Multi-user collisions: Applications to Discrete Logarithm, Even-Mansour and PRINCE (Full version*)

Pierre-Alain Fouque[1] and Antoine Joux[2] and Chrysanthi Mavromati[3]

[1] Université Rennes 1, France and Institut Universitaire de France, France
[2] CryptoExperts, France and Chaire de Cryptologie de la Fondation de l'UPMC
Laboratoire d'Informatique de Paris 6, UPMC Sorbonne Universités, France
[3] Sogeti/ESEC R&D Lab, France
Université de Versailles Saint-Quentin-en-Yvelines, France
pierre-alain.fouque@univ-rennes1.fr,
antoine.joux@m4x.org,chrysanthi.mavromati@sogeti.com

Abstract. In this paper, we investigate the multi-user setting both in public and in secret-key cryptanalytic applications. In this setting, the adversary tries to recover keys of many users in parallel more efficiently than with classical attacks, *i.e.*, the number of recovered keys multiplied by the time complexity to find a single key, by amortizing the cost among several users. One possible scenario is to recover a single key in a large set of users more efficiently than to recover a key in the classical model. Another possibility is, after some shared precomputation, to be able to learn individual keys very efficiently. This latter model is close to traditional time/memory tradeoff attacks with precomputation. With these goals in mind, we introduce two new algorithmic ideas to improve collision-based attacks in the multi-user setting. Both ideas are derived from the parallelizable collision search as proposed by van Oorschot and Wiener. This collision search uses precomputed chains obtained by iterating some basic function. In our cryptanalytic application, each pair of merging chains can be used to correlate the key of two distinct users. The first idea is to construct a graph, whose vertices are keys and whose edges are these correlations. When the graph becomes connected, we simultaneously recover all the keys. Thanks to random graph analysis techniques, we can show that the number of edges that are needed to make this event occurs is small enough to obtain some improved attacks. The second idea modifies the basic technique of van Oorschot and Wiener: instead of waiting for two chains to merge, we now require that they become *parallel*.
We first show that, using the first idea alone, we can recover the discrete logarithms of L users in a group of size N in time $\widetilde{O}(\sqrt{NL})$. We put these two ideas together and we show that in the multi-user Even-Mansour scheme, *all* the keys of $L = N^{1/3}$ users can be found with $N^{1/3+\epsilon}$ queries for each user (where N is the domain size). Finally, we consider the PRINCE block cipher (with 128-bit keys and 64-bit blocks) and find the keys of 2 users among a set of 2^{32} users in time 2^{65}. We also describe a new generic attack in the classical model for PRINCE.

1 Introduction

The multi-user setting is a very interesting practical scenario, which is sometimes overlooked in cryptography. Indeed, cryptosystems are designed to be used by many users, and usually cryptographers prove the security of their schemes in a single-user model except in some cases such

as key exchange, public-key encryption and signatures. At EUROCRYPT 2012, Menezes [22] gave an invited talk pointing out the discrepancy between security proofs for message authentication code in the single-user and in the multi-user setting. As it was already been pointed out in [11], he showed that there is a straightforward reduction between the security proof for one user and the security proof for L users with a success probability divided by L. Next, he recalled the key collision attack due to Biham [3] that matches this bound and that can be applied on various deterministic MACs (CMAC, SIV, OCB, EME, ...). In this attack, the adversary asks the MAC tag of a single message M for L different users; we call this the set of *secret* MACs. Then, for a subset W of size N/L of known keys (N is the key size), he computes $MAC(k, M)$ for all $k \in W$ and builds the set of *public* MACs. If a collision occurs between the public and secret sets, then we learn one of the L secret keys.[4] For MAC schemes with an 80-bit security level, it is possible with time/memory tradeoff to make this reasonably practical and derive a key recovery of a single key among a set of $L = 2^{20}$ users, using time and memory 2^{40}. Menezes thus insists that cryptographers have to consider this practical setting when devising or analyzing cryptosystems. For more results on multi-user attacks, the reader can also refer to [4].

In this paper, we are interested in collision-based attacks [26] in the multi-user setting. We rely on the distinguished point technique to propose new attacks on the generic discrete logarithm problem, on the Even-Mansour cipher and on PRINCE. Collision-based methods have been nicely improved by van Oorschot and Wiener to become parallelizable using the distinguished point technique of Rivest and Quisquater and Delescaille [24]. Here, we extend these methods and apply them to cryptanalysis in the multi-user setting.

