this paper, we are mainly interested in attacks which are not cipher-specific and can thus be applied to (almost) any iterated EM construction (with some fixed key schedule). The first class of such attacks was presented by Mendel et al. [27], which explored ways to generalize Daemen's differential attack on the EM construction [9] to related-key attacks on iterated EM constructions. For iterated EM with independent subkeys, Mendel et al. showed that Daemen's attack can be generalized to yield a related-key attack on r rounds with complexity of $O(r2^{n/2})$ (where n is the subkey size).[4] For iterated EM with identical subkeys, Mendel et al. showed that Daemen's attack can be used to break two rounds with complexity of $O(2^{n/2})$ in the related-key model. Finally, they also presented some attacks in the single-key and related-key models, which assume that some of the internal permutations exhibit a high probability differential characteristic. The related-key attacks are mostly incomparable to our results, as we consider the more standard single-key attack model throughout our paper.

The second class of generic attacks was presented by Nikolić et al. [29] in their work on LED. They considered iterated EM with identical subkeys and showed that 2 rounds are not sufficient to provide full security. The attack model considered in [29] is similar to ours, and we succeed to extend the results of [29] by a full round, describing the first attack on 3-round iterated EM (which has roughly the same complexity as the attack of [29] on 2-round EM). In addition, we present improved attacks on 2-round iterated EM, improving the attack of [29] in both data and memory complexities.

More recently, in [12], the authors studied the security of certain iterated EM variants with 2 subkeys, and in particular with alternating subkeys of the form $K_1, K_2, K_1, K_2, \ldots$. This study is driven by applications that use this specific instantiation of iterated EM, such as the LED-128 block cipher. This work studied variants with up to 4 rounds

[3] The works in the information-theoretic model are not directly related to our work since they assume that the time complexity of the adversary is unbounded (and the only bound is on the number of queries). Hence, they are not reviewed in this paper.

[4] It should be noted that a similar attack was presented in [7].

and showed that 4-round iterated EM with alternating n-bit keys provides at most n-bit security (compared to $O(2^{2n})$ of exhaustive search). In our paper, we do not consider this EM variant separately (since it is less natural from the theoretic point of view), but rather show that our attack on 3-round EM with identical subkeys can be easily adapted to give an attack on 8-round EM with alternating subkeys with complexity of $O(2^{2n-\log n})$. As [12] considered an iterated EM variant with significantly less rounds (4 vs. 8), but provided attacks with much lower complexity, their results are incomparable with the results of the this paper (as our main motivation is to investigate Question 1).

3. Our Complexity Model

Throughout the paper, we follow the standard conventions in the analysis of time and memory complexities. Our basic unit of memory is an n-bit block. Our basic unit of time is a single evaluation of the encryption or the decryption function, i.e., the full r-round iterated EM scheme. The full encryption function requires the evaluation of the r permutations P_i (which are assumed to be heavy operations) and a small number of simple operations (such as XORs), which are assumed to require negligible time. Thus, we assume that an invocation of a single permutation P_i (or its inverse) costs $1/r$ time units.

For the sake of convenience, we often partition the attack into an *offline preprocessing phase*, which analyzes the properties of the public P_i's, and an *online attack phase* which analyzes the given plaintexts and ciphertexts. However, we always define the time complexity of the attack as the sum of the complexities of its offline and online phases. This is different from the model used by Hellman in his time/memory trade-off attack [21], which allowed unlimited free preprocessing and considered only the online complexity (note that in our model, Hellman's attack is not better than exhaustive search). To prevent other types of "cheating," we always add the time required to generate the data to the final time complexity and add the space required to hold the data to the final space complexity.

All of our attacks are better than exhaustive search by small factors, which raises the natural question whether they should be considered as legitimate attacks. This is a general problem in cryptanalysis, since it is difficult to decide whether an attack such as the Biclique attack on AES-128 [6] (which requires 2^{126} time) really "breaks" a scheme whose exhaustive search requires 2^{128} time. Some researchers suggested that this issue should be decided by the nature of the attack: If an attack on an n-bit scheme has an outer loop which tries 2^n different possibilities, but performs for each one of them an operation which is cheaper than a single encryption, then the attack should be called an "improvement of exhaustive search" rather than a "real attack," and the scheme is not said to be "broken" by it. However, this is a fragile definition since the same attack can be described in multiple ways, and it is not always clear whether it tries 2^n or fewer possibilities.

Fortunately, in cryptographic schemes such as EM which can be naturally defined for arbitrarily large key sizes n, we can avoid this fragility by analyzing the asymptotic complexity of the attack. As we show in this paper, our attacks are about $n/\log(n)$ times faster than exhaustive search. Since this ratio is unbounded when n increases, our attacks are asymptotically better than any standard or improved version of exhaustive search, and this is a robust statement since it ignores all the multiplicative constants, which are associated with a particular model of computation.

A delicate issue which we face in the analysis of our attacks is the complexity of sorting. As most papers in the field, our analysis assumes that this complexity is negligible compared to the complexity of cipher evaluations performed by our attacks. Moreover, since we are also interested in asymptotic analysis, we further assume that the asymptotic complexity gap between the cipher evaluations and sorting is sufficiently large, and thus, sorting complexity can also be neglected asymptotically. This assumption can be justified by the (implicit) requirement to make the public permutations P_i sufficiently strong as n grows. In particular, sorting a list of S n-bit elements requires $O(n \cdot S \cdot \log(S))$ bit operations, and our attacks sort lists of approximate size $S = \log(n)/n \cdot 2^n$, requiring $O(n \cdot \log(n)/n \cdot 2^n \cdot \log(\log(n)/n \cdot 2^n)) = O(n \log(n) \cdot 2^n)$ bit operations. Our attacks also make about $S = \log(n)/n \cdot 2^n$ cipher evaluations, and thus, if we assume that each evaluation requires (at least) $\theta(n^2)$ bit operations, then the complexity of the cipher evaluations is $\theta(n \log(n) \cdot 2^n)$ bit operations, which is also the asymptotic complexity of the full attacks (implying that the complexity of sorting can be neglected asymptotically). Note that as the complexity of exhaustive search is $\theta(2^n)$ cipher evaluations, it is equivalent in this case to $\theta(n^2 \cdot 2^n)$ bit operations, and thus, our attacks are faster by a factor of about $n/\log(n)$.

Some of the concrete schemes we consider in this paper (such as LED and AES^2) pose the following problem: They use the general iterated EM framework, but instantiate P with a fixed-key AES-like permutation, which is defined only for a few values of n, and thus, it is difficult to define their asymptotic security. We solve this problem in two ways. First, we observe that all our attacks are completely generic and do not exploit any particular properties of P besides its randomness. We can thus analyze the performance of our attacks, assuming that AES is replaced in these schemes by a random permutation over n-bit values and show that their asymptotic time complexity is smaller than that of exhaustive key search by a factor of $n/\log n$. In addition, we carefully analyze the exact complexity of our attacks for the particular values of n recommended for these schemes and show that they are between 7 and 64 times faster than exhaustive search, depending on the particular scheme we attack.

4. Attacks on Iterated Even–Mansour with Identical Subkeys

In this section, we consider iterated EM schemes with r permutations $P_1, P_2, \dots, P_r$, and $r+1$ identical subkeys, all equal K, as shown in Fig. 1 (note that if all the permutations are also the same, the scheme is extremely vulnerable to slide attacks [5]). Our goal is to use properties of one of the public permutations $P \in \{P_1, P_2, \dots, P_r\}$ in order to

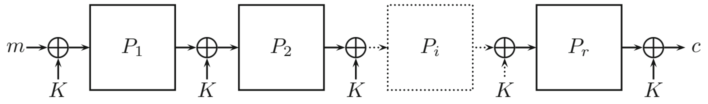

Fig. 1. Iterated EM with identical subkeys.

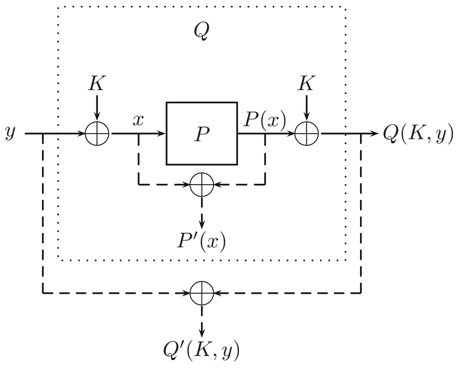

Fig. 2. The Functions P, P', Q, Q'.

deduce properties of the associated keyed permutation[5] $Q(K, x) = K \oplus P(x \oplus K)$ (used inside the EM construction), which hold for any value of K.

As Daemen pointed out in 1991 [9], for any value of K and in any invocation of $Q(K, x)$, the XOR of its input and output is equal to the XOR of the input and output of the internal P function in the same invocation, i.e., $x \oplus Q(K, x) = (x \oplus K) \oplus P(x \oplus K)$. Another interesting observation is that when K is unknown, we cannot determine $x \oplus K$, but the addition of K just renames the input vertices in the bipartite graph of $P'(x) = x \oplus P(x)$, and thus, it preserves the distribution of in-degrees of its output vertices. In particular, if some output values of P' are more likely than expected (i.e., appear more than the average), then we can predict the value $Q(K, x)$ with a higher probability than expected even when K is unknown. More specifically, any t-way collision on the value v in P', namely $x_1, x_2, \ldots, x_t$ such that $x_1 \oplus P(x_1) = x_2 \oplus P(x_2) = \cdots = x_t \oplus P(x_t) = v$ for some value of v, yields a t-way collision on the value v in the function $Q'(K, x) = x \oplus Q(K, x) = x \oplus K \oplus P(x \oplus K)$ (see Fig. 2). Assume that indeed we manage to find during a preprocessing phase a large t-way collision in the public $P'(x)$ on the output value v. Since it also yields a t-way collision on the value v in the keyed function $Q'(K, x)$, there are at least t values of x for which $Q'(K, x) = v$, and thus, $Q(K, x) = x \oplus v$. Consequently, we can guess $Q(K, x)$ with a probability, which is t times higher than the expected $1/2^n$ even when we know nothing about K.

