More Rounds, Less Security?

Jian Guo[1], Jérémy Jean[1], Nicky Mouha[2,3], and Ivica Nikolić[1]

[1] Nanyang Technological University, Singapore
[2] Dept. Electrical Engineering-ESAT/COSIC, KU Leuven, Leuven and iMinds, Ghent, Belgium.
[3] Project-team SECRET, Inria, France.
ntu.guo@gmail.com, JJean@ntu.edu.sg, Nicky.Mouha@esat.kuleuven.be, inikolic@ntu.edu.sg

Abstract. This paper focuses on a surprising class of cryptanalysis results for symmetric-key primitives: when the number of rounds of the primitive is increased, the complexity of the cryptanalysis result *decreases*. Our primary target will be primitives that consist of identical round functions, such as PBKDF1, the Unix password hashing algorithm, and the Chaskey MAC function. However, some of our results also apply to constructions with non-identical rounds, such as the PRIDE block cipher. Firstly, we construct distinguishers for which the *data complexity* decreases when the number of rounds is increased. They are based on two well-known observations: iterating a random permutation increases the expected number of fixed points, and iterating a random function decreases the expected number of image points. We explain that these effects also apply to components of cryptographic primitives, such as a round of a block cipher. Secondly, we introduce a class of key-recovery and preimage-finding techniques that correspond to exhaustive search, however on a smaller part (e.g. one round) of the primitive. As the *time complexity* of a cryptanalysis result is usually measured by the number of full-round evaluations of the primitive, increasing the number of rounds will lower the time complexity. None of the observations in this paper result in more than a small speed-up over exhaustive search. Therefore, for lightweight applications, implementation advantages may outweigh the presence of these observations.

Keywords: Iterated cipher, fixed points, slide attack, PRIDE, Chaskey, PKCS, PBKDF1, Unix password hashing algorithm, Even-Mansour, FX-construction.

1 Introduction

How to determine the number of rounds of a symmetric-key primitive, for example of an iterated block cipher? O'Connor noted that "Most ciphers are secure after sufficiently many rounds," to which Massey replied that "Most ciphers are too slow after sufficiently many rounds" From this point of view, the challenge is to design a cipher that is both fast and secure.

But does increasing the number of rounds always make a primitive more secure? Although this is generally the case, it is not necessarily true. A common

counterexample are primitives that are vulnerable to slide attacks [7,8]. As stated by Biruykov in [6], slide attacks "realize the dream of cryptanalysts: if the cipher is vulnerable to such an attack, the complexity of the attack is independent of the number of rounds of the cipher." In that context, this paper describes what certainly must be the nightmare of any cryptographer. We will discuss cryptanalysis results where the complexity decreases when the number of rounds is increased.

Our basic observations hold for primitives that have identical round functions, such as Unix's `crypt(3)` and PKCS #5's PBKDF1 password hashing algorithms, or the Chaskey MAC function [24]. But as we will show, our results also apply to primitives with non-identical round functions, such as the recently proposed PRIDE block cipher [1].

The very nature of the results in this paper seem to defy logic, however designers do not need to worry too much: their practical impact is very low, as all of them are very close to exhaustive search. Nevertheless, this paper should be interesting from an academic point of view, given the interest in recent years into "brute-force-like" cryptanalysis. These include, for example, the recently proposed biclique results for many ciphers, including the full AES [9].

In this light, this paper is particularly interesting for the domain of lightweight cryptography. Designers may want to explicitly allow small speed-ups of exhaustive search, if such design choices lead to a more efficient implementation.

Our Contributions. In the domain of provable security, it is well-known that iterating an ideal primitive will result in a loss of security. This is already evident from the very first results in provable security, so we certainly do not claim a new result in that area. However, in the literature on the cryptanalysis of non-ideal primitives, we could not find any mention of cryptanalysis results for which the complexity *decreases* where the number of rounds of the primitive is *increased*. The goal of this paper is to perform the first comprehensive study on this subject. We give an overview of the existing literature, which typically involves ideal primitives. Then, we apply these insights to non-ideal components (e.g. one round of a block cipher) that arise in the field of cryptanalysis. We obtain new cryptanalysis results on ciphers built on the Even-Mansour block cipher and the FX-construction, which we then apply respectively to Chaskey and PRIDE.

Outline. After discussing related work in Sect. 2, we describe a framework for distinguishers on iterated primitives in Sect. 3, and speed-ups of exhaustive search on iterated primitives in Sect. 4. We apply them to a variety of primitives, and present new results for Chaskey MAC function, the PRIDE block cipher and a variant of PBKDF1 without a salt. We conclude the paper in Sect. 5, where we also provide suggestions for future work.

2 Related Work

Wagner and Goldberg [27] performed an analysis of the Unix password hashing algorithm, which consists of 25 applications of DES on an all-zero plaintext. The password of the user is used as the key of the DES algorithm. Note that this description omits the salt value, an important feature of the password hashing algorithm. However, Wagner and Goldberg do not analyze the effect of the salt in their paper, and neither will we for the sake of simplicity.

They analyzed this construction by observing its close relation to the security the CBC-MAC algorithm [3, 4]. Their security bound shows that Unix password hashes may not be uniformly random. However, as their analysis assumes that the adversary obtains only one password hash, this effect becomes negligible. The implications of changing the number of iterations is not considered in their paper.