Our Contributions. From a cryptanalytic point of view, there are many ways to perform attacks in the multi-user setting. In this paper, we are interested by several scenarios. The first option is to recover all the users' keys (or a large fraction thereof) in time less than the product of the number of users by the time complexity to recover one key. Another direction is to improve Biham's attack and recover a single key in the multi-user setting with a reduced memory cost. Finally, we consider time/memory attacks starting with a precomputation whose result can then be used later to recover individual keys much faster.

Giant connected component. The multi-user setting for the discrete logarithm problem has been studied by Kuhn and Struik in [19]. They show that it is possible to adapt the parallel version of the Pollard rho technique with distinguished points to recover L keys in time $\sqrt{NL}$ where N is the size of the group as long as $L \ll \sqrt[4]{N}$. In the parallel version of Pollard rho method described by van Oorschot and Wiener, (see App. C) we run random walks in parallel, stop them once a distinguished point is reached and store this value for many starting points. We get a *public* set of distinguished points for the walks that begin at $y_a = g^a$ for which we know a and a *secret* set from a user public key y for starting points yg^b where b is known. Kuhn and Struik generalize this method by using many secret sets, one for each user. Once a distinguished point appears twice in the public and secret sets, the discrete logarithm of one user can be discovered, and consequently, we also know the discrete logarithm of all the distinguished points that were discovered during the random walks for this user. Therefore, as the number of "known" points increases, the probability of a collision between a secret point and a known one becomes higher. Similar results can be found in [21, 1, 2].

Here, we show another method that works without any restriction on L and keeps the symmetry between all read points. Indeed, we do not have to wait until the first collision

[4] Provided that the tag length is greater than the key length.

between a public point and a secret one happens, but we also consider collisions between secret points. More precisely, as soon as a collision between the public walks and the secret walks happens, we learn many discrete logarithms, since when two secret chains collide, we learn the difference between the discrete logarithm. We can then construct a graph whose vertices are the users and we add an edge if we know the difference of the discrete logarithm between these users. At some point, when the number of edges becomes slightly larger than the users, a giant component emerges in our random graph and if the public user is in this component (with high probability in time $2L \ln L$), then the discrete logarithm of *all* users will be known.

Our method has an advantage towards the method proposed by Kuhn and Struik as we use parallelism extensively. However, a disadvantage is that in our case we do not learn any discrete logarithms until the very end, when a giant component appears in the graph. In contrast, Kuhn and Struik's algorithm is sequential and so they find each discrete logarithm one after the other. Overall, the main goal of section 2 is to provide an educational example of the graph connexity approach and show that it is much simpler to analyze.

Lambda Method for two different Even-Mansour style functions. We were also able to apply similar techniques on Even-Mansour with domain size N. Indeed, using some functions related to the encryption scheme, we show that we can learn the Xor between the keys of two users. The previous technique can also be used to recover the keys of all users. However, in this case, we get a new problem: the two functions we iterate are no longer the same. Consequently, contrary to the DL case, once a collision appears, the chains will no longer merge and we cannot use distinguished point technique. To solve this issue, we tweak the two functions and define related functions that will no longer merge but become parallel. We show that this parallel method is as efficient as the previous one. For instance, we show an attack that partially solves an open problem of Dunkelman *et al.* that asked to find a memoryless attack on Even-Mansour with D queries to the secret function and $T = N/D$ to the public function with $D \ll \sqrt{N}$. We propose an attack that matches these bounds ($D = N^{2/5}, T = N^{3/5}$) but where the memory is $N^{1/5}$ as an application of our lambda-method. Furthermore, we also describe a multi-user attack which allows to learn *all* the keys in a set of $N^{1/3}$ users in data complexity $N^{1/3+\epsilon}$ to each user and $T = N^{1/3+\epsilon}$ time complexity by combining the two algorithmic tools. This attack exhibits new tradeoff where the amortized data complexity per user times the time complexity is reduced to $N^{2/3+\epsilon}$ instead of N.

Application to PRINCE. PRINCE cipher [8] is a new block cipher recently introduced at ASIACRYPT 2012 with blocklength 64 bits and keylength 128 bits. Its design has a α-reflection property which is a related-key relation that transforms the decryption algorithm to the encryption process with a related-key. Here, we propose generic attacks on the full number of rounds. At FSE 2014 [9], an attack on 10 rounds of PRINCE has been presented, with time complexity $2^{60.7}$ and data complexity $2^{57.94}$. In [16], an attack with slightly less than 2^{128} allows to break all the rounds, but our attacks have a particular low time complexity. They are similar to the one on Even-Mansour but we have to take into account that in PRINCE, the internal permutation uses a secret key. They make use both of the α-reflection property and of the specific key scheduling of PRINCE, *i.e.* the relationship between the two whitening keys. The first attack allows to recover the keys of two users among a set of 2^{32} users in time 2^{65} and the second one allows to recover the keys of all users in time 2^{32} after a precomputation of time 2^{96} and 2^{64} in memory. Finally, we do not contradict the security bound showed in the original paper, but we show that different tradeoffs are possible.