This graph theoretic property is strongly related to the one used in [29], but we use it in a different way. Whereas we use properties of the public permutations (which can be observed during a preprocessing phase), [29] exploits properties of the given plaintext–ciphertext pairs: assume that $m_j \oplus c_j = v$ for multiple plaintext–ciphertext pairs (m_j, c_j). Then, for all of these pairs, they know that $(m_j \oplus K) \oplus (c_j \oplus K) = v$. Thus, the attack of [29] is based on the property that the XOR of the inputs to the first and last public permutations P_1 and P_r^{-1} attains the value v more than the expected number of times. In particular, in their attack, it is not clear how to compute such a v during a preprocessing phase, and they have to wait for the actual data in order to search for the best v in it. Our attacks, on the other hand, are based on the property that the XOR of the input and output of a single public permutation attains some value v more than the

[5] In general, given some public permutation P_i, we denote $Q_i(K, x) = K \oplus P_i(x \oplus K)$.

expected number of times, and thus, we can find the best v once and for all, before any data are given for a particular key.

In order to estimate the highest expected in-degree in the bipartite graph of $P'(x) = x \oplus P(x)$, we assume that for a random choice of the permutation P, the function P' behaves as a random function. This is not completely true, since there are some extremely expensive ways to distinguish between such cases (for example, the XOR of all the 2^n values of P' is zero, whereas the XOR of all the outputs of a truly random function is unlikely to be zero). However, it is easy to verify with appropriate simulations that the in-degree distributions of the two models behave almost identically, which is all we need in our attack.[6]

The main problem in applying this attack is that going over all the 2^n possible values of x in order to find the most popular v will make our attack slower than exhaustive search (since we do not allow free preprocessing). Fortunately, we can find vertices v', which are almost as popular by trying only a subset $X \subseteq \{0, 1\}^n$ of possible inputs. We denote this restricted function by $f_{|X}$ and note that it induces a subgraph in the bipartite graph associated with f, in which the left side of the graph contains only the vertices in X. Our goal now is to analyze the expected distribution of the in-degrees in random subgraphs of random functions.

Random functions have been extensively analyzed in the literature (e.g., see [16]). It is well known that the in-degree of an element in the range of $f_{|X}$ is distributed according to the Poisson distribution with an expectation λ, which is equal to the average in-degree (i.e., $\lambda = |X|/2^n$, which is the ratio between the sizes of the domain and range of $f_{|X}$). Given a parameter t, the probability that an arbitrary element v will have an in-degree of t is thus $(\lambda^t e^{-\lambda})/t!$ (see, for example, [32]). We have 2^n elements in the range, implying that we expect that about $(2^n \cdot \lambda^t e^{-\lambda})/t!$ vertices will have an in-degree of t. If we equate this number to 1 and ignore low-order terms, we can deduce that the largest expected in-degree t satisfies $t \cdot \log(t) = n$, and thus, t is approximately equal to $n/\log(n)$. The crucial point is that this highest in-degree grows in an unbounded way as n increases, and thus, any complexity of the form $O(2^n/t)$ behaves asymptotically as $o(2^n)$. If we reduce this maximal t to $t - i$ for a small i, we expect to find about $(t/\lambda)^i$ vertices which have this reduced in-degree. Since $t > 1$ and $\lambda < 1$, this number grows exponentially with i, and we can thus find a huge number of vertices, which have almost maximal in-degrees. To get a sense of the concrete values implied by this distribution for $n = 64$ (the block size of the LED block cipher), refer to Table 1.

The attacks in this paper are described in terms of several parameters, and it is usually possible to obtain various trade-offs between their time, data and memory complexities by tweaking the parameter values. However, since there is no simple formula which describes the exact trade-off curves, one needs to determine favorable trade-off points on the curves by plugging in a few values for the parameters and calculating the resultant complexities of the algorithms. This is demonstrated in our attacks, where we suggest concrete points on the curves which minimize the time complexity, but stress that there are other options as well.

[6] In fact, collisions in $P'(x)$ are slightly less likely to occur when P is a random function, since if $P(x) = P(y)$ (for $x \neq y$), then $P'(x) \neq P'(y)$, whereas if P is random permutation, then $x \neq y$ implies $P(x) \neq P(y)$, and the probability for $P'(x) = P'(y)$ is a bit higher. As a result, our analysis slightly underestimates the highest expected in-degree, and thus, the attacks that we describe are actually (negligibly) faster.

Table 1. Concrete in-degree values with $n = 64$.

Number of inputs	Vertex degree	Expected number of vertices
2^{64}	20	2 or 3
	19	55
	18	1060
2^{63}	17	1
	16	8
	15	260
2^{60}	10	4
	9	695
	8	$100130 \approx 2^{16.6}$

4.1. *Attacks on 2-Round Iterated Even–Mansour with Identical Subkeys*

We start by describing a very basic attack, $2Round1KeyBasic$. Let S and D be parameters.

Preprocessing:

PR1. Evaluate P_1' on an arbitrary subset of inputs X, such that $|X| = S$, and store the output values (without their associated input values) in a sorted list.

PR2. Traverse the sorted list and find the output v_1 which occurs the maximal number of times (in t_1 consecutive locations).

Online:

O1. Ask for the encryption of D arbitrary plaintexts.

O2. For each plaintext–ciphertext pair (m_i, c_i):

- (a) Assume that $Q_1(K, m_i) = m_i \oplus v_1 \triangleq z_i$ and calculate $P_2(z_i)$.
- (b) Test the suggestion for the key $K' = P_2(z_i) \oplus c_i$ by checking whether indeed $Q_1(K', m_i) = m_i \oplus v_1$. If the test fails, increment i and return to Step O2. Otherwise, return the suggested key.

The time complexity of the preprocessing phase is S evaluations of P_1, and its memory complexity is also S. Note that the output of the preprocessing phase is only the value v_1 and the corresponding number t_1, and we can discard the rest of the sorted list (as mentioned above, in our model we ignore the sorting time of the list). In addition, since we can execute the online phase in streaming mode by working on each given plaintext–ciphertext pair independently and discarding it afterward, its memory complexity is negligible. The expected time and data complexities of the online phase depend on the value of t_1: We know that there are at least t_1 values of x such that $Q_1(K, x) = x \oplus v_1$. According to the birthday paradox, after trying about $2^n/t_1$ arbitrary messages, we expect[7] that at least one m_i will satisfy $Q_1(K, m_i) = m_i \oplus v_1$ and suggest the correct value of K. Thus, the expected data complexity of the online algorithm is $2^n/t_1$, and in order to compute its time complexity, we need to sum $2^n/t_1$ evaluations of P_2 in Step

[7] The attack (and similar attacks described in this paper) succeeds to recover the key with probability of about $1 - e^{-1} \approx 0.63$. For a higher success probability, we need to increase the data and time complexities accordingly.

O2.(a), $2^n/t_1$ evaluations of Q_1 (or P_1) in Step O2.(b), and $2^n/t_1$ encryptions in order to generate the data.

4.1.1. *Optimizing the Basic Algorithm*

We now describe several useful optimizations of the $2Round1KeyBasic$ algorithm. The first optimization is to use the freedom to choose the subset X during the preprocessing phase in order to immediately filter out most of the wrong key suggestions that are now filtered only in Step O2.(b) of the online algorithm and thus avoid the Q_1 evaluations in these cases. The idea uses a technique that resembles (but is not the same as) splice-and-cut [2]: Assume that we choose the set X of size S as the subspace of values x in which the $n - \log(S)$ LSBs are zero (or any other constant). Then, the value of these $n - \log(S)$ LSBs in all the t_1 inputs x that satisfy $P'_{1|X}(x) = v_1$ is zero. Consequently, we know that for any plaintext m_i, if $m_i \oplus K$ is one of these t_1 inputs, then the $n - \log(S)$ LSBs of K are equal to those of m_i. Thus, before testing the suggested key in Step O2.(b), we can check whether its $n - \log(S)$ LSBs are equal to those of m_i and otherwise discard it without evaluating Q_1. We note that in this attack, the saving in time complexity due to this optimization is small; however, in Sect. 4.3, we show that a similar idea yields a more significant saving in our attacks on 3-round iterated EM. We alert the reader that even though the values in X are now chosen in a specific way, the attack remains a known-plaintext attack since there is no restriction on the choice of the m_i's.

The second optimization is to consider $\ell > 1$ outputs of P'_1 with a high in-degree instead of just one. This allows us to reduce the data complexity of the attack at the expense of using more memory and slightly more time during the online phase of the attack. Since the original online algorithm required only negligible memory, this trade-off seems favorable. Our optimized algorithm $2Round1KeyOpt$ is described below, using S, D and ℓ as parameters.

Preprocessing:

```
PR1. Evaluate $P'_1$ on a subset of $S$ inputs, $X$, such that the $n - \log(S)$ LSBs of each
     $x \in X$ are zero. Store the output values in a sorted list.
PR2. Traverse the sorted list and store the outputs $v_1, v_2, \ldots, v_\ell$ which have the highest
     in-degrees. Denote the in-degrees of the outputs $v_1, v_2, \ldots, v_\ell$ by $t_1, t_2, \ldots, t_\ell$,
     respectively.
```

Online:

```
O1. Ask for the encryption of $D$ arbitrary plaintexts.
O2. For each plaintext–ciphertext pair $(m_i, c_i)$:
    (a) For $j \in \{1, 2, \ldots, \ell\}$:
        (i)  Assume that $Q_1(K, m_i) = m_i \oplus v_j \triangleq z_{ij}$ and calculate $P_2(z_{ij})$.
        (ii) Let $K' = P_2(z_{ij}) \oplus c_i$. If the $n - \log(S)$ LSBs of $K'$ are different from
             those of $m_i$, discard it and return to Step O2.(a) (if $j = \ell$ return to Step
             O2). Otherwise, test $K'$ by checking whether $Q_1(K', m_i) = m_i \oplus v_j$. If
             the test succeeds, return $K'$, otherwise, if $j < \ell$ return to Step O2.(a) and
             if $j = \ell$ return to Step O2.
```

As in the $2Round1KeyBasic$, the time complexity of the preprocessing phase is S evaluations of P_1, and its memory complexity is also S. However, in $2Round1KeyOpt$, a bigger list of size ℓ is carried over to the online algorithm, and thus, its memory complexity is increased to ℓ. In order to calculate the time and data complexities, we denote by $\bar{t}$ the average value of $t_1, t_2, \ldots, t_\ell$, and thus, there are $\bar{t}\ell$ values of x for which $Q_1(K,x) = x \oplus v_j$ for $j \in \{1, 2, \ldots, \ell\}$. According to the birthday paradox, after trying about $2^n/(\bar{t}\ell)$ arbitrary messages, we expect that at least one m_i will satisfy $Q_1(K, m_i) = m_i \oplus v_j$ and suggest the correct value of K. Thus, the expected data complexity of the attack is $2^n/(\bar{t}\ell)$. Since we perform ℓ evaluations of P_2 per given message, the expected time complexity of the online algorithm is about $\ell \cdot D = 2^n/\bar{t}$ evaluations of P_2, $S/2^n \cdot 2^n/\bar{t} = S/\bar{t}$ evaluations of P_1 in Step O2.(a).ii, and $2^n/(\bar{t}\ell)$ time to generate the data.