The same effect was analyzed by Bard et al. [2], when they calculated the expected number of fixed points of a random permutation that is iterated k times. They found that it is equal to $\tau(k)$, where τ is the number-of-divisors function. For example, when this result is applied to Unix password hashing where $k = 25$, they calculated that the expected number of fixed points is $\tau(25) = 3$, as 25 has three divisors: 1, 5 and 25. Thus, by counting the number of fixed points, we can distinguish this construction from random. Note that for the particular case of Unix password hashing, this distinguisher will have a low, but nevertheless non-negligible success probability. The reason for the low success probability is that the length of the password (56 bits) is smaller than the length of the hash value (64 bits).

Bard et al. did not state that their cryptanalysis result becomes better when the number of rounds is increased. In fact, strictly speaking, such a statement would not be correct. There are infinitely many prime numbers, and if the number of rounds is a prime number, the expected number of fixed points is always two. However, the limit superior of $\tau(k)$ goes to infinity, and $\tau(k)$ is strictly increasing for commonly used subsets of k, such as powers of two: $\tau(2^\ell) = \ell + 1$, or powers of ten: $\tau(10^\ell) = (\ell + 1)^2$. Under this more narrow interpretation, we argue that this can be seen as a distinguisher for which the complexity decreases when the number of rounds is increased.

In subsequent work, Yao and Yin [28, 29] analyzed the two standardized password-based key derivation functions of PKCS #5: PBKDF1 and PBKDF2. PBKDF1 concatenates the password and salt, and then iteratively applies a hash function k times to the result. Note that k is not a fixed parameter, but depends on the implementation. They argue that PBKDF1 becomes more secure when k is increased, which they refer to as the effect of "key-stretching." However, the numerical examples from which they derive this statement, assume that the adversary performs only a small number of hash function evaluations. When we consider adversaries that make a very large number of hash function evaluations, the security bound on PBKDF1 becomes weaker when the iteration count k is increased.

Gligoroski and Klima [17] showed how this weaker security bound also corresponds to an observation on iterated random functions such as PBKDF1. They recalled a theorem by Flajolet and Odlyzko [16]: if an n-bit random function is iterated k times, the expected number of image points is $(1 - t_k) \cdot 2^n$ (for large n), where t_k satisfies the recurrence relation $t_0 = 0$ and $t_k + 1 = \exp(-1 + t_k)$. From this, it can be shown that the expected number of image points is 2^{n-i+1} when a random function is iterated $k = 2^i$ times. This approximation is valid for $i \geq 5$, and becomes more accurate when i increases. Note that the observations of Gligoroski and Kilma also hold if k independent random functions are used, instead of one random function that is iterated k times.

Using this theorem, Gligoroski and Klima constructed distinguishers for several iterated constructions, including PBKDF1 and several narrow-pipe hash functions. From their observations, they argued for wide-pipe instead of narrow-pipe hash functions. They did not, however, point out that increasing the number of rounds of a primitive makes their distinguishers more successful. This is of course evident from the formulas, but it was not part of their story line.

The same "entropy loss" was evaluated in the context of stream ciphers, for example by Hong and Kim [18] and by Röck [26] for the MICKEY stream cipher. However, the context is not the same as the one that we consider in this paper. Their analysis concludes that security decreases when more output bits are known, whereas we do not increase the number of output bits, but only the number of rounds of the primitive.

At the rump session of the ECRYPT II Hash Function Retreat in 2009 [12], Dunkelman showed how to speed up preimage search for the ESSENCE hash function [23] by the number of rounds. Unfortunately, this result was never published, and appears to be unknown to most of the research community. In Sect. 4, we will recall his observations and extend them to other constructions.

3 Distinguishers on Iterated Primitives

In this section, we will study the security of iterated primitives. Firstly, we will look at constructions where an ideal primitive is iterated, which can be a random function or a random permutation. We will provide short security proofs for these constructions, which can be seen as simplified versions of the proofs for CBC-MAC [3–5, 25].

Then, we will recall two distinguishers on these constructions: entropy-loss distinguishers and fixed-point distinguishers. As mentioned in Sect. 2, these distinguishers have been described in existing literature when a random function or a random permutation is used. We will explain how they also apply to constructions with non-ideal primitives, by presenting a fixed-point distinguisher for the Chaskey MAC function. Interestingly, doubling the number of rounds will *decrease* the complexity of this fixed-point distinguisher.

3.1 Security Bounds on Iterated Primitives

Let us assume that adversaries are computationally unbounded, and are only limited by the number of queries that they make. Without loss of generality, we assume that all queries made by an adversary are distinct. Let $\mathsf{perm}(n)$ denote the set of all permutations on n bits, and let $\mathsf{func}(n)$ be the set of all functions from n to n bits. We then define the advantage of an iterated primitive distinguisher as follows (see also Fig. 1).

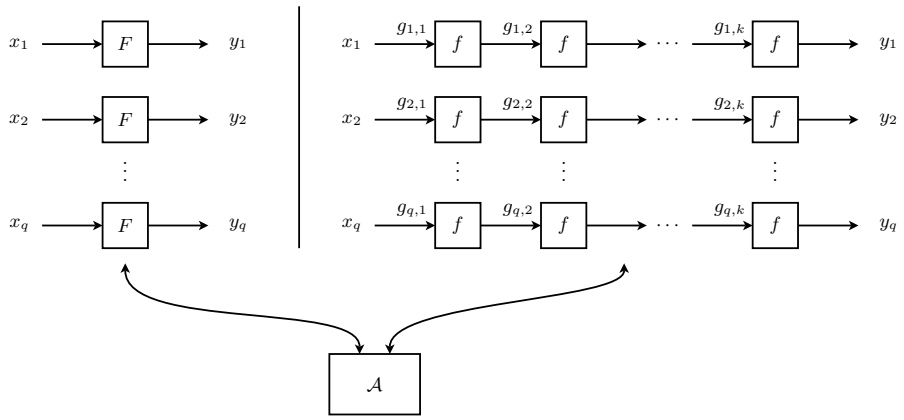

Fig. 1. Distinguishing between a random function (resp. permutation) F and a random function (resp. permutation) f that is iterated k times. The adversary $\mathcal{A}$ makes at most q queries. All queries are assumed to be distinct. In case F and f are permutations, the $\mathcal{A}$ can make both forward and inverse queries.