Organization of the paper. In section 2, we present our results on the discrete logarithm problem in the multi-user setting and we use the properties of random graph in this setting.

Then, we present various results concerning the security of Even-Mansour: new time/memory/data (denoted T/M/D) tradeoffs, new time/memory (denoted T/M) attack solving the open problem of Dunkelman *et al.* and in the multi-user setting. In this part, we show how we can adapt the lambda-method when searching for collisions for two different functions based on the Even-Mansour idea. Finally, in the last section, we present various generic attacks on the PRINCE block cipher, one in the multi-user setting and the other in the classical model.

2 Discrete Logarithms in the Multi-User Setting

In this section, we present a new algorithmic idea for performing T/M attacks with distinguished points in the multi-users setting. Our technique allows to compute the discrete logarithms of L public keys $y_i = g^{x_i}$ for $i = 1, \ldots, L$ in time[5] $\widetilde{O}(\sqrt{NL})$ for any value of L where $N = |\langle g \rangle|$. Starting from the parallel version of Pollard rho method [26], we compute $cL/2$ chains consisting of pseudo-random walks from y_i ($c/2$ chains for each user by randomizing the starting point) until we discover a distinguished point $d_i \in S_0$ where S_0 denotes the set of distinguished points[6]. Then, all distinguished points found are sorted and each collision between the distinguished points of different users d_i and d_j reveals a linear relation between x_i and x_j. We also compute a few chains starting from random points for which the discrete logarithm is known g^{x_0}. Finally, we construct the random graph where the vertices are the public keys and we add an edge between y_i and y_j if we have a collision between d_i and d_j (this process can be described more formally using a random graph processas it is recalled in Appendix A). This edge is labelled with the linear relation between x_i and x_j. Once we have computed a sufficient number of collisions, a small constant time the number of users, then a giant component will appear with high probability. More precisely, in a graph with L vertices and $cL/2$ randomly placed edges with $c > 1$, there is a giant component whose size is almost exactly $(1 - t(c))L$, (see [7]recalled in Appendix A) where:

$$t(c) = \frac{1}{c} \sum_{k=1}^{\infty} \frac{k^{k-1}(ce^{-c})^k}{k!}.$$

For $c = 4$, we get $1 - t(c) = 0.98$. The discrete logarithm of all the points in the component of the x_0's are known. If we want to recover the discrete logarithm of *all* users with overwhelming probability, we need $2L \ln L$ edges to connect all connected components according to the coupon collectors problem and not $cL/2$, as it is recalled in Theorem 2(see Appendix A).

Let ℓ the average length of the chains and S_0 the set of distinguished points. The average length of each chain is $\ell = N/|S_0|$. Assume we have computed i chains that do not collide, the probability that the $(i+1)^{\text{th}}$ chain collides with one of the previous is $i\ell \times \ell/N$(according to Theorem 3 in Appendix A for the expected number of collisions between two pseudo-random walks). Consequently, the expected number of collisions Coll is:

$$\mathbb{E}[\mathsf{Coll}] = \sum_{i=1}^{L-1} \frac{i\ell^2}{N} \approx \frac{L^2}{2} \cdot \frac{\ell^2}{N} = \frac{L^2}{2} \cdot \frac{(N/|S_0|)^2}{N} = \frac{L^2 N}{2|S_0|^2}.$$

We want the number of collisions to be larger than $cL/2$, which implies $L^2N/2|S_0|^2 \geq cL/2$, thus $|S_0| \leq \sqrt{LN/c}$. Consequently, the overall cost is dominated by the computation of the chains, *i.e.* $L \times N/|S_0|$ which is about $\sqrt{cLN}$ if $|S_0| = \sqrt{LN/c}$. Finally, in order to have $cL/2$ edges in our graph, each user has to compute a small number of chains using a small number

[5]The $\widetilde{O}$ notation hides logarithmic terms.

[6]This algorithm can also be adapted to the Pollard-lambda algorithm [23].

of random input points of the form $g^{x_i+r_i}$ for known value of r_i. The overall complexity of our attack is $\widetilde{O}(\sqrt{NL})$ for any value of L while Kuhn and Struik analysis achieves the value $\sqrt{2LN}$ for $L \ll \sqrt[4]{N}$.