4.1.2. *Concrete Parameters*

For $n = 64$, let $S = 2^{60}$, which implies $\lambda = 2^{60}/2^{64} = 2^{-4}$. As shown before, by using the formula $(2^n \cdot \lambda^t e^{-\lambda})/t! = 2^{64} \cdot (2^{-4t} e^{-1/16})/t!$ with $t = 10$, it is easy to check that in such an evaluated subgraph of a random function we expect to see at least $\ell = 4$ vertices with an in-degree of 10. With these parameters, the time complexity of the preprocessing phase is 2^{60} evaluations of P_1 (which is equivalent to 2^{59} evaluations of the 2-round scheme), and its memory complexity is 2^{60}. The memory complexity of the online algorithm is negligible, its data complexity is $2^{64}/(10 \cdot 4) = 2^{58.7}$ known plaintexts and its time complexity is $2^{64}/10$ evaluations of P_2 and $2^{60}/10$ evaluations of P_1, which is equivalent to about $2^{59.8}$ time units. Adding the $2^{58.7}$ time required to generate the data, we obtain a total time complexity of about $2^{60.4}$, which is about 12 times faster than exhaustive search.

We can significantly reduce the data complexity by considering all the vertices with an in-degree of at least 8, whose number ℓ is expected to exceed 2^{16}. This does not affect the time and memory complexities of the preprocessing phase. The memory complexity of the online algorithm is now 2^{16} (which is still quite small), its data complexity is $2^{64}/(8 \cdot 2^{16}) = 2^{45}$ known plaintexts and its time complexity is now $2^{64}/8$ evaluations of P_2 and $2^{60}/8$ evaluations of P_1, which is equivalent in total to about $2^{60.1}$ time units, or about 15 times faster than exhaustive search. In this case, the 2^{45} time required to generate the data has a negligible effect on the total time complexity. Note that since we use significantly less data compared to the previous attack (that used $2^{58.7}$ data), we actually gain in time complexity.

4.2. *A Low-Memory Attack on 2-Round Iterated Even–Mansour with Identical Subkeys*

In this section, we present another attack on 2-round iterated EM with identical subkeys. The attack is different from all other attacks described in this paper, since it does not exploit vertices with a high in-degree in random functions. In fact, the attack does not assume any property of the public permutations and thus is expected to work even for those which are highly non-random. The advantage of the attack compared to the attacks presented above is that it requires significantly less memory during preprocessing (in fact, the preprocessing computation of the attack is negligible). On the other hand, it requires more data than the data-efficient attack presented above, and in addition, it requires chosen plaintexts (rather than known plaintexts).

In order to simplify our notation, let v be an n-bit vector and X a set of n-bit vectors. We define the operation $v \oplus X \triangleq \{v \oplus x | x \in X\}$. We now describe a generalization of the filtering technique that we used in our optimized attack in Sect. 4.1: consider an arbitrary message m and an affine subspace X of dimension d_1, which is defined with a set of $n - d_1$ linear equations. If $m \oplus K \in X$, then $m \oplus K$ satisfies the $n - d_1$ linear equations which define X. Thus, if we know m, then we know the value of these linear equations on K. More generally, given an affine subspace V of dimension d_2, if there exists a message $m \in V$ such that $m \oplus K \in X$, then the intersection of the subspaces $K \oplus V$ and X is non-empty, and this imposes $n - d_1 - d_2$ linear equations on K. Our attack is based on this generalized filtering technique. Let S and D be parameters, such that $S + \log(S) < n$:
Preprocessing:

PR1. Evaluate P_1' on the subspace of inputs X of size S, which contains all the vectors whose $n - \log(S)$ LSBs are set to zero, and the $\log(S)$ MSBs attain all S possible values. Store the S outputs $v_1, v_2, \ldots, v_S$. Denote by V the linear vector space of dimension (at most) S spanned by $v_1, v_2, \ldots, v_S$.
PR2. Compute and store the (minimum of) $n - \log(S) - S$ linear equations imposed on K assuming that $(K \oplus V) \bigcap X \neq \emptyset$. Denote the computed matrix by A.

Online:

O1. Ask for the encryption of D structures (affine subspaces of dimension S) of plaintexts, defined by $m \oplus V$, where m is an arbitrary message.
O1. For each structure $m_i \oplus V$:
   (a) Assume that $((m_i \oplus K) \oplus V) \bigcap X \neq \emptyset$ and use m_i to compute the values of the linear equations of A on K.
   (b) For each $m_i^j \in m_i \oplus V$, compute $z_i^j \triangleq P_2(m_i^j)$ and use it to compute the value of the linear equations of A on z_i^j. Store these 2^S values in a sorted list L_1, next to z_i^j.
   (c) For each ciphertext c_i^j in the structure, compute the value of the linear equations of A on $c_i^j \oplus K$ (their value on K is known from Step O2.(a)). Search for matches of this value in L_1.
   (d) For each match associated with c_i^j and z_i^ℓ (for some value of ℓ), obtain a suggestion for the key $K' = c_i^j \oplus z_i^\ell$ and test it using one plaintext–ciphertext pair. If the test succeeds, return the key.

In order to understand the main idea of the attack, assume that indeed $((m_i \oplus K) \oplus V) \bigcap X \neq \emptyset$ for some m_i (as assumed in Step O2.(a)), i.e., $(m_i \oplus K \oplus v) \in X$ for some $v \in V$. Then, we know that $P_1'(m_i \oplus K \oplus v) \in V$, and thus $Q_1'(K, m_i \oplus v) = P_1'(m_i \oplus K \oplus v) \in V$, i.e., $Q_1(K, m_i \oplus v) = m_i \oplus v \oplus v'$ for some $v' \in V$. Since V is a linear subspace, it implies that $Q_1(K, m_i \oplus v) \in m_i \oplus V$. As we compute P_2 for all $m_i^j \in m_i \oplus V$ in Step O2.(b), we will get a match in step 2.(c) with the ciphertext of our plaintext $m_i \oplus v$, and this match will suggest the correct key.

The time complexity of the preprocessing phase is S evaluations of P_1 in addition to some linear algebra of complexity of about n^3 bit operations in Step PR2 (which can be performed by Gaussian elimination). As calculated shortly, these computations are negligible compared to the computations performed in the online phase of the attack. The memory complexity of the preprocessing phase is also proportional to S, with the addition if about n^2 bits required in order to store A. Again, this storage is negligible compared to the storage required by the online algorithm.

The memory complexity of the online algorithm is about 2^S, required to store L_1 (note that we can work on the ciphertexts of each structure in streaming mode, and thus, we do not need to store them in memory). The attack finds the correct key once $((m_i \oplus K) \oplus V) \bigcap X \neq \emptyset$ for some m_i. Since $|X| = S$, we need about $2^n/S$ data for this to occur with high probability, i.e., we require $D \approx 2^n/(S \cdot 2^S)$ structures of 2^S plaintexts. For each such structure, we perform 2^S evaluations of P_2. In order to calculate the expected number of trial encryptions performed in Step O2.(d) for each structure, we notice that we match two lists of size 2^S (although only L_1 is explicitly stored in memory). The number of matched bits is (at least) $n - \log(S) - S$, and thus, the expected number of trial encryptions per structure is $2^{2S-n+\log(S)+S} = 2^{3S+\log(S)-n}$.

If we take $S = n/4$, then the time complexity of the trial encryptions is negligible compared to the complexity of evaluations of P_2, which is equivalent in total to about $0.5 \cdot 2^n/S$ encryptions. Adding the additional $2^n/S$ encryptions required to prepare the data, the total time complexity of the attack is equivalent to about $2^{n+0.6}/S$ encryptions.

We note that there are several possible extensions to this algorithm, which include a more complicated preprocessing phase. In particular, we can combine this attack with the techniques of Sect. 4.1 by looking for subsets X whose elements have a large in-degree in the associated graph of P'_1. However, we do not describe these highly complicated extensions in this paper.

4.2.1. *Concrete Parameters*

For $n = 64$, let $S = 16$. Then, the memory complexity of the attack is 2^{16}, its expected data complexity is 2^{60} chosen plaintexts and its expected time complexity is about $2^{60.6}$.

4.3. *Attacks on 3-Round Iterated Even–Mansour with Identical Subkeys*

In the attacks on 2-round iterated EM with identical subkeys, we use properties of P_1 in order to guess a value of $Q_1(K, x)$ with a higher probability than expected. We then apply to this guess the public permutation P_2, which immediately gives us a suggestion for the key by XORing the obtained value with the ciphertext. In order to attack 3-round iterated EM with identical subkeys, we start with the same idea. However, after the evaluation of P_2, we cannot immediately get a suggestion for the key, as we still have to apply the complex operation of XOR'ing the unknown key, applying P_3, and XOR'ing the unknown key again, before we can compare the result to the ciphertext. Nevertheless, we notice that given the value at the output of P_2, we reduce the key recovery problem to attacking a single-round EM scheme with one key, to which we can apply the simple attack of [14]. Thus, we run an additional preprocessing step which evaluates and stores in a sorted list the values of $P'_3(x) = x \oplus P_3(x)$ for various inputs x. The sorted list is

used in the online algorithm in order to obtain suggestions for the key, as described in the basic algorithm $3Round1KeyBasic$ below, which uses S_1, S_3 and D as parameters.
Preprocessing:

PR1. Evaluate P_1' on an arbitrary subset of inputs X_1 such that $|X_1| = S_1$ and store the output values in a sorted list.

PR2. Traverse the sorted list and find an output v_1 with a maximal in-degree, denoted by t_1.

PR3. Evaluate P_3' on an arbitrary subset of inputs X_3, such that $|X_3| = S_3$, and store the output values $P_3'(x)$ in a sorted list L_3 next to the corresponding value of $P_3(x)$.

Online:

O1. Ask for the encryption of D arbitrary plaintexts.

O2. For each plaintext–ciphertext pair (m_i, c_i):

(a) Assume that $Q_1(K, m_i) = m_i \oplus v_1 \triangleq z_i$ and calculate $P_2(z_i)$.

(b) Look for the value of $P_2(z_i) \oplus c_i$ in L_3. If there is no match, return to Step O2 and increment i.