Definition 1 (Iterated Primitive Distinguisher). *Let F be a random variable over $\mathsf{perm}(n)$ (resp. over $\mathsf{func}(n)$). For an adversary $\mathcal{A}$, the advantage of distinguishing a random function (resp. random permutation) from the k-th iterate of a random function (resp. random permutation) is*

$$\mathbf{Adv}_{F,f^k}(\mathcal{A}) = \left|\Pr\left(\mathcal{A}^F \to 1\right) - \Pr\left(\mathcal{A}^{f^k} \to 1\right)\right| \ , \tag{1}$$

where the adversary can make both forward and inverse queries in case F and f are permutations. Note that the adversary is only given access to F and f^k, and not to f directly.

The maximum advantage could be derived from the security bounds of CBC-MAC, as was done by Wagner and Goldberg [27]. We now provide a much simpler proofs, and obtain tighter security bounds. Theorem 1 considers the case where both F and f are random functions. In Theorem 2, both F and f are random permutations.

Theorem 1 (Iterated Random Function Security). *Let F be a random function, and f^k be a random function that is iterated k times. Then for all $\mathcal{A}$ making at most q queries,*

$$\mathbf{Adv}_{F,f^k}(\mathcal{A}) \leq \frac{kq(kq-1)}{2^{n+1}} - \frac{q(q-1)}{2^{n+1}} \quad . \tag{2}$$

Proof. *Let $\mathbf{E_1}$ be the event that there exist $g_{i,j} = g_{i',j'}$ such that $\neg(i = i' \wedge j = j')$. Given that $\mathbf{E_1}$ does not happen, no input of f will be reused. Note that trivial reuse occurs when there exist ℓ, ℓ' such that $x_\ell = x_{\ell'}$, but we do not have to consider this as we assumed w.l.o.g. that the adversary makes distinct queries. Under the negation of $\mathbf{E_1}$, all queries are independent of each other, and the reply to every query follows a uniformly random distribution, so that F and f^k are indistinguishable. By the fundamental lemma of game playing, $\mathbf{Adv}_{F,f^k} \leq \Pr(\mathbf{E_1})$. From the union bound, the probability that $g_{i,j} = g_{i',j'}$ where $\neg(i = i' \wedge j = j')$, is at most $0/2^n + 1/2^n + \cdots + (kq-1)/2^n = kq(kq-1)/2^{n+1}$. However, given that all queries are distinct, we know that $g_{i,1} \neq g_{i',1}$ for $i \neq i'$, so we can subtract $0/2^n + 1/2^n + \cdots + (q-1)/2^n = q(q-1)/2^{n+1}$ from this expression to obtain a more accurate bound. We then have $\Pr(\mathbf{E_1}) \leq kq(kq-1)/2^{n+1} - q(q-1)/2^{n+1}$, from which the theorem follows.* □

Theorem 2 (Iterated Random Permutation Security). *Let F be a random permutation, and f^k be a random permutation that is iterated k times. Then for all $\mathcal{A}$ making at most q queries,*

$$\mathbf{Adv}_{F,f^k}(\mathcal{A}) \leq \frac{kq(kq-q)}{2^{n+1}} \quad . \tag{3}$$

Proof. *Let $\mathbf{E_2}$ be the event that $g_{i,j} = g_{i',j'}$ such that $\neg(i = i' \wedge j = j')$. As inverse queries are allowed, we must also consider the case that an output of f is reused, i.e. that there exist $f(g_{i,j}) = f(g_{i',j'})$ such that $\neg(i = i' \wedge j = j')$. As f is a permutation, this can be seen as equivalent to the first statement by applying f^{-1} to both sides of the equation. As such, the number of queries contributes to the bound, but the direction of each query does not make a difference. Following a similar reasoning as in the previous proof, we find that F and f^k are indistinguishable given the negation of $\mathbf{E_2}$, and that $\Pr(\mathbf{E_2}) \leq kq(kq-1)/2^{n+1} - kq(q-1)/2^{n+1}$. Here, we subtracted k times $q(q-1)/2^{n+1}$: because f is a permutation and all queries are distinct, we have $\forall 1 \leq j \leq k : g_{i,j} \neq g_{i',j}$ for $i \neq i'$. Therefore, $\Pr(\mathbf{E_2}) \leq kq(kq-q)/2^{n+1}$, from which the theorem follows.* □

3.2 Entropy-Loss Distinguishers and Fixed-Point Distinguishers

In Sect. 2, we described two distinguishers for iterated primitives. We will refer to these as entropy-loss distinguishers for the case of random functions, and as fixed-point distinguishers when random permutations are considered. As we explained, these distinguishers have already been applied to several iterated constructions, including the Unix password hashing algorithm and PBKDF1. For each of these constructions, the underlying building blocks were assumed to be ideal.