Another possible approach to analyze known, unknown points and collisions between them would be to use a matrix. For this, we consider a symmetric matrix M where $M[i,j]$ represents the linear relation between the discrete logarithms of i and j. Then we apply a random variable in order to sparsify the matrix. More precisely, we multiply the coefficient (i,j) of the matrix by 1 with probability p and by 0 with probability $(1-p)$, where these probabilities are independent. When we multiply by 1, that means, that we know the differences between the discrete logarithms of i and j.The question then becomes how many rows (with 2 non-zero coefficients) do we need to achieve full column-rank, which naturally leads to the same results: $O(L * \log(L))$. However, when considering rows with $O(\log(L))$ non-zero coefficients, we only needs $O(L)$ rows. This would imply that for multi-user discrete logarithms the overall complexity can be reduced by a factor $\log(L)$ to $O(sqrt(L*N))$ by spending a factor $\log(L)$ more work in generating starting points of random combinations of $\log(L)$ known/unknown points (e.g., see [12]). We choose to analyze the complexity in the same form as Wiener and van Oorschot which is usually the case for crypto papers, *i.e* we do not care on the $\log N$ factors that arise in such birthday algorithms. Indeed, the Kuhn and Struik algorithm hides also a $\log(N)$ factor in order to get collisions with very high probability because a $1/2$ probability is not sufficient since we need many collisions of this type.

3 Even-Mansour in the Single and Multi-User Settings

3.1 Brief description of Even-Mansour

At Asiacrypt 1991, Even and Mansour in [15] describe a very efficient design (called EM in the following) to construct a block cipher, i.e. a keyed permutation family Π_{K_1,K_2} from a large permutation π. The key K_1 is first xored with the plaintext, then the fixed permutation is applied and finally the key K_2 is xored to obtain the final value.

$$\Pi_{K_1,K_2}(P) = \pi(P \oplus K_1) \oplus K_2.$$

Their main result is a security proof that any attack that uses D on-line plaintext/ciphertext pairs (queries to Π) and T off-line computations (queries to π) must satisfy $DT = N$, where $N = 2^n$ with n the size of the plaintext and key and which will be called the EM curve. The important part of the proof is that it is a lower bound for all attacks including *known-plaintext* attacks. It appears that the use of two keys K_1 and K_2 does not add much more resistance to the scheme. This variant of using $K = K_1 = K_2$ has been proposed under the name *Single-Key Even-Mansour* and we denote it by Π_K. The security of this minimal version has been proved secure with the same bound as for the two-key version by Dunkelman *et al.* This minimal version is amazingly resistant and guarantees the same security bound, but it is not unexpected since usually the attacks look for the two keys independently and once the key K_1 is recovered, there is no security for K_2. In the following, we see that the two-key version does not improve the security since most of the attacks on the single-key can be levered to this version.

In this section, we describe new results concerning the security of the Even-Mansour scheme which has recently been the subject of many papers [14, 20]. We recall the basic attacks and then, we present a basic T/M tradeoff for known plaintext attacks with better on-line complexity (Sect. 3.3) and a better T/M tradeoff for adaptive queries (Sect. 3.4). For this attack, we introduce our second algorithmic trick to discover collisions for two different functions based on the Even-Mansour construction. The main difficulty we have to solve is that when a lambda-like

method is used to recover collisions, if two different functions are used, after the collision, the chain will no longer merge. To this end, we adapt the lambda-method to have parallel chains when the collision happens. Finally, we show that in the multi-user setting (Sect. 3.5) the precomputation cost can be amortized. It is possible to balance all the complexities to recover all the keys of $N^{1/3}$ users with $N^{1/3+\epsilon}$ adaptive queries to each user, a precomputation time of $N^{1/3+\epsilon}$ and the attack requires $N^{1/3+\epsilon}$ in memory and $N^{1/3+\epsilon}$ for the on-line time.

3.2 Previous attacks on Even-Mansour

In [13], Daemen showed that the EM curve $TD = N$, is valid for a known plaintext attack at the point $(T = N/2, D = 2)$. He also gave a chosen-plaintext attack that matches the EM curve for any value of D and T and in particular at the point $(T = N^{1/2}, D = N^{1/2})$. Later, Biryukov and Wagner described a sliding attack that matches the EM curve for known-plaintext but only at the point $(T = N^{1/2}, D = N^{1/2})$(See Appendix B for more details about previous attacks). Recently, Dunkelman *et al.* introduce a new twist on the sliding attack whose complexities match the whole curve for any value of D and T using a known-plaintext attack which is exactly the result proved by Even and Mansour. Finally, Dunkelman *et al.* also provide a slidex attack on the two-key Even-Mansour scheme.