(c) For each match of $P_2(z_i) \oplus c_i$, obtain the value of $P_3(x)$ (for which $P_2(z_i) \oplus c_i = P_3'(x) = x \oplus P_3(x)$) and test the key suggestion $K' = P_3(x) \oplus c_i$ by checking whether $Q_1(K', m_i) = m_i \oplus v_1$. If the test fails, continue with the next match (if none remain, return to Step O2). Otherwise, return the key.

The time complexity of the preprocessing phase is S_1 evaluations of P_1 and S_3 evaluations of P_3, and its memory complexity is $max(S_1, S_3)$. Note that we do not need to store any of the values generated in the first step of the preprocessing after Step PR2 terminates. The memory complexity of the online algorithm is S_3. In order to calculate the expected time and data complexities of the online algorithm, we notice that after we process D pairs (m_i, c_i), we expect that at least $(t_1 \cdot D)/2^n$ of them satisfy $Q_1(K, m_i) = m_i \oplus v_1$, and consequently at least $(t_1 \cdot D \cdot S_3)/2^{2n}$ pairs will be matched and suggest the correct value for the key in Step O2.(c). Thus, in order to obtain a correct suggestion for the key, we require $(t_1 \cdot D \cdot S_3)/2^{2n} = 1$, implying that the data complexity of the attack is $D = 2^{2n}/(t_1 \cdot S_3)$. We expect a match in Step O2.(c) for a fraction of $S_3/2^n$ of the (m_i, c_i) pairs. Thus, we estimate the time complexity of the online algorithm as $D = 2^{2n}/(t_1 \cdot S_3)$ evaluations of P_2, $S_3/2^n \cdot 2^{2n}/(t_1 \cdot S_3) = 2^n/t_1$ evaluations of P_1, and $2^{2n}/(t_1 \cdot S_3)$ time required to generate the data.

4.3.1. *Optimizing the Basic Algorithm*

Similarly to our $2Round1KeyOpt$ attack, we would like to use the freedom to choose the subset X_1 during preprocessing in order to reduce the time complexity of the attack. However, in this attack we will use this freedom in a different way: We "synchronize" the sets X_1 and X_3 such that we can instantly rule out most pairs (m_i, c_i) (just by comparing bits of m_i and c_i) that do not simultaneously satisfy both $Q_1(K, m_i) = m_i \oplus v_1$ and $P_3^{-1}(c_i \oplus K) \in X_3$. Thus, we can discard most pairs (m_i, c_i), which will suggest a wrong key (or suggest no key at all) with a negligible computation.

We now assume that $|X_1| = |X_3| = S$. Similarly to the $2Round1KeyOpt$ algorithm, we choose X_1 as a subspace of values x in which the $n - \log(S)$ LSBs are zero (or

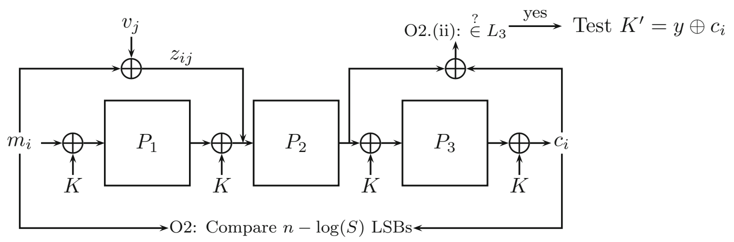

Fig. 3. The online algorithm of $3Round1KeyOpt$.

any other constant). This implies that for any plaintext m_i, if $m_i \oplus K$ is one of the t_1 inputs that satisfy $P'_{1|X_1}(x) = v_1$, then the $n - \log(S)$ LSBs of K are equal to those of m_i. As for $x \in X_3$, we store the values of $P'_3(x) = x \oplus P_3(x)$ and set the additional condition that the $n - \log(S)$ LSBs of $P_3(x)$ are zero (or any other constant). In fact, during preprocessing, we do not evaluate $P_3(x)$ on $x \in X_3$, but rather evaluate $P_3^{-1}(y)$ for each $y \in Y_3$, where Y_3 contains all n-bit vectors whose $n - \log(S)$ LSBs are zero. Thus, we know that if $c_i \oplus K \in Y_3$, then the $n - \log(S)$ LSBs of K are equal to those of c_i. Combining the conditions on m_i and c_i, we know that a pair (m_i, c_i) will suggest a correct key in our algorithm only if the $n - \log(S)$ LSBs of m_i and c_i are equal.

Similarly to the $2Round1KeyOpt$ attack, the second optimization is to consider $\ell > 1$ outputs of P'_1 with a high in-degree (instead of just one), which allows us to reduce the data complexity of the attack. Our optimized algorithm $3Round1KeyOpt$ is described below, and Fig. 3 illustrates its online part. Let S, D and ℓ be parameters.

Preprocessing:

PR1. Evaluate P'_1 on a subset of S inputs, X, such that the $n - \log(S)$ LSBs of each $x \in X$ are zero. Store the output values in a sorted list.

PR2. Traverse the sorted list and store the outputs $v_1, v_2, \ldots, v_\ell$ with the highest in-degrees. Denote the in-degrees of outputs $v_1, v_2, \ldots, v_\ell$ by $t_1, t_2, \ldots, t_\ell$, respectively.

PR3. Let Y_3 be the subspace of the $|S|$ n-bit vectors in which the $n - \log(S)$ LSBs are zero. For each $y \in Y_3$, store $P_3^{-1}(y) \oplus y = P'_3(P_3^{-1}(y))$ in a sorted list L_3 next to y.

Online:

O1. Ask for the encryption of D arbitrary plaintexts.

O2. For each plaintext–ciphertext pair (m_i, c_i), if the $n - \log(S)$ LSBs of m_i and c_i differ, discard it. Otherwise:

(a) For $j \in \{1, 2, \ldots, \ell\}$:

(i) Assume that $Q_1(K, m_i) = m_i \oplus v_j \triangleq z_{ij}$ and calculate $P_2(z_{ij})$.

(ii) Look for the value of $P_2(z_{ij}) \oplus c_i$ in L_3. If there is no match: if $j < \ell$ return to Step O2.(a), otherwise ($j = \ell$) return to Step O2.

(iii) For each match of $P_2(z_{ij}) \oplus c_i$, obtain the value of y (such that $P_2(z_i j) \oplus c_i = P_3^{-1}(y) \oplus y = P'_3(P_3^{-1}(y))$), and test the key suggestion $K' = y \oplus c_i$ by checking whether $Q_1(K, m_i) = m_i \oplus v_j$. If the test succeeds, return

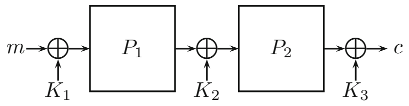

Fig. 4. A 2-round iterated EM with independent subkeys.

K', otherwise, if $j < \ell$ return to Step O2.(a), and if $j = \ell$ return to Step O2.

The time complexity of the preprocessing phase is S evaluations of P_1 and P_3^{-1}, and its memory complexity is $S + \ell$. The memory complexity of the online algorithm is also $S + \ell$, and we note that we will generally select a value of ℓ, which is negligible compared to S. Denote by $\bar{t}$ the average value of $t_1, t_2, \ldots, t_\ell$, and thus, there are $\bar{t}\ell$ values of x for which $Q_1(K, x) = x \oplus v_j$ for $j \in \{1, 2, \ldots, \ell\}$. Consequently, in order to obtain a correct suggestion for the key, we require that $(\bar{t}\ell \cdot D \cdot S)/2^{2n} = 1$, implying that the data complexity of the attack is $D = 2^{2n}/(\bar{t}\ell \cdot S)$. We process a pair (m_i, c_i) (i.e., we do not discard it in step 2) with probability $S/2^n$, and for each such pair we perform ℓ evaluations of P_2 and for a $S/2^n$ fraction of those we also evaluate Q_1 (or P_1). The expected time complexity of the online algorithm is thus $\ell \cdot S/2^n \cdot D = 2^n/\bar{t}$ evaluations of P_2, $S/\bar{t}$ evaluations of P_1, and $2^{2n}/(\bar{t}\ell \cdot S)$ time required to generate the data.

Thus, the attack has about the same time complexity as the $2Round1KeyOpt$ attack, and for $\ell = 1$, it is more efficient than the $3Round1KeyBasic$ attack by a factor of about $2^n/S$.

4.3.2. *Concrete Parameters*

For $n = 64$, let $S = 2^{60}$, i.e., $\lambda = 2^{60}/2^{64} = 2^{-4}$. Again, we use the formula $(2^n \cdot \lambda^t e^{-\lambda})/t! = (2^{64} \cdot 2^{-4t} e^{-1/16})/t!$ with $t = 8$, such that we expect at least $\ell = 2^{16}$ vertices with an in-degree of 8. With these parameters, the time complexity of the preprocessing phase is 2^{60} evaluations of P_1 (equivalent to about $2^{58.5}$ evaluations of the 3-round scheme), and its memory complexity is 2^{60}. The memory complexity of the online algorithm is 2^{60}, its expected data complexity is $2^{128}/(8 \cdot 2^{16} \cdot 2^{60}) = 2^{49}$ known plaintexts and its expected time complexity is $2^{64}/8$ evaluations of P_2 and $2^{60}/8$ evaluations of P_1, whose sum is equivalent to about $2^{59.6}$ time units (the time required to generate the data is negligible). Note that it is possible to reduce the data complexity further at the expense of increasing the time complexity by considering vertices of a lower in-degree.[8]

5. Attacks on 2-Round Iterated Even–Mansour with Independent Subkeys

Unlike iterated EM schemes with identical subkeys, where the time complexity of exhaustive search is 2^n, 2-round iterated EM with independent subkeys has $3n$ key bits (see Fig. 4), and thus, straightforward exhaustive search has a time complexity of 2^{3n}.

[8] For example, we expect more than 2^{23} vertices with an in-degree of at least 7, and thus, if we use only $2^{128}/(8 \cdot 2^{23} \cdot 2^{60}) = 2^{42}$ known plaintexts for the attack, the time complexity of the online algorithm slightly increases from $2^{59.6}$ to about $2^{59.8}$.

It is very easy to improve the time complexity of the attack to 2^{2n} by noticing that $K_3 = P_2(P_1(m \oplus K_1) \oplus K_2) \oplus c$, and thus guessing K_1 and K_2 suffices in order to compute K_3. A further simple improvement is to use a meet-in-the-middle attack in order to reduce the time complexity to about 2^n, at the expense of using about 2^n memory. Such a MITM attack on an iterated EM scheme with an arbitrary number of rounds is given in [7], and we describe it here for $r = 2$ (under the name $2Round3KeyMITM$) for the sake of completeness.