We introduce a new result in this paper, by looking at a non-ideal building block that is iterated, such as a round of a block cipher. We find that the same distinguishers apply, although their complexity is difficult to evaluate. To back up our findings, we perform experiments of small-scale variants of ciphers. These distinguishers have a lower data complexity when the number of rounds is increased. They apply regardless of the underlying algorithm, and even when the algorithm is not known to the adversary. In the next section, we show an application of this idea to the Chaskey MAC function.

3.3 Application to Chaskey

Chaskey [24] is a Message Authentication Code (MAC) algorithm designed by Mouha et al. as a collaboration between KU Leuven-COSIC and Hitachi YRL. Currently, Chaskey is in a study period for standardization by ISO/IEC JTC 1/SC 27/WG 2. Standardization is expected to start in October 2015, when its maturity is confirmed. ITU-T SG17 has recently added new work items related to IoT and ITS security, for which Chaskey seems to be well-suited.

Although Chaskey uses a dedicated mode of operation to process variable-length messages, in this paper we will focus only on one-block messages. In that case, Chaskey uses a 128-bit key K to transform a 128-bit message M into a tag T of at most 128 bits. It does this using a variant of the Even-Mansour block cipher [14, 15]: $C = E_{K_1,K_2}(P) = K_2 \oplus \pi(P \oplus K_1)$, where K_1 and K_2 are related to K by a linear function. Here, the plaintext P is equal to the one-block message, and the ciphertext C corresponds to the (untruncated) tag T. The Chaskey permutation π consists of eight identical rounds, one of which is shown in Fig. 2. A sixteen-round variant of Chaskey, named Chaskey-LTS, was proposed as well.

The derivation of the subkeys K_1 and K_2 from K goes as follows. Let us interchangeably consider an element a of $GF(2^{128})$ as the integer $2^{127}a[127] + 2^{126}a[126] + \ldots 2^0 a[0]$ in decimal notation, and as the polynomial $a(x) = a[n-1]x^{n-1} + a[n-2]x^{n-2} + \ldots + a[0]$ with binary coefficients. To multiply two elements a and b, we represent them as two polynomials $a(x)$ and $b(x)$, and calculate $a(x)b(x) \bmod f(x)$ where $f(x) = x^{128} + x^7 + x^2 + x + 1$. Using this notation, the subkeys of Chaskey are $K_1 = 3K$ and $K_2 = 2K$.

The claimed security of the Chaskey block cipher is about $T = 2^{128}/D$ permutation evaluations when D plaintext-ciphertexts are available. We will refer to T and D as the time and the data complexity, respectively. The currently best known attack on Chaskey is by Leurent [22] on 7 rounds out of 8 rounds, and uses between 2^{45} and 2^{48} chosen plaintexts.

In the fixed-point distinguishers that we will now consider, we will have $T = 0$. This means that we will not perform a single permutation evaluation: the distinguisher does not depend on the permutation π, and even works if the algorithmic description of π is unknown to the adversary. Going back to the setting of Even-Mansour [14, 15], this means that the entire codebook (all 2^{128} plaintexts and ciphertexts) should appear as if it was generated by a random permutation.

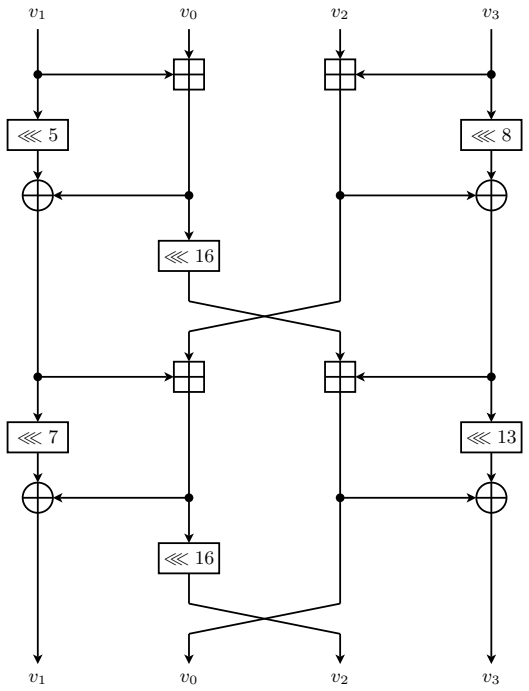

Fig. 2. One round of the Chaskey permutation π, defined as: $v_0\|v_1\|v_2\|v_3 \leftarrow \pi(v_0\|v_1\|v_2\|v_3)$. The values v_0 and v_1 are intentionally swapped, as this reduces the number of crossing lines in the figure.

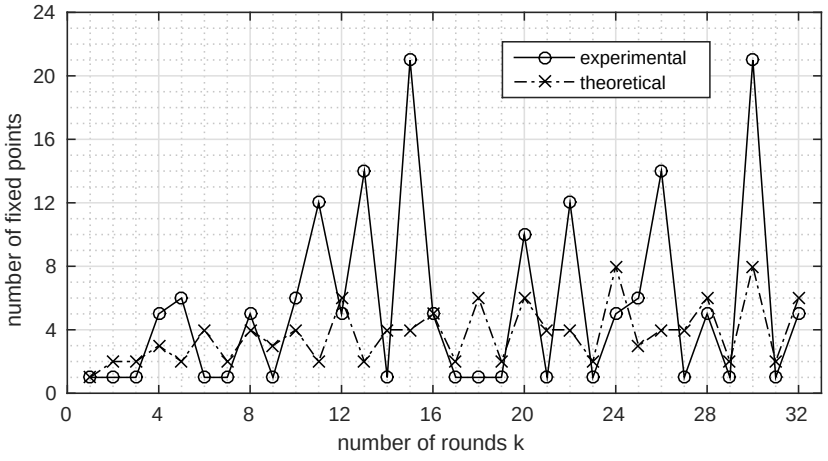

Fig. 3. Number of fixed points for $1 \leq k \leq 32$ rounds of SmallChaskey, compared to the expected number of fixed points if every round were a random permutation.