Simpler collision-based attack on the Single-Key Even-Mansour. In the single-key case a simpler attack achieves the same performance. The basic idea is to apply the Davies-Meyer construction to Π and to π. More precisely, write:

$$F_\Pi(x) = \Pi(x) \oplus x \quad \text{and } F_\pi(x) = \pi(x) \oplus x.$$

For any value of x, the equality $F_\Pi(x) = F_\pi(x \oplus K)$ is satisfied. Moreover, any collision between these two functions $F_\Pi(x) = F_\pi(y)$ indicates that $x \oplus y$ is a likely candidate for the key K.

With this idea in mind, the problem of attacking the single key Even-Mansour scheme is reduced to the problem of finding a collision (or rather a few collisions) between F_Π and F_π. The simplest approach is simply to compute F_π on T distinct random values and F_Π on D distinct random values. When $DT \approx N$, one expects to find the required collisions.

Moreover, this can be done in a more efficient way by using classical collision search algorithms with reduced memory. Indeed, it is possible to use Floyd's cycle finding algorithm to obtain such a solution for the special case $D = T = N^{1/2}$, without using memory. However, in this case the attack is no longer a known-plaintext attack and becomes an adaptively chosen plaintext attack.

Dunkelman, Keller and Shamir ask whether it is possible to generalize this and to find memoryless attacks using D queries to Π and N/D to π where $D \ll N^{1/2}$?

In this paper, we partially answer this question, proposing attacks that use less than $D \ll N^{1/2}$ data and memory lower than $\min(T, D)$ if we require the unkeyed queries to be precomputed. Without this requiring, we achieve a memoryless attack.

3.3 Extending the simple attack

Dealing with two keys Even-Mansour. A first important remark is that the simple attack on Single-Key EM can be extended to the two-key case. The idea is simply to replace the function $\pi(x) \oplus x$ by another function with similar properties. A first requirement is that the chosen function needs to be expressed by two different formulas, one based on π and the other on Π. The other requirement is that a collision on two evaluations, one of each type, should yield good candidates for the keys.

We now construct the required function and show that the simple attack on the single-key variant can be extended to two keys. We first choose a random non-zero constant δ and let:

$$F_{\Pi}(x) = \Pi(x) \oplus \Pi(x \oplus \delta) \text{ and } F_{\pi}(x) = \pi(x) \oplus \pi(x \oplus \delta).$$

We remark that $F_{\Pi}(x) = F_{\pi}(x \oplus K_1)$ and that $F_{\Pi}(x \oplus \delta) = F_{\pi}(x \oplus K_1)$ are both satisfied. As a consequence, every collision now suggests two distinct input keys $K_1 = x \oplus y$ and $K_1 = x \oplus y \oplus \delta$. Except for this detail, the attack remains unchanged. Note that once K_1 has been found, recovering K_2 is a trivial matter.

Reducing the on-line time complexity. In this section, we focus on *known-plaintext* attacks and we first show that the EM security model does not separate the on-line and off-line time complexities, as usually done in T/M/D tradeoff. It is then possible to use T/M/D tradeoff for this blockcipher design as suggested in [5] by Biryukov and Shamir.

Let us separate the on-line time denoted by T_{on} and the off-line time denoted by T_{off}. Clearly, the total time complexity T is $T_{\text{on}} + T_{\text{off}}$.

The main idea of this section is to use a different approach to find a collision between F_{Π} and F_{π}. More precisely, given a value of F_{Π}, we try to invert F_{π} on this value. If we succeed, we clearly obtain the desired collision. In order to inverse F_{π}, we rely on Hellman's algorithm(recalled in Appendix D). The T/M/D tradeoff is

$$T_{\text{on}} M^2 D^2 = N^2 \text{ and } D^2 \leq T_{\text{on}} \leq N.$$

In order to fully use Hellman tradeoff with multiple tables, we can use the δ in the definition of the function $F_{\pi}(x) = \pi(x) \oplus \pi(x \oplus \delta)$ to define different and independent functions for each table. These attacks achieve $T_{\text{on}} D \ll N$ while $TD = N$.

Using less data than memory. Despite its optimal efficiency in term of known-plaintext attack matching the EM curve, the Slidex attack presents an important drawback. Indeed, the public permutation π needs to be evaluated at points which depend on the result of the queries to the keyed Even-Mansour construction Π. As a consequence, with this attack, it is not possible to precompute the queries to π in order to improve the online time required to obtain the key to Π.

Our previous attack based on Hellman's tables no longer requires adaptive queries, however, it is less costly than the Slidex attack in term of on-line time complexity but more costly than the simple collision-based attack (which uses adaptive chosen plaintext). The goal of the next subsection is to present an attack on Π, which is based on classical collision search algorithms and works by using queries to π and Π without any cross-dependencies. However, the queries to Π are adaptive but this new attack is more flexible to perform T/M tradeoff.