1. Obtain 3 known plaintext–ciphertext pairs — (m_1, c_1), (m_2, c_2), (m_3, c_3).
2. For each value of K_1, compute $\Delta_1 = P_1(m_1 \oplus K_1) \oplus P_1(m_2 \oplus K_1)$ and store (Δ_1, K_1) in a list sorted according to Δ_1.
3. For each value of K_3:
   (a) Compute $\Delta_3 = P_2^{-1}(c_1 \oplus K_3) \oplus P_2^{-1}(c_2 \oplus K_3)$ and search for Δ_3 in the sorted list.
   (b) For each match, test the key K_1, $K_2 = P_1(m_1 \oplus K_1) \oplus P_2^{-1}(c_1 \oplus K_3)$, K_3 using (m_3, c_3), and if the tests succeeds return the key.

The memory complexity of the algorithm is 2^n, required for storing the list in Step 2. The time complexity of Step 2 is $2 \cdot 2^n$ evaluations of P_1. We expect a single match in Step 3.(a) for a value of K_3, and hence, the expected time complexity of Step 3 is $2 \cdot 2^n$ evaluations of P_1 and 2^n evaluations of the full cipher. The expected time complexity of the attack is thus $3 \cdot 2^n \approx 2^{n+1.6}$. We note that it is possible to trade some memory at the expense of time. However, in this paper we are mainly interested in the attack with the lowest time complexity.

In order to devise a more efficient attack, we use a property of the permutation P_i, which is shared by the keyed permutation $Q_i(K_i, K_{i+1}, x) = P_i(x \oplus K_i) \oplus K_{i+1}$ for any value of K_i and K_{i+1}: These permutations have the same difference distribution table. In order to demonstrate this, consider an entry with the value of t in the difference distribution table of P_i and denote its input and output differences by Δ_1 and Δ_2, respectively. Let us denote the t corresponding input–output pairs[9] by $((x_1, y_1), (x_1 \oplus \Delta_1, y_1 \oplus \Delta_2)), \ldots, ((x_t, y_t), (x_t \oplus \Delta_1, y_t \oplus \Delta_2))$. Then, the t input–output pairs $((x_1 \oplus K_1, y_1 \oplus K_2), (x_1 \oplus K_1 \oplus \Delta_1, y_1 \oplus K_2 \oplus \Delta_2)), \ldots, ((x_t \oplus K_1, y_t \oplus K_2), (x_t \oplus K_1 \oplus \Delta_1, y_t \oplus K_2 \oplus \Delta_2))$ correspond to the same entry in the difference distribution table of Q_i (i.e., the entry with input and output differences Δ_1 and Δ_2, respectively).

Using the property above, if we find an entry $[\Delta_1, \Delta_2]$ in the difference distribution table of P_i with a large value, then we can use a similar attack to the one given in [27] on 2-round iterated EM,[10] in order to break the scheme. However, our main observation is that we can find such an entry by preprocessing the public function P_i, which does not need to admit any special property in order to attack the scheme. Thus, our attack adds a preprocessing algorithm to the online algorithm of the attack of [27] (which assumes that we have an entry in the difference distribution table of P_i with a large value). In addition (as we will see later), in the case of independent keys, the basic attack of [27] is not

[9] In this paper, we consider unordered pairs, i.e., $((x, y), (u, v))$ and $((u, v), (x, y))$ are considered the same pair.

[10] Although the attack of [27] was previously applied to 2-round iterated EM with one key, it can be adapted to work for the case of independent keys.

better than exhaustive search, and we will need to add another non-trivial component to this attack. The details of our unoptimized attack $2Round3KeyBasic$ are given below, where S_1, S_2, D are parameters:

Preprocessing:

PR1. Choose an arbitrary input difference $\Delta_1 \neq 0$ and evaluate P_1 on S_1 arbitrary input pairs with input difference Δ_1. For each pair $(x, P_1(x)), (x \oplus \Delta_1, P_1(x \oplus \Delta_1))$, store the output difference $P_1(x) \oplus P_1(x \oplus \Delta_1)$ in a sorted list, next to x.

PR2. Traverse the sorted list and find the most common output difference Δ_2 (if there are several options for Δ_2, choose one arbitrarily). Keep only the entries of the list which correspond to pairs with the output difference of Δ_2 (assume that we have t such pairs). For each such entry, recalculate and store $(x, P_1(x)), (x \oplus \Delta_1, P_1(x \oplus \Delta_1))$ (note that we did not store $P_1(x)$ and $P_1(x \oplus \Delta_1)$ in the previous step in order to save memory).

PR3. Evaluate P_2 on S_2 arbitrary input pairs with input difference Δ_2. For each pair $(y, P_2(y)), (y \oplus \Delta_2, P_2(y \oplus \Delta_2))$, store the output difference $P_2(y) \oplus P_2(y \oplus \Delta_2)$ in a sorted list L_2, next to y.

Online:

O1. Ask for the encryption of D arbitrary input pairs with difference Δ_1.

O2. For each pair of plaintext–ciphertext pairs $((m_i^1, c_i^1), (m_i^2 = m_i^1 \oplus \Delta_1, c_i^2))$:

(a) Search for the output difference $c_i^1 \oplus c_i^2$ in L_2, (if there is no match, discard the pair and return to Step O2).

(b) For each match $(y, P_2(y)), (y \oplus \Delta_2, P_2(y \oplus \Delta_2))$, we have 2 candidates for K_3: $P_2(y) \oplus c_i^1$ and $P_2(y) \oplus c_i^2$. We also have $2t$ candidates for K_1: the candidates $x \oplus m_i^1$ and $x \oplus m_i^2$ for each of the t values of x. As each pair of values for K_1 and K_3 suggests a value for K_2, we have $4t$ suggestions of the full key to test using another plaintext–ciphertext pair.

Similarly to our analysis of random functions, assuming that P_1 is a random permutation, then each entry in its difference distribution table is distributed according to the Poisson distribution [30].[11] This will allow us to easily determine the expected value of t and use it in order to analyze the expected complexity of our algorithm.

The memory complexity of the preprocessing phase is $max(S_1, S_2)$, and its time complexity is $2 \cdot S_1$ evaluations of P_1 and $2 \cdot S_2$ evaluations of P_2, or $S_1 + S_2$ evaluations of the full scheme. The memory complexity of the online algorithm is S_2. The data complexity is D plaintext–ciphertext pairs, i.e., $2D$ chosen plaintexts. Using the birthday paradox, out of the D plaintext–ciphertext pairs evaluated in the online phase, at least $(D \cdot t)/2^{n-1}$ are expected to have a difference of Δ_2 after P_1 (note that we have 2^{n-1} unordered pairs with a given difference). Using the same argument, we expect that $(D \cdot t \cdot S_2)/2^{2(n-1)}$ of them will match the pairs evaluated for P_2 during preprocessing. Thus, we require that $(D \cdot t \cdot S_2)/2^{2(n-1)} = 1$, or $D = 2^{2(n-1)}/(t \cdot S_2)$ in order to find the key with high probability. Without going into the details of the time complexity analysis, note that we are using only two plaintext–ciphertext pairs to filter the key suggestions,

[11] However, we note that since we consider unordered pairs, then we have only 2^{n-1} possible pairs of a given difference, and each pair can attain (almost) all 2^n output differences.

tested in Step O2.(b). As we have $3n$ bits of key and $2n$ bits of filtering, we need to test at least 2^n keys in Step O2.(b), and thus, the attack is not faster than the simple MITM attack on this scheme.

5.1. *A Time-Optimized Attack on 2-Round Iterated Even–Mansour with Independent Subkeys*

In order to improve the attack, we need to add more filtering conditions, and thus, we actually work on triplets, as described in the improved algorithm $2Round3KeyOpt$:
Preprocessing:

PR1. Choose an arbitrary input difference $\Delta_1 \neq 0$ and evaluate P_1 on S_1 arbitrary input pairs with input difference Δ_1. For each pair $(x, P_1(x)), (x \oplus \Delta_1, P_1(x \oplus \Delta_1))$, store the output difference $P_1(x) \oplus P_1(x \oplus \Delta_1)$ in a sorted list, next to x.

PR2. Traverse the sorted list and find the most common output difference Δ_2 (if there are several options for Δ_2, choose one arbitrarily). Keep only the entries of the list which correspond to pairs with the output difference of Δ_2 (assume that we have t such pairs). For each such entry, recalculate and store the full pair $(x, P_1(x)), (x \oplus \Delta_1, P_1(x \oplus \Delta_1))$ in a list L_1.

PR3. Choose another nonzero input difference Δ_1'. For each value x stored in L_1, evaluate P_1 an additional time to obtain the pair $(x \oplus \Delta_1', P_1(x \oplus \Delta_1'))$. Store the (total of) additional t output differences $P_1(x) \oplus P_1(x \oplus \Delta_1')$ in a separate sorted list of differences, L_1'.

PR4. Evaluate P_2 on S_2 arbitrary input pairs with input difference Δ_2. For each pair $(y, P_2(y)), (y \oplus \Delta_2, P_2(y \oplus \Delta_2))$, store the output difference $P_2(y) \oplus P_2(y \oplus \Delta_2)$ in a sorted list L_2, next to y.

Online:

O1. Ask for the encryption of D arbitrary input triplets of the form $m_i^1, m_i^2 = m_i^1 \oplus \Delta_1$ and $m_i^3 = m_i^1 \oplus \Delta_1'$ (for D arbitrary values of m_i^1).

O2. For each pair of plaintext–ciphertext pairs $((m_i^1, c_i^1), (m_i^2, c_i^2))$:

(a) Search for the output difference $c_i^1 \oplus c_i^2$ in the list L_2 (if there is no match, discard the pair and return to Step O2).

(b) For each match $(y, P_2(y)), (y \oplus \Delta_2, P_2(y \oplus \Delta_2))$, compute the 2 candidates for K_3: $K_3' = P_2(y) \oplus c_i^1$ and $K_3'' = P_2(y) \oplus c_i^2$.

(c) Compute $y' = P_2^{-1}(c_i^3 \oplus K_3')$ and $y'' = P_2^{-1}(c_i^3 \oplus K_3'')$.

(d) Search L_1' for the four possibilities of the third difference obtained at this stage: $y' \oplus y$, $y' \oplus \Delta_2 \oplus y$, $y'' \oplus y$, $y'' \oplus \Delta_2 \oplus y$ (if there is no match, discard the pair and return to Step O2).