Any number of rounds of the Chaskey permutation always has at least one fixed point: the all-zero input. The search for additional fixed points seems to be a computationally difficult problem, even for only one round of Chaskey. For this reason, we performed some experiments on SmallChaskey, a variant of Chaskey that we introduce in this paper. In SmallChaskey, all 32-bit words are replaced by 8-bit words, and the rotation constants 16, 8, 13, and 7 are replaced by 4, 2, 6, and 3, respectively.

The results are plotted in Fig. 3. As we can see from this figure, the number of fixed points is significantly higher than one, which is the number of fixed points expected for a random permutation. It also seems to be sometimes much higher than expected for an iterated random permutation. We conjecture that this behavior also holds for the full-size Chaskey and for other cryptographic round functions.

If this is the case, an adversary can distinguish Chaskey from a random permutation by collecting close to $D = 2^{128}$ plaintext-ciphertext pairs and counting the number of fixed points. This distinguisher has no impact on the Chaskey MAC function, which restricts the data complexity available to the adversary to at most 2^{64} plaintext-ciphertexts. Nevertheless, this result distinguishes the Chaskey block cipher from an Even-Mansour cipher from a random permutation, which may be of independent academic interest.

4 Speeding Up Exhaustive Search

The distinguishers that we presented in Sect. 3 have the property that the *data complexity* decreases when the number of rounds is increased. The time complexity was not a parameter, in fact the distinguishers even work if the underlying algorithm is not known to the adversary.

We now look at cryptanalysis results where the *time complexity* decreases when the number of rounds is increased. These results hold the conventional model that is used in cryptanalysis: the time complexity is calculated in terms of the equivalent number of full-round computations.

However, we assume that the time complexity is only determined by the number of round evaluations. Any other computations that are performed by the adversary, are not taken into account. As a result of this, the speed-up over exhaustive search may be less than our simplified model shows. If the overhead of these other computations is significant, it may even be that there is only a speed-up over exhaustive search if the cipher uses a sufficiently large number of rounds.

Note that our analysis uses computational model that is common for meet-in-the-middle and biclique-style cryptanalysis. However, those types of cryptanalysis can be prevented by increasing the number of rounds. This is not the case for our observations. In fact, our observations have a lower complexity when the number of rounds is increased.

After recalling an observation by Dunkelman et al. on the ESSENCE hash function, we will introduce new speed-ups over exhaustive search for the Chaskey

MAC function, for the PRIDE block cipher, and for a variant of PBKDF1 without a salt.

4.1 ESSENCE

Let us recall the speed-up over exhaustive preimage search by Dunkelman [12] on the ESSENCE hash function. The ESSENCE compression function consists of 32 identical rounds, shown in Fig. 4. A preimage for the compression function (a "pseudo-preimage" for the hash function) can be found as follows. Let X be the target compression function value, so that we are looking for a message M and a chaining value CV for which $C(CV, M) = X$, where C is the ESSENCE compression function.

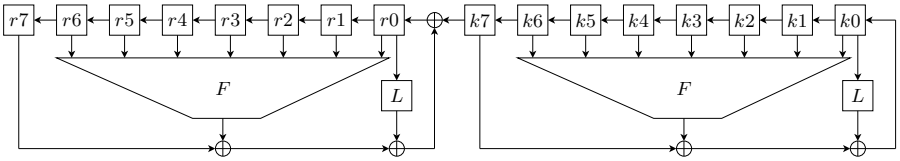

Fig. 4. One round of the ESSENCE hash function. The chaining value CV is loaded into the r_i registers, whereas the message M is loaded into the k_i registers. Every register is either 32 or 64 bits, resulting in a hash value of either 256 or 512 bits. Function F is a bitwise non-linear function, and L is a linear function operating on an entire word. ESSENCE consists of 32 rounds, after which a feed-forward is applied.

1. Initialize the message M and the chaining value CV with random values.
2. Iterate the ESSENCE round function 32 times, storing all intermediate values of M and CV.
3. Apply the feed-forward function, and check if the resulting compression output corresponds to the target value X.
4. If not, apply only one additional round of ESSENCE, and return to the previous step.

For the sake of completeness, we should note that a pseudo-preimage may not exist for the target value X, in which case this algorithm (and any other algorithm) will fail. The algorithm will also fail if it cycles back to the initial (M, CV) without encountering the target value X. This problem can be solved by selecting a new values of (M, CV) and restarting the algorithm.

If we assume that the round ESSENCE round function evaluations dominates any of the other calculations, then Dunkelman's observation effectively speeds up the search for a pseudo-preimage by the number of rounds. In the conventional security model where the attack complexity is determined by the equivalent number of compression function evaluations, this means that the time complexity goes down when the number of rounds is increased.

4.2 Chaskey

We now describe a speed-up of exhaustive search using known plaintexts on the Chaskey MAC algorithm. In the observation, all messages consist of one block, and the tags are not truncated. As already explained in Sect. 3.3, the Chaskey MAC function then becomes an Even-Mansour construction with two keys K_1 and K_2.