3.4 Time/Memory/Data tradeoff attack on Even-Mansour

Attacking Even-Mansour using distinguished points methods. In order to attack Even-Mansour using a distinguished point methodas recalled in Appendix C, we would like to construct a set of chains using the public permutation π and then find a collision with a chain obtained from the keyed permutation Π. One difficulty is that chains computing from π and from Π can never merge since they are based on different functions contrary to discrete logarithm section. We introduce here a new idea to solve this dilemma when the functions are based on the Even-Mansour construction. Let us define:

$$F_{\Pi}(x) = x \oplus \Pi(x) \oplus \Pi(x \oplus \delta) \text{ and } F_{\pi}(x) = x \oplus \pi(x) \oplus \pi(x \oplus \delta).$$

We remark that $F_{\Pi}(x \oplus K_1) = F_{\pi}(x) \oplus K_1$. As a consequence, two chains based on F_{Π} and F_{π} cannot merge, but they may become *parallel*. Indeed, using the equation $F_{\Pi}(x \oplus K_1) = F_{\pi}(x) \oplus K_1$ and let two points X and x such that $X = x \oplus K_1$, where X (resp. x) belongs to an F_{Π} chain (resp. x belongs to an F_{π} chain), the next element $Y = F_{\Pi}(X)$ in the F_{Π} chain and the next element $y = F_{\pi}(x)$ in the F_{π} chain will satisfy:

$$Y = F_{\Pi}(X) = F_{\Pi}(x \oplus K_1) = F_{\pi}(x) \oplus K_1 = y \oplus K_1.$$

So $Y = y \oplus K_1$, which means that Y and y satisfy the same relation as X and x, and so on. Therefore, as soon as by chance $X = x \oplus K_1$ where X is an element of an F_{Π} chain and x is an element of an F_{π} chain, the same relation remains with the subsequent points of the two chains, i.e. we get two parallel chains.

Moreover, the detection of this good event is compatible with the distinguished point method. Indeed, it suffices to define a distinguished point x as a point with a value of $\pi(x) \oplus \pi(x \oplus \delta)$ in S_0. Similarly, for chains constructed by using F_{Π}, we define a distinguished point X as a point with a value of $\Pi(X) \oplus \Pi(X \oplus \delta)$ in S_0. Now if $X = x \oplus K_1$ and x is a distinguished point in a π chain, then since

$$\Pi(X) \oplus \Pi(X \oplus \delta) = \pi(X \oplus K_1) \oplus \pi(X \oplus K_1 \oplus \delta) = \pi(x) \oplus \pi(x \oplus \delta),$$

the point X is also a distinguished point in the Π chain, and therefore $X \oplus x$ gives a candidate for K_1. Since the values $\pi(x) \oplus \pi(x \oplus \delta)$ and $\Pi(X) \oplus \Pi(X \oplus \delta)$ are needed to compute the next element in the chains, using this definition does not add any extra cost for distinguished point detection. The important point, is that for a parallel chain based on F_{Π}, a point $X = x \oplus K_1$ corresponds to a distinguished point x if and only if $\Pi(X) \oplus \Pi(X \oplus \delta)$ is in S_0.

An important difference compared to the classical search for collisions is that we do not need to backtrack to the beginning of the chains and identify where the chains merge. Indeed, seeing parallel distinguished points suffices to get candidates values for K_1.

Analysis of the attack with precomputation. Since there is a clear symmetry between the keyed and unkeyed queries, we may assume that the number of unkeyed queries T is larger than the number of keyed queries D. Let B_T the number of unkeyed chains to increase the probability of a collision between keyed and unkeyed chains. Moreover, this is the most reasonable scenario, since keyed queries are usually the most constrained resource. In this case, we need to choose the expected length ℓ of the chains we are going to construct and B_T that satisfy the following relations:

$$T = \ell \cdot B_T \quad \text{and} \quad N = B_T \ell^2.$$

Thus, $\ell = N/T$ and $B_T = T^2/N$. The required memory to store those chains is of size $O(B_T)$.

After terminating the computation of the unkeyed chains, we can turn to the keyed side. On this side, we want to perform about $D = N/T$ evaluations of the function. Since $D = \ell$, this means that we compute a single keyed chain and expect it to (parallel) collide with an unkeyed chain.