(e) Test the $4t$ suggestions of the full key using (m_i^3, c_i^3). If the test succeeds, return the key.

The time and memory complexities of the preprocessing phase are similar to those of the $2Round3KeyBasic$ attack (the additional t evaluations of P_1 and t units of storage are negligible). Using the calculation done for $2Round3KeyBasic$, the online algorithm requires $D = 2^{2(n-1)}/(t \cdot S_2)$ plaintext–ciphertext triplets. For each processed triplet, we expect to find a match in L_2 with probability $S_2/2^n$. For each such matched triplet, we need to compute $P_2(y)$ (in order to compute K'_3 and K''_3) and evaluate P_3^{-1} twice in order to compute y' and y''. Once we do so, the probability of a match in L'_1 in Step O2.(d) is proportional to $t/2^n$. This is a negligible probability, and thus, we can neglect the complexity of the trial encryptions in Step O2.(e). Thus, the online time complexity (without counting the data) is about $3 \cdot D \cdot S_2/2^n = 0.75 \cdot 2^n/t$ evaluations of P_2, or $0.375 \cdot 2^n/t$ evaluations of the full scheme.

The data complexity of the attack is D triplets, or $3D$ chosen plaintexts. However, we can easily reduce it to $2D$ by requesting encryptions of structures containing the messages $m_i^1, m_i^1 \oplus \Delta_1, m_i^1 \oplus \Delta'_1$ and $m_i^1 \oplus \Delta_1 \oplus \Delta'_1$. Each such structure of 4 plaintexts contains two triplets which we can exploit, implying that the data complexity of the attack is only $2D$. If we add the time to generate the data to the time complexity, we get that the total time complexity of the online attack is about $2D + 0.375 \cdot 2^n/t$ evaluations of the full scheme.

An application of the algorithm to the block cipher AES^2 is given in Sect. 7.

6. Attacks on Other Variants of the Iterated Even–Mansour Scheme

In this section we study three other variants of the iterated EM construction that may be relevant for concrete block cipher instances, as described in the Introduction. In Sect. 6.1, we consider iterated EM with identical subkeys where the permutations are chosen at random among the family of *involutions*. We show that unlike the case of single-key EM (where choosing the permutation to be an involution does not reduce the security), in the iterated EM case, this choice drastically reduces the security of the scheme. In Sect. 6.2, we consider iterated EM with two n-bit keys K_1, K_2 that are used alternately as subkeys (as in the block cipher LED-128) and leverage our attack on 3-round EM with identical subkeys to an attack on 8 rounds of this scheme. Finally, in Sect. 6.3 we show that all of our attacks on iterated EM with identical subkeys can be adapted (with about the same complexity) to schemes in which the subkeys are derived from the key by any linear function.

6.1. *Optimized Attacks on Iterated Even–Mansour Schemes with Involutional Permutations*

A permutation P is called an *involution*, if it is equal to its inverse, i.e., $P = P^{-1}$. In [14], the authors studied one-round EM schemes in which the internal permutation P is chosen uniformly at random among the set of all the possible involutions on n-bit strings. They showed a significant difference between the original (two-key) EM and the single-key variant of EM. For single-key EM, they proved that using a random involution does not reduce the security of the scheme with respect to key recovery attacks. On the contrary, for two-key EM they presented an attack with $D = 2^{n/2}$ and no evaluations of P, showing that the security of involutional one-round EM schemes is greatly reduced from the original $DT = 2^n$ curve, obtained for schemes built using random permutations.

In this section, we show that while iterated EM with identical subkeys is a natural generalization of *single-key* EM, the effect of using even a single random involution as one of its permutations is a considerable reduction in security. This is somewhat similar to the case of *two-key* EM (considered in [14]); however, the effect of involutions on the security of iterated EM schemes with identical subkeys is even more extreme: While the total time complexity of the best attack on involutional two-key EM remains $D = 2^{n/2}$ (as in standard two-key EM), the total time complexity of the best attacks on iterated EM schemes with identical subkeys drops significantly when implementing them with involutions. To show this, we optimize our attacks on 2-round and 3-round iterated EM schemes with identical subkeys, to cases where some of their internal permutations are chosen as random involutions. In particular, we show that even if we choose only one of the internal permutations in such schemes as an involution, both the 2-round and 3-round EM schemes become significantly weaker.

Our attacks are based on a well-known property of an n-bit random involution, namely that it has an expected number of about $2^{n/2}$ fixed points [17] for which $x = P(x)$. As this implies that $x \oplus P(x) = 0$, the vertex $v = 0$ in the graph of $P'(x) = x \oplus P(x)$ is expected to have an in-degree of about $2^{n/2}$. This is significantly larger than the expected $O(n)$ maximal degree of any vertex of $P'(x)$, when P is chosen as a random permutation and results in attacks which are significantly more efficient.

We note that although a random involution (chosen uniformly at random among the set of all the possible involutions on n-bit strings) has an expected number of about $2^{n/2}$ fixed points, there are many involutions with significantly fewer fixed points, or with no fixed points at all. For example, the involution $F(x) = x \oplus c$ has no fixed points for $c \neq 0$. However, when considering a relatively complex involution (obtained, for example, by computing several rounds of a keyless Feistel structure), a natural assumption is that it behaves similarly to a random involution, and thus, it has about $2^{n/2}$ fixed points.

6.1.1. *An Attack on 2-Round Iterated EM with Identical Subkeys where P_1 is an Involution*

We show how to optimize the $2Round1KeyBasic$ attack of Sect. 4.1, in the case that P_1 is an involution: Given that we know in advance that the in-degree of $v = 0$ in the graph of P_1' is expected to be about $2^{n/2}$, we can completely remove the preprocessing phase. By exploiting the special vertex $v = 0$, the time and data complexities of $2Round1KeyBasic$ are both $2^n/t_1 \approx 2^{n/2}$ (as $t_1 \approx 2^{n/2}$), and it requires only a negligible amount of memory. We note that in the case when P_1 is an involution, the optimization techniques of $2Round1KeyOpt$ seem useless, as the in-degrees of all vertices $v \neq 0$ are expected to be much smaller than $2^{n/2}$, and exploiting them in our attack is inefficient.

Finally, we note that the optimized attack in case P_2 is an involution is simply obtained by reversing the roles of encryption and decryption and applying the same optimization as above.

6.1.2. *An Attack on 3-Round Iterated EM with Identical Subkeys where P_1 is an Involution*

We show how to optimize the $3Round1KeyBasic$ attack of Sect. 4.3, in the case that P_1 is an involution. Similarly to the case of the $2Round1KeyBasic$ attack, we can remove the first two steps of the preprocessing phase which evaluate P_1. However, as we still need to preprocess P_3, the time and memory complexities of the preprocessing phase are now S_3. Recall that the time and data complexities of the online phase of

$3Round1KeyBasic$ are both about $2^{2n}/(t_1 \cdot S_3)$, and as $t_1 = 2^{n/2}$, they are equal to $2^{1.5n}/S_3$. Thus, the total time complexity of the attack is $max\{2^{1.5n}/S_3, S_3\}$, giving a time–data trade-off of $TD = 2^{1.5n}$ for any $2^{0.5n} \leq D \leq 2^{0.75n}$. In particular, for $D = 2^{0.75n}$, we obtain $T = 2^{0.75n}$, using memory of $M = 2^{0.75n}$. We note that when $D < 2^{0.75n}$, we can obtain $M = D$ by first storing the data in memory and delaying the processing of P_3, which can now be done "on-the-fly."

We additionally note that as in the case of the 2-round attack, the optimization techniques of $3Round1KeyOpt$ seem useless in the scenario considered here. Moreover, similarly to the 2-round attack, the optimized attack in case P_3 is an involution is simply obtained by reversing the roles of encryption and decryption and applying the same optimization as above. When P_2 is an involution, the attack can still be applied with a trade-off of $TD = 2^{1.5n}$. However, it requires more advanced techniques of enumerating the internal values of the fixed point around P_2, which are similar to the techniques recently published in [12].

Finally, we point out a very interesting property of iterated EM schemes with identical subkeys: As described in Sect. 4.1, our best attack on the single-key 2-round EM scheme, built using random permutations P_1 and P_2, has a time complexity of about $2^n/n$. Surprisingly, the result obtained above shows that not only does adding an arbitrary involutional round (which is completely unrelated to the original permutations) to this scheme not increase its security, but it rather significantly reduces it to $T = 2^{0.75n}$ (while also significantly reducing the memory complexity of the attack). To the best of our knowledge, this is the first non-trivial backdoor that can be "hidden" at plain sight concerning symmetric-key cryptosystems.

6.2. *Extending Our Attacks to Iterated Even–Mansour Schemes with Alternating Subkeys*

As was pointed out in [27,29] in the specific case of LED-128, it is easy to reduce an $2r+2$-round variant of EM with alternating subkeys to an r-round variant with identical subkeys. This is done by guessing K_1 and combining consecutive pairs of permutations (along with the XOR'ed key between them) into a single known permutation. In particular, [29] used this technique to devise the best known attack on LED-128, which can break 6-step LED-128, based on an attack on 2-round EM with identical subkeys. Similarly, we can leverage our attack on 3-round EM with identical subkeys to attack 8-round EM with alternating keys. This shows that the minimal number of rounds required for assuring $2n$-bit security for iterated EM with alternating n-bit subkeys is at least 9. The full details of the procedure are presented in Sect. 7, where it is applied to the lightweight block cipher LED-128 (which is an instance of an iterated EM with alternating subkeys).

6.3. *Extending Our Attacks to Iterated Even–Mansour Schemes with Linearly Derived Subkeys*

All of our attacks on 2-round and 3-round EM schemes with identical subkeys can be generalized to iterated EM schemes with linearly derived subkeys. Let us denote the first key as $K_1 = K$, and thus, $K_2 = L_2(K)$, $K_3 = L_3(K)$, and in 3-round iterated

EM, $K_4 = L_4(K)$, where L_2, L_3, L_4 are linear mappings. In order to attack such schemes, all of the steps in our algorithms which are based on the equality of the keys should be modified to handle the case where the keys are linearly dependent. This can be easily done using simple linear algebra without increasing the complexity of the algorithms. For example, our attacks on iterated EM schemes with one key are based on the property $x \oplus Q(K, x) = x \oplus K \oplus P(x \oplus K)$. In the more general scheme, we have $Q^*(K, x) = L(K) \oplus P(x \oplus K)$ (for some linear mapping L and permutation P), and it implies that $L(x) \oplus Q^*(K, x) = L(x \oplus K) \oplus P(x \oplus K)$. Thus, if we know that some values of the function $P''(z) \triangleq L(z) \oplus P(z)$ are more likely than expected (by finding t-way collisions in this function), then we can predict the value of $Q^*(K, x)$ with a higher probability than expected and exploit this property in the same way as in our attacks on schemes with identical subkeys.