Unfortunately the chosen-plaintext attack by Daemen [11] and the known-plaintext attacks by Biryukov and Wagner [8] and by Dunkelman et al. [13] do not seem to apply here. In those attacks, the adversary must be able to evaluate the permutation π on inputs of its choosing. This is not possible when we apply the technique explained in Sect. 4.1 to speed up exhaustive search.

We therefore introduce a new cryptanalysis result on the Even-Mansour construction used in Chaskey, using the fact that the two keys K_1 and K_2 are derived from the key K as $K_1 = 3K$ and $K_2 = 2K$. The result goes as follows. First, D known plaintexts (P_i, C_i) are obtained, which are encrypted under the secret key K. Each known plaintext can be transformed into another known plaintext (P'_i, C'_i), encrypted under an unknown key K'_i, where $P'_i = 0$, $C'_i = C_i \oplus 2 \cdot 3^{-1} P_i$ and $K \oplus 3^{-1} P_i$. It is easy to check from the key schedule that both (P_i, C_i) and (P'_i, C'_i) are valid plaintext-ciphertext pairs (under keys K and K' respectively), as input and output to the underlying permutation π remain the same. Then, store (C'_i, P_i) into a hash table, indexed by the first coordinate.

We then obtain T input-output pairs (x_j, y_j) of the permutation π. This is done using the same trick as for ESSENCE (Sect. 4.1): once one input-output pair of the permutation is calculated, any additional input-output pairs can be obtained from only one additional round evaluation. If an all-zero plaintext were encrypted, (x_j, y_j) would correspond to a plaintext (P_j, C_j) under a known key K_j, where $P_j = 0$, $C_j = y \oplus 2 \cdot 3^{-1} x$ and $K_j = 3^{-1} x$. We therefore check for each (x_j, y_j) if C_j appears as the first element in the hash table. If this is the case for element i, we can check the guess $K = 3^{-1}(P_i \oplus x_j)$.

The probability that $C_i = C_j$ for any i, j is 2^{-128}. As our data contains $TD = 2^{128}$ pairs (C_i, C_j), we will find a match with a non-negligible probability of success. As each of the T permutation evaluations can be generated with an equivalent time complexity of T/k, where k is the number of rounds of the cipher, exhaustive search is effectively sped up with a factor of k. Of course, this analysis assumes that any other calculations besides the Chaskey round function evaluations are negligible. If this is not the case, the speed-up over exhaustive search will be smaller than k. Regardless of this assumption, the time complexity our observation will decrease when the number of rounds is increased.

All results in this section were experimentally verified on a small-scale variant of Chaskey.

4.3 PRIDE

PRIDE is a lightweight block cipher proposed by Albrecht et al. [1]. It processes a 64-bit plaintext P using a 128-bit key K, which is split into two 64-bit

keys K_0 and K_1. Similar to PRINCE [10], PRIDE is also based on the FX-construction [20, 21] and also claims a security of $TD = 2^{128}$, where T and D are again, respectively, the time and data complexity of any cryptanalysis result. The PRIDE construction is illustrated in Fig. 5. Our description of PRIDE omits a bit permutation that is applied to the plaintext and ciphertext, and adds a diffusion layer to the last round, so that every round becomes identical. As such, our description of PRIDE is equivalent up to linear functions that are applied to the plaintext and ciphertext. An adversary can easily apply these functions to any plaintext-ciphertext, as they do not require knowledge of the key.

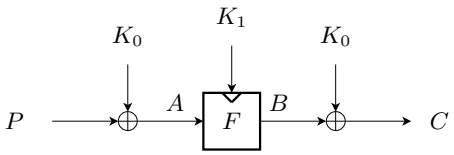

Fig. 5. The PRIDE block cipher, which processes 64-bit plaintext P using a 128-bit key K, which is split into two 64-bit keys K_0 and K_1. Key K_0 is used for prewhitening and postwhitening. The block cipher F is a 20-round Substitution-Permutation Network (SPN) that uses K_1.

PRIDE has a very simple key schedule. The designers claim no resistance against related-key attacks, and in fact note that PRIDE can be distinguished trivially in this setting. For this reason, we will tackle the security bound $TD = 2^{128}$, by showing a speed-up over exhaustive search to go below this bound in the cryptanalysis model used in this paper. But first, we describe the round function and key schedule of PRIDE.

The block cipher F used inside PRIDE is a 20-round Substitution-Permutation Network (SPN). It consists of a subkey addition (the XOR of $f_i(K_1)$ for round i), followed by a substitution layer S and a permutation layer P, as shown in Fig. 6. The subkeys $f_i(K_1)$ are derived from K_1 as follows. Let $\|$ denote the concatenation of binary strings. First, we split K_1 into eight bytes: $K_1 = u_1\| \dots \|u_8$. Then,

$$f_i(K_1) = u_1\|u_2 + 193i\|u_3\|u_4 + 165i\|u_5\|u_6 + 81i\|u_7\|u_8 + 197i \ , \qquad (4)$$

where all operations are performed modulo 2^8.

This results in the following pair of slid keys K_1, K_1':

$$K_1 = a_1\|a_2 \quad \|a_3\|a_4 \quad \|a_5\|a_6 \quad \|a_7\|a_8 \ , \qquad (5)$$
$$K_1' = a_1\|a_2 + 193\|a_3\|a_4 + 165\|a_5\|a_6 + 81\|a_7\|a_8 + 197 \ , \qquad (6)$$

where again all calculations are performed modulo 2^8.