We are interested in values for M such that $M < D$. Consequently, as $M = T/D = N/D^2$, we have $N < D^3$. Let us consider $N^{1/3} < D = N^{\alpha} < N^{1/2}$. For example, if $D = N^{2/5}$ and $T = N^{3/5}$, then $M = N^{1/5}$ is much smaller than $N^{2/5}$. This attack requires a number of data $D \ll N^{1/2}$ and despite this attack is not memoryless (as in the open problem), the memory is less than the data.

Relaxing the precomputation requirement. Another alternative[7] is to perform the same attack while computing the keyed queries before the unkeyed ones. In this case, since there is a single keyed chain to be stored, we can achieve the attack using a constant amount of memory. Moreover, this variation works for any $D = N^{\alpha} \leq N^{1/2}$ using $T = N/D$.

3.5 Attacks in the multi-user setting

In the multi-user setting, we assume that L different users are all using the Even-Mansour scheme based on the same public permutation π, with each user having its own key[8], chosen uniformly at random and independently from the keys of the other users.

Of course, the attack from Section 3.4 can be easily applied in this context. Depending on the exact goal of the cryptanalysis, we have two main options:

1. If the goal is to recover the key of all users, the previous attack can be applied by repeating the D key-dependent queries for each user, while amortizing the T unkeyed queries across users. A typical case is to consider $L = N^{1/3}$ users, to perform $T = N^{2/3+\epsilon}$ unkeyed queries ($N^{1/3+\epsilon}$ chains of $N^{1/3}$ queries, memory $N^{1/3}$). For each new user, we need $N^{1/3+\epsilon}$ key-dependent queries. As a consequence, the amortized cost per user (up to constant factors $c_0 = 20$) is $N^{1/3+\epsilon}$ queries of each type and the required memory also is $N^{1/3}$.
2. If the goal of the cryptanalyst is to obtain at least one user key among all the users, it suffices to split the D key-dependent queries arbitrarily across the users.

However, we present in this section a much more efficient tradeoff in the multi-user setting. This tradeoff becomes possible without precomputation in $N^{2/3}$, but by distributing the unkeyed queries among the users and by reusing the graph algorithmic idea of the section 2. For this, we construct a graph whose vertices are labelled by the users. Whenever we obtain a collision ${F_{\Pi}}^{(i)}(x) = {F_{\Pi}}^{(j)}(y)$ for users i and j, we add an edge between the corresponding vertices labelled with $x \oplus y$ which is expected equal to $K^{(i)} \oplus K^{(j)}$. Note that this indicates that we know the exclusive-or of the first keys of the two users.

If we have L vertices and $cL/2$ randomly edges with $c = 4$, there is a giant component whose size is 98% of the points, and with $cL \ln L$, all the points are in this component with overwhelming probability(see Theorems 1 and 2 in Appendix A). Consequently, we obtain the exclusive-or of the first keys for an arbitrary pair of users. To conclude the attack, it suffices to find a single collision between any of the users functions F_{Π} of the large connected component and the unkeyed function F_{π} to reveal all the keys of these users.

Algorithm Description.

1. Create a constant number $c/2$ of chains for each user up to a distinguished point.
2. Sort the distinguished points.
3. Bring together the distinguished points into subsets, where we test whether the key candidate is really the good one. It is indeed easy to check with a few more queries if the xor of two keys is correct.
4. Construct the giant component and expect that the public user (the user with the unkeyed function), lies in this giant component. To this end, we initially begin with the set of reachable users containing only the public user. Then, we add to this set all the users that are in a group where a reachable user is present. At some point, the reachable set is stable and we stop.

[7] We thank an anonymous reviewer of Asiacrypt 2014 for pointing this out.

[8] Or key-pair depending on whether we are considering the single or dual key scheme.

5. From the public user, we cross over the giant component and determine the keys of each user.

The first step requires $cL\ell/2$ data and time $O(c\ell)$ on average per user where ℓ is the average length of the chains. Then, the remaining steps are performed in time linear in the number of users L. Typical parameters are: for an arbitrary small positive constant c, we expect with $N^{1/3}$ users, $c \cdot N^{1/3}$ queries per user and $N^{1/3}$ unkeyed queries, to recover almost all the $N^{1/3}$ keys with overwhelming probability. If we want to recover *all* users, we need to have $L \ln L = cN^{1/3} \ln N = N^{1/3+\epsilon}$ edges (instead of $cL/2$) to connect all components according to the coupon collector's problemrecalled in Theorem 2 in Appendix.

Analysis of the attack. We want to use results from graph theory to prove the correctness of our algorithm, this means that we have to prove that the assumptions of the giant component theorem are satisfied. We have to show that we construct of a random graph according to the Erdös-Rényi model of random graphs, in which each possible edge connecting pairs of a given set of L vertices is present, independently of the other edges, with probability p. In this case, we know that with this model of random graph, if the number of edges $c.L/2$ is larger than the number of vertices L, there is with high probability a single giant component, with all other components having size $O(\log L)$ according to [7](also recalled in Theorem 1 in Appendix).