The application of the attack to the former AES candidate Crypton (which can be viewed as iterated EM with linearly derived subkeys) is presented in the next section.

7. Applications

7.1. *LED-64 and LED-128*

LED is a 64-bit block cipher designed for resource-constrained environments, proposed by Guo et al. at CHES 2011 [20]. The two main variants of LED are LED-64 (which supports 64-bit keys) and LED-128 (which supports 128-bit keys). The design of LED can be viewed as a special case of iterated EM schemes: LED-64 is in fact an 8-step iterated EM scheme[12] with identical subkeys, and LED-128 is a 12-step iterated EM scheme with alternating subkeys K_1 and K_2. The inner permutations of LED are based on the AES design framework; however, since our attacks do not exploit any properties of these permutations, we do not specify them here and refer the reader to [20] for further details.

In the single-key model, the best attack published so far on reduced LED-64 breaks 2 steps of this cipher [13]. For LED-128, the largest number of attacked steps was 6 (see [29]). In this paper, we use our generic attacks in order to improve the data complexity of the attack on 6-step LED-128 from 2^{59} to 2^{45}, while keeping the time and memory complexities similar to the original attack. More significantly, we present the first single-key attacks which are faster than exhaustive search on 3-step LED-64 and on 8-step LED-128. The previously best known attacks on LED in the single-key model and our new attacks are summarized in Table 2. Note, in particular, that our new attack on 8-step LED-128 actually has a slightly better time complexity and requires about a thousand times less data than the best previous attack, which could only be applied to 6 steps of LED-128, out of the full 12.

7.1.1. *An Attack on 3 Steps of LED-64*

We can attack 3-step LED-64 by directly applying $3Round1KeyOpt$ attack with $n = 64$, presented in Sect. 4.3. Thus, the preprocessing phase has a time complexity of about

[12] In the design of LED, the term "step" is used in order to describe what we refer to as a "round" of an iterated EM scheme. On the other hand, a "round" of LED is used in order to describe a smaller component of its internal permutation. Thus, in order to avoid confusion, we will use the term "step" in this section.

Table 2. Single-key attacks of step-reduced LED.

References	Cipher	Steps	Time	Data	Memory
[13]	LED-64	2	2^{48}	2^{16} CP	2^{17}
This paper	LED-64	3	$2^{60.2}$	2^{49} KP	2^{60}
[29]	LED-128	6	$2^{124.4}$	2^{59} KP	2^{59}
This paper	LED-128	6	$2^{124.5}$	2^{45} KP	2^{60}
This paper	LED-128	8	$2^{123.8}$	2^{49} KP	2^{60}

The data complexity is given in chosen plaintexts (CP), or in known plaintexts (KP)

$2^{58.5}$ and memory complexity of 2^{60}. The online algorithm has a memory complexity of 2^{60}, data complexity of 2^{49} known plaintexts and time complexity of $2^{59.6}$. Since in this paper, we consider the preprocessing time as part of the attack (i.e., we assume that we are trying to attack the scheme for the first time), the total time complexity of the algorithm is about $2^{60.2}$, which is about 14 times better than exhaustive search.

7.1.2. *An attack on 6 Steps of LED-128*

As was pointed out in [27,29], it is easy to reduce $2r + 2$ steps of LED-128 (with its alternating use of two keys) into an iterated EM scheme variant with identical subkeys by guessing K_1 and combining consecutive pairs of permutations (along with the XOR'ed key between them) into a single known permutation. In particular, [29] leveraged their attack on 2-step iterated EM into an attack on 6-step LED-128. Similarly, we guess K_1, and for each guess, we partially encrypt and decrypt the given plaintext–ciphertext pairs and remain with a 2-step iterated EM scheme with identical subkeys all equal to K_2). Thus, we can apply our 2-step iterated EM attack (presented in Sect. 4.1) for each guess of K_1. However, we note that the preprocessing phase of our $2Round1KeyOpt$ attack should be executed for each guess of K_1, and it is thus now a part of the online algorithm of the attack on LED-128. Moreover, the algorithm can no longer be performed in streaming mode, as we need to reuse each plaintext–ciphertext pair for each guess of K_1. The general framework of the algorithm is given below.

1. Ask for the encryption of D arbitrary plaintexts and store them.
2. For each value of K_1:
   (a) Apply the $2Round1KeyOpt$ attack (including the preprocessing steps) on the resultant scheme, with plaintext–ciphertext pairs $(P_1(m_i \oplus K_1),\ P_6^{-1}(c_i \oplus K_1))$. Test each returned key using another pair (m_j, c_j).

Using the parameters of our $2Round1KeyOpt$ attack (presented in Sect. 4.1), the expected data complexity of the attack is 2^{45} known plaintexts and its memory complexity is 2^{60} (required for preprocessing, which is now part of the online algorithm). We calculate the expected time complexity of the algorithm as follows: adding the preprocessing and online time complexities, the main procedure of the attack performed for each guess of K_1 requires about $2^{60.1} + 2^{60} \approx 2^{61.1}$ evaluations of 4 out of the 6 permutations, which is equivalent to about $2^{60.5}$ evaluations of the full scheme. Compared to this complexity, the partial encryption and decryption of each (m_j, c_j) pair, and the trial

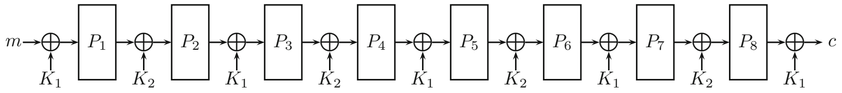

Fig. 5. 8-step LED-128.

encryptions using (m_j, c_j) (performed on average once per guess of K_1) are negligible. Thus, the expected time complexity of the attack is about $2^{64+60.5} = 2^{124.5}$, which is about 11 times better than exhaustive search.

7.1.3. *An Attack on 8 Steps of LED-128*

We use the same framework of our 6-step attack on LED-128 in order to attack 8 steps of LED-128 (shown in Fig. 5). Namely, we guess K_1, and for each guess, we partially encrypt and decrypt the given plaintext–ciphertext pairs and remain with a 3-step iterated EM scheme with one key (K_2). We then apply our $3Round1KeyOpt$ attack (presented in Sect. 4.3) for each guess of K_1. Thus, the memory complexity of the attack is 2^{60} and its data complexity is 2^{49} known plaintexts. We calculate the expected time complexity of the algorithm as follows: Adding the preprocessing and online time complexities, the main procedure of the algorithm performed for each guess of K_1 requires about $2^{58.5} + 2^{59.6} \approx 2^{60.2}$ evaluations of 6 out of the 8 permutations, equivalent to about $2^{59.8}$ evaluations of the full scheme. Thus, the expected time complexity of the attack is about $2^{64+59.8} = 2^{123.8}$, which is about 18 times better than exhaustive search.

7.2. *AES*^{2}

AES^2 is a 128-bit block cipher presented at Eurocrypt 2012 by Bogdanov et al. [7]. The cipher is a 2-round iterated EM construction, where each of the public permutations P_1 and P_2 is based on an invocation of full AES-128 with a prefixed and publicly known key. The designers of the scheme claim that its security is 2^{128}. However, the best attack known to the designers (as claimed in [7]) is the MITM attack presented in Sect. 5, and based on our analysis, it has a slightly higher time complexity of $3 \cdot 2^{128} \approx 2^{129.6}$ and a memory complexity of 2^{128}.

In order to attack AES^2, we use our $2Round3KeyOpt$ attack with $S_1 = 2^{124}$ and $S_2 = 2^{125.4}$. This implies that the memory complexities of both the preprocessing and online phases are $2^{125.4}$. The time complexity of the preprocessing phase is $S_1 + S_2 = 2^{124} + 2^{125.4} \approx 2^{125.9}$ evaluations of the full scheme. Using the formula $(2^n \cdot \lambda^t \cdot e^{-\lambda})/t!$ with $\lambda = 2^{124}/2^{128} = 1/16$ and $t = 18$, it is easy to check that we expect to find at least 10 entries in the difference distribution table with a value of 18 (we need only one). Plugging in these values into the formula $D = 2^{2(n-1)}/(t \cdot S_2)$, we obtain $D \approx 2^{124.4}$, implying that the data complexity of the attack is $2^{125.4}$ chosen plaintexts. The time complexity of the online attack is $2D + 0.375 \cdot 2^n/t \approx 2^{125.6}$, and adding the preprocessing time, the total time complexity of the algorithm is about $2^{125.9} + 2^{125.6} \approx 2^{126.8}$. This is better than the $2^{129.6}$ time complexity of the MITM attack by a factor of about 7 and clearly violates the 128-bit security claimed for AES^2 in [7]. We also note that the memory complexity is improved from 2^{128} to about $2^{125.4}$; however, the data complexity is greatly increased to $2^{125.4}$.

7.3. *Crypton*

Crypton is a 128-bit block cipher, supporting key sizes up to 256 bits. It was designed by Lim [26] and submitted as a candidate to the AES selection process. The structure of Crypton is similar to that of AES. It has 12 rounds where each round employs subkey addition and an SPN (Substitution-Permutation Network) structure for processing a 16-byte block. A short time after it was published, a tweaked version called Crypton v1.0. was presented by the designer [26], featuring a slight strengthening of the round transformation and a significant strengthening of the key schedule.

As the exact structure of the round function is not important for our attack, we omit its description and describe only the part of the key schedule that is important for us. In both Crypton and Crypton v1.0., the round subkeys are generated from the key in a two-phase process. First, an expanded 256-bit key is generated in a nonlinear way from the user-supplied key. Then, the expanded key is divided into two 128-bit words, which are used to derive the subkeys in a linear way. The crucial point exploited in our attack is that these two words are used *independently*: All subkeys of odd rounds are linear functions of the first 128 bits of the expanded key, and all subkeys used in the even rounds are linear functions of the last 128 bits of the expanded key. We note that in the transition to Crypton v1.0., the linear key derivation functions were replaced by significantly more complex functions, but their linearity and independence were preserved, thus not effecting our attack.