This observation could be used to construct a slide attack under related keys K_1, K_1'. However, related-key attacks do not violate the PRIDE design criteria.

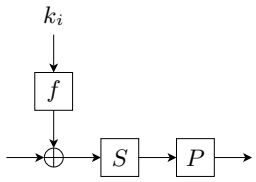

Fig. 6. One round of the PRIDE Substitution-Permutation Network (SPN), where S denotes the substitution layer and P refers to the permutation layer.

We can, however, use this pair of slid keys to speed up exhaustive search. This can be done as follows.

At the core of the attack, we will speed up the generation of (A, B)-values for the block cipher F, shown in Fig. 5. A full-round encryption is required to obtain one (A, B)-value under an adversary-chosen key K_1, after which each additional round will result in another (A', B'), where the relation between K_1 and K_1' is given by (5)-(6).

For the FX-construction used in PRIDE, the existing attacks proceed as follows: iterate over all keys K_1, and then apply an attack of the Even-Mansour block cipher. This strategy is used in the chosen-plaintext attack by Kilian and Rogaway [20, 21], and in the known plaintext attack for $D = 2^{n/2}$ by Biryukov and Wagner [8], where n denotes the block size. In a straightforward way, the attack by Dunkelman et al. [13] on Even-Mansour also can be turned into a known plaintext attack on the FX-construction for any value of D.

However, none of these attacks seem to be applicable to speed up exhaustive search for PRIDE. This is because all of them require a loop over K_1, whereas our speed-up will generate (A, B)-values under random keys K_1. For this reason, we now introduce a new cryptanalysis result on the FX-construction. First, D known plaintexts (P_i, C_i) are collected, each encrypted under the secret key K. The values $(P_i \oplus C_i, P_i)$ are stored in a hash table, using the first coordinate as the index. Then for each of the T values $(A_j, B_j, K_{1,j})$, look up $A_j \oplus B_j$ in the hash table. If a match is found, we can check the key guess $K_0 = P_i \oplus A_j$, $K_1 = K_{1,j}$. To reach a non-negligible probability of success, this key must be checked with at least one other plaintext-ciphertext pair.

The probability that $P_i \oplus C_i = A_j \oplus B_j$ is 2^{-64}. As the data consists of $TD = 2^{128}$ pairs, we expect to find 2^{64} matches. When checking each of these against another plaintext-ciphertext, we expect that one candidate key will survive. Note that each of the T block cipher evaluations were generated with an equivalent time complexity of T/k. Therefore, this effectively speeds up exhaustive search by a factor of k.

We have experimentally verified all parts of this observation on a small-scale variant of PRIDE.

4.4 Password Hashing: PKCS #5's PBKDF1

We now show how exhaustive search for password hashing algorithms can be sped up, in particular for PKCS #5's PBKDF1. Recall the PBKDF1 construction from Sect. 2: the password and salt are concatenated, after which a hash function is iteratively applied k times to obtain the password hash. The PKCS #5 standard recommends to use at least $k = 1000$, however in recent real-world applications, k is often ten or a hundred times higher. More specifically, in this section we will consider a variant of PBKDF1 that does not use a salt value.

Let n be the output size of the hash function. Given one password hash, a classical preimage search requires about $T = 2^n$ evaluations of PBKDF1 to obtain a non-negligible success probability. If D password hashes are given, a straightforward calculation shows that recovering any of these requires about $T = 2^n/D$ PBKDF1 evaluations.

Now observe that the time complexity can be reduced by a factor of k. This is because evaluating one password guess requires k evaluations of the hash function used inside PBKDF1, but every additional guess has an additional cost of only one hash function evaluation. This effectively speeds up exhaustive search by a factor of k: given D password hashes, recovering any of them has a time complexity of $2^n/(D \cdot k)$. Yet again, this cryptanalysis result has a time complexity that decreases when the number of rounds is increased.

We constructed a small-scale variant of this password hashing function to verify our results experimentally.

5 Conclusion and Future Work

This paper focused on cryptanalysis results on symmetric-key primitives for which the complexity goes down when the number of rounds is increased. To the best of our knowledge, this paper provided the first comprehensive study of these types of observations. We investigated two classes of observations for iterated symmetric-key constructions: distinguishers and speed-ups of exhaustive search.

The distinguishers exploited the fact that iterating identical round functions always leads to security erosion. The expected number of fixed points increases when a random permutation is iterated, whereas iterating a random function reduces the expected number of image points. We explained how these effects also appear for non-ideal primitives, and used this to construct a new observation for the Chaskey MAC function.

When the underlying primitives are ideal, interestingly, none of the aforementioned distinguishers are tight. Between the security bounds of these constructions, for which we presented simple proofs in this paper, and the currently best known cryptanalysis results, there is a large gap.

From this perspective, the most promising target for future work may be on iterated random permutations: to either prove a better security bound, or to improve the current cryptanalysis. When a random permutation is iterated

a prime number of times k, the expected number of fixed points is always two. This result holds regardless of the number of iterations k, which seems to indicate that improvements can be made here, in particular when k is large.

For ciphers that consist of identical rounds, we explained that after one output has been evaluated, any subsequent input-output pairs can be obtained at the cost of only one additional round evaluation. We showed how to use this observation to speed up exhaustive search for several algorithms, including Chaskey, PRIDE, and an unsalted variant of the PKCS #5's PBKDF1 password hashing algorithm.