The best previous attacks on Crypton in the single-key model are two statistical attacks on 8-round Crypton with complexity of $2^{114.6}$ presented by Minier and Gilbert [28], and a truncated differential attack on 8-round Crypton v1.0. with complexity of $2^{126.8}$, presented by Kim et al. [22]. No attacks are known on more than 8 rounds of Crypton in the single-key model. We note that in the related-key model, an attack on 9-round Crypton was presented by Wei et al. [37].

Using our generic attack on iterated EM constructions, we obtain an attack on 8-round Crypton and Crypton v1.0. with 256-bit keys. Our attack is much slower than the attacks of [22,28]. However, it requires only known plaintexts and furthermore has the advantage that it is much more generic: It works in the same way even if the round function of Crypton is replaced by a very strong function, while both attacks of [22,28] use the internal properties of the round function of Crypton in a very strong way.

To attack 8-round Crypton, we combine two of the extensions of our attacks on iterated EM with identical subkeys presented in Sect. 6. As there is almost no practical difference between recovering the user-supplied key and recovering the expanded 256-bit key, we concentrate on recovering the expanded key. First, we guess the first 128 bits of the expanded key. For each such guess, we can treat every two consecutive rounds of Crypton (in which the key added in the middle is known), as a public permutation in an iterated EM construction. Thus, 8-round Crypton is reduced to 3-round iterated EM with linearly derived subkeys. Then, as described in Sect. 6, we can apply a simple variant of our $3Round1KeyOpt$ attack (presented in Sect. 4.3) for each guess of the first 128 bits of the expanded key.

We note that although Crypton has relatively weak round functions compared to the other ciphers considered in this paper, we did not find any reason that the relevant distribution to the $3Round1KeyOpt$ attack (namely[13] $L(x) \oplus P(x)$) would significantly deviate from that of a random function. Therefore, we analyze the attack on Crypton similarly to the $3Round1KeyOpt$ attack, as detailed below.

Reusing the analysis of the $3Round1KeyOpt$ attack with $n = 128$, let $S = 2^{123}$, i.e., $\lambda = 2^{123}/2^{128} = 2^{-5}$. We use the formula $(2^n \cdot \lambda^t e^{-\lambda})/t! = (2^{128} \cdot 2^{-5t} e^{-1/32})/t!$ with $t = 14$, such that we expect at least $\ell = 2^{20}$ vertices with an in-degree of 14. Thus, the memory complexity of the attack is about 2^{123}, and its data complexity is 2^{108} known plaintexts. Calculating the expected time complexity of the main procedure of the algorithm performed for each guess of the expanded K_1 (by adding the preprocessing and online time complexities of the $3Round1KeyOpt$ attack), we conclude that it requires less than 2^{125} evaluations of single permutations. This is equivalent to less than 2^{122} evaluations of the full 8-round scheme, and thus, the expected time complexity of the attack is about $2^{128+122} = 2^{250}$, which is about 64 times better than exhaustive search.

8. Conclusions

In this paper, we considered the security of iterated Even–Mansour schemes in the computational model. We aimed at partially answering the question: What is the minimal number of rounds r such that r-round iterated EM provides full security with respect to key recovery attacks? We showed that for iterated EM with identical subkeys, $r \geq 4$, and that for iterated EM with completely independent subkeys, $r \geq 3$. We then applied our techniques to devise the best known attacks (in terms of number of attacked rounds) on several block ciphers, including LED-64, LED-128 and AES^2. For standard values of the block size n, our attacks are between 7 and 64 times faster than exhaustive search, but they differ from other improvements of exhaustive search, as their improvement factor increases to infinity as n grows. Even though most of our attacks are not likely to be practically significant, they indicate that block ciphers based on the iterated EM scheme with identical subkeys should have at least 4 rounds, regardless of how strong we make their internal permutations.

References

[1] E. Andreeva, A. Bogdanov, Y. Dodis, B. Mennink, J. P. Steinberger, On the indifferentiability of key-alternating ciphers. in *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, ed. by R. Canetti, J.A. Garay (Springer, Berlin, 2013), pp. 531–550

[2] K. Aoki, Y. Sasaki, Preimage attacks on one-block MD4, 63-step MD5 and more. in *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, ed. by R.M. Avanzi, L. Keliher, F. Sica (Springer, Berlin, 2008), pp. 103–119

[3] P.S.L.M. Barreto, V. Rijmen, The ANUBIS Block Cipher. Submission to the NESSIE project, 2000

[4] P.S.L.M. Barreto, V. Rijmen, The Khazad Legacy-Level Block Cipher. Submission to the NESSIE project, 2000

[5] A. Biryukov, D. Wagner, Slide attacks. in Knudsen [23], pp. 245–259

[13] In the expression $L(x) \oplus P(x)$ considered in Sect. 6.3, L denotes the linear mapping between the derived subkeys and P denotes two consecutive rounds of Crypton in which a known subkey is added in the middle.

[6] A. Bogdanov, D. Khovratovich, C. Rechberger, Biclique cryptanalysis of the full AES. in *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, ed. by D.H. Lee, X. Wang (Springer, Berlin, 2011), pp. 344–371

[7] A. Bogdanov, L.R. Knudsen, G. Leander, F.-X. Standaert, J. P. Steinberger, E. Tischhauser, Key-alternating ciphers in a provable setting: encryption using a small number of public permutations - (extended abstract). in Pointcheval and Johansson [31], pp. 45–62

[8] S. Chen, J.P. Steinberger, Tight security bounds for key-alternating ciphers. in *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, ed. by P.Q. Nguyen, E. Oswald (Springer, Berlin, 2014), pp. 327–350

[9] J. Daemen, Limitations of the Even-Mansour construction. in *ASIACRYPT*, volume 739 of *Lecture Notes in Computer Science*, ed. by H. Imai, R.L. Rivest, T. Matsumoto (Springer, Berlin, 1991), pp. 495–498

[10] J. Daemen, M. Peeters, G.V. Assche, V. Rijmen, Nessie Proposal: NOEKEON. Submission to the NESSIE project, 2000

[11] I. Dinur, O. Dunkelman, N. Keller, A. Shamir. Key recovery attacks on 3-round Even-Mansour, 8-step LED-128, and full AES^2. in Sako and Sarkar [33], pp. 337–356

[12] I. Dinur, O. Dunkelman, N. Keller, A. Shamir, Cryptanalysis of iterated Even-Mansour schemes with two keys. in P. Sarkar, T. Iwata, eds. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science* (Springer, Berlin, 2014), pp. 439–457

[13] I. Dinur, O. Dunkelman, N. Keller, A. Shamir, Improved linear sieving techniques with applications to step-reduced LED-64. Presented at FSE 2014, to Appear to Lecture Notes in Computer Science, 2014

[14] O. Dunkelman, N. Keller, A. Shamir, Minimalism in cryptography: the Even-Mansour scheme revisited. in Pointcheval and Johansson [31], pp. 336–354

[15] S. Even and Y. Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *J. Cryptology*, 10(3):151–162, 1997.

[16] P. Flajolet, A.M. Odlyzko, Random mapping statistics. in *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, ed. by J.-J. Quisquater, J. Vandewalle (Springer, Berlin, 1989), pp. 329–354

[17] P. Flajolet, R. Sedgewick. *Analytic Combinatorics*. (Cambridge University Press, Cambridge, 2009)

[18] B. Gérard, V. Grosso, M. Naya-Plasencia, F.-X. Standaert, Block ciphers that are easier to mask: how far can we go? in *CHES*, volume 8086 of *Lecture Notes in Computer Science*, ed. by G. Bertoni, J.-S. Coron (Springer, Berlin, 2013), pp. 383–399

[19] B. Gérard, V. Grosso, M. Naya-Plasencia, F.-X. Standaert, Block ciphers that are easier to mask: how far can we go? Cryptology ePrint Archive, Report 2013/369, 2013. http://eprint.iacr.org/

[20] J. Guo, T. Peyrin, A. Poschmann, M.J.B. Robshaw, The LED block cipher. in *CHES*, volume 6917 of *Lecture Notes in Computer Science*, ed. by B. Preneel, T. Takagi (Springer, Berlin, 2011), pp. 326–341

[21] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[22] J. Kim, S. Hong, S. Lee, J. H. Song, H. Yang, Truncated differential attacks on 8-round CRYPTON. in *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, ed. by J.I. Lim, D.H. Lee (Springer, Berlin, 2003), pp. 446–456

[23] L.R. Knudsen, ed. *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24–26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science* (Springer, Berlin, 1999)

[24] R. Lampe, J. Patarin, Y. Seurin, An asymptotically tight security analysis of the iterated Even-Mansour cipher. in Wang and Sako [36], pp. 278–295

[25] R. Lampe, Y. Seurin, How to construct an ideal cipher from a small set of public permutations. in Sako and Sarkar [33], pp. 444–463

[26] C.H. Lim, A revised version of crypton - crypton V1.0. in Knudsen [23], pp. 31–45

[27] F. Mendel, V. Rijmen, D. Toz, K. Varici, Differential analysis of the LED block cipher. in Wang and Sako [36], pp. 190–207

[28] M. Minier, H. Gilbert, Stochastic cryptanalysis of Crypton. in *FSE*, volume 1978 of *Lecture Notes in Computer Science*, ed. by B. Schneier (Springer, Berlin, 2000), pp. 121–133

[29] I. Nikolic, L. Wang, S. Wu, Cryptanalysis of round-reduced LED. in *FSE*, volume 8424 of *Lecture Notes in Computer Science*, ed. by S. Moriai (Springer, Berlin, 2013), pp. 112–129

[30] L. O'Connor, On the distribution of characteristics in bijective mappings. in *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, ed. by T. Helleseth (Springer, Berlin, 1993), pp. 360–370

[31] D. Pointcheval, T. Johansson, eds. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15–19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science* (Springer, Berlin, 2012)

[32] S.M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*, 2 edn. (Academic Press, New York, 2000)

[33] K. Sako, P. Sarkar, eds. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1–5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science* (Springer, Berlin, 2013)

[34] H. Soleimany, Probabilistic slide cryptanalysis and its applications to LED-64 and Zorro. Presented at FSE 2014, to appear to Lecture Notes in Computer Science. 2014

[35] J. Steinberger, Improved security bounds for key-alternating ciphers via Hellinger distance. Cryptology ePrint Archive, Report 2012/481, 2012. http://eprint.iacr.org/

[36] X. Wang, K. Sako, eds. *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2–6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science* (Springer, Berlin, 2012)

[37] Y. Wei, C. Li, and B. Sun. Related-Key Impossible Differential Attacks on Crypton. *International Journal of Intelligent Computing Research*, 1(4):168–175, 2010.