As we explained, the actual speed-up of exhaustive search is difficult to evaluate, but nevertheless decreases when the number of rounds is increased. As a prerequisite for our speed-ups over exhaustive search, developed two new variants of existing attacks: one on the Even-Mansour construction, and another on the FX-construction.

As our analysis of the PRIDE block cipher showed, our results also apply to a block cipher with non-identical round functions. An interesting direction for future work is to see if our speed-ups of exhaustive search can be applied to other ciphers with non-identical rounds.

Acknowledgments. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by Research Fund KU Leuven, OT/13/071, and by the French Agence Nationale de la Recherche through the BLOC project under Contract ANR-11-INS-011. Nicky Mouha is supported by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO-Vlaanderen), and by a JuMo grant from KU Leuven (JuMo/14/48CF). Jérémy Jean and Ivica Nikolić are supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block Ciphers - Focus on the Linear Layer (feat. PRIDE). In Garay, J.A., Gennaro, R., eds.: Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Volume 8616 of LNCS., Springer (2014) 57–76
2. Bard, G.V., Ault, S.V., Courtois, N.T.: Statistics of Random Permutations and the Cryptanalysis of Periodic Block Ciphers. Cryptologia **36**(3) (2012) 240–262
3. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In Desmedt, Y., ed.: Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Volume 839 of LNCS., Springer (1994) 341–358
4. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. **61**(3) (2000) 362–399
5. Bernstein, D.J.: A short proof of the unpredictability of cipher block chaining (January 2005) `http://cr.yp.to/antiforgery/easycbc-20050109.pdf`.
6. Biryukov, A.: Slide Attack. In van Tilborg, H.C.A., Jajodia, S., eds.: Encyclopedia of Cryptography and Security, 2nd Ed. Springer (2011) 1221–1222

7. Biryukov, A., Wagner, D.: Slide Attacks. In Knudsen, L.R., ed.: Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings. Volume 1636 of LNCS., Springer (1999) 245–259
8. Biryukov, A., Wagner, D.: Advanced Slide Attacks. In Preneel, B., ed.: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Volume 1807 of LNCS., Springer (2000) 589–606
9. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of LNCS., Springer (2011) 344–371
10. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Wang, X., Sako, K., 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 LNCS., Springer (2012) 208–225
11. Daemen, J.: Limitations of the Even-Mansour Construction. [19] 495–498
12. Dunkelman, O.: An Observation on PRIDE. Presented at the Rump Session of the ECRYPT II Hash Function Retreat (2009)
13. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In Pointcheval, D., Johansson, T., eds.: EUROCRYPT. Volume 7237 of LNCS., Springer (2012) 336–354
14. Even, S., Mansour, Y.: A Construction of a Cipher From a Single Pseudorandom Permutation. [19] 210–224
15. Even, S., Mansour, Y.: A Construction of a Cipher from a Single Pseudorandom Permutation. J. Cryptology **10**(3) (1997) 151–162
16. Flajolet, P., Odlyzko, A.M.: Random Mapping Statistics. In Quisquater, J., Vandewalle, J., eds.: Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings. Volume 434 of LNCS., Springer (1989) 329–354
17. Gligoroski, D., Klima, V.: Practical Consequences of the Aberration of Narrow-Pipe Hash Designs from Ideal Random Functions. In Gusev, M., Mitrevski, P., eds.: ICT Innovations 2010. Volume 83 of Communications in Computer and Information Science., Springer (2011) 81–93
18. Hong, J., Kim, W.: TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY. In Maitra, S., Madhavan, C.E.V., Venkatesan, R., eds.: Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings. Volume 3797 of LNCS., Springer (2005) 169–182
19. Imai, H., Rivest, R.L., Matsumoto, T., eds.: Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. In Imai, H., Rivest, R.L., Matsumoto, T., eds.: ASIACRYPT. Volume 739 of LNCS., Springer (1993)
20. Kilian, J., Rogaway, P.: How to Protect DES Against Exhaustive Key Search. In Koblitz, N., ed.: Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Volume 1109 of LNCS., Springer (1996) 252–267
21. Kilian, J., Rogaway, P.: How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). J. Cryptology **14**(1) (2001) 17–35

22. Leurent, G.: On Chaskey. Presented at the Early Symmetric Crypto - ESC 2015 (2015)
23. Martin, J.W.: ESSENCE: A Family of Cryptographic Hashing Algorithms. Submission to the NIST SHA-3 Competition (Round 1) (2008) `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html`.
24. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In Joux, A., Youssef, A.M., eds.: Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. Volume 8781 of LNCS., Springer (2014) 306–323
25. Nandi, M.: A Simple and Unified Method of Proving Indistinguishability. In Barua, R., Lange, T., eds.: Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings. Volume 4329 of LNCS., Springer (2006) 317–334
26. Röck, A.: Stream Ciphers Using a Random Update Function: Study of the Entropy of the Inner State. In Vaudenay, S., ed.: Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings. Volume 5023 of LNCS., Springer (2008) 258–275
27. Wagner, D., Goldberg, I.: Proofs of Security for the Unix Password Hashing Algorithm. In Okamoto, T., ed.: Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Volume 1976 of LNCS., Springer (2000) 560–572
28. Yao, F.F., Yin, Y.L.: Design and Analysis of Password-Based Key Derivation Functions. In Menezes, A., ed.: Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings. Volume 3376 of LNCS., Springer (2005) 245–261
29. Yao, F.F., Yin, Y.L.: Design and Analysis of Password-Based Key Derivation Functions. IEEE Transactions on Information Theory **51**(9) (2005) 3292–3297