Consequently, we need to prove that we construct a random graph and that the edges are added *independently* of each others. We will define an idealized version of the attack and we will show that the attack works in this version. Then, we will prove that the idealized version and the attack are equivalent using simulation argument.

In the idealized model, the simulator randomly chooses L keys $K_1, \ldots, K_L$ uniformly at random. Then it iterates the functions $F_\Pi^{(i)}(x) = K_i \oplus F_\pi(x \oplus K_i)$ until $x_\ell \oplus K_i \in S_0$, where S_0 is the set of pairs containing a distinguished point d_i and an identificator of this point $id(d_i)$. The identificators are unique, which means that we do not have collision on them. Finally, the simulator reveals the identificator of the point $x_\ell \oplus K_i$ and the point x_ℓ. The value K_i cannot be recovered from the information that the simulator returns.

To show that the attack works in this ideal model, we just have to see that if two users have the same identificator, then $x_\ell \oplus K_i = x_{\ell'} \oplus K_j$ and therefore $x_\ell \oplus x_{\ell'} = K_i \oplus K_j$ which is the same information as in the real attack.

Now, we will prove that the simulator does not need to know $F_\Pi^{(i)}$ and can simulate the information by only using the public random function F_π and that the distribution of its outputs is indistinguishable from the idealized model. The simulator generates at random L random keys for the EM scheme. For each key, we will show that the pairs distinguished point/identificator can be generated only using F_π. Indeed, x_ℓ the ℓth iteration of $F_\Pi^{(i)}$ with key K_i from the value x_0 is the value $K_i \oplus x_\ell$ and this value is also the result of the iteration of the public function F_π from the value $x_0 \oplus K_i$. Consequently, to generate the pairs (distinguished point, identificator), the simulator can compute $(x_\ell \oplus K_i, id(x_\ell))$ without interacting with the users. As in this last case, the pairs are generated at random without interacting and knowing the function and since the function F_π are random, the edges in the graph are added at random and independently of each others and so that the graph is a random graph according to the Erdös-Rényi graph model.

Experimental results. We implement the previous attacks on an Even-Mansour cryptosystem using the DES with a fixed key and $n = 64$. We simulate 2^{22} users and for each user we create 8 chains (80 for the public user). We use distinguished points containing 21 zeroes and so the expected length is 2^{21} on average. We bound the length of the chains to 2^{24}, this means that if we remove the chain if we have not seen a distinguished point after 2^{24} evaluations. In all, we

have generated $33,543,077$ chains ($2^{25} = 33,554,432$, it misses the abandoned chains) and the number of groups containing at least two parallel chains is $4,109,961$. Experimentally, the size of the giant component contains $3,788,059$ users (among the $4,194,304$) and so we can deduce the keys of 90% of the users. This result is what is expected from theory since the number of vertices in this experiment is below the number of nodes. The 98% that is previously given as result in section 3.5, would require twice as many vertices.

The time to generate the chains is 1600 sec using 4096 cores in parallel and the analysis of the graph requires a few minutes on a standard PC.

4 Attacks on the PRINCE cipher in the Multi-User and Classical Setting

PRINCE is a lightweight block cipher published at ASIACRYPT 2012 [8]. It is based on the FX construction [18] which is actually an Even-Mansour like construction. PRINCE has been the interest of many cryptanalysts [10, 25, 16] who attack either the full cipher, or its reduced version.

The designers of PRINCE claim that its security is ensured up to 2^{127-n} operations when an adversary acquires 2^n plaintext/ciphertext pairs. This bound has been reduced in [16] to 2^{126} operations with a single plaintext/ciphertext pair. After a brief presentation of PRINCE, we describe a generic attack in the multi-user setting that allow to recover the key of a pair of users in a set of 2^{32} users with complexity 2^{64} computations. The identification of the pair of users uses the idea similar to the attack on Even-Mansour. However, details are different since PRINCE is not an Even-Mansour scheme as the internal permutation uses a secret key. Finally, we present another generic attack in the classical model that after a precomputation of 2^{96} time and 2^{64} in memory, allows to recover the key of every single user in time 2^{32}. Both attacks work for all rounds of PRINCE.

4.1 Brief description of PRINCE

PRINCE [8] uses a 64-bit block and a 128-bit key which is split into two equal parts of 64 bits, *i.e.* $k = k_0 \| k_1$. In order to extend the key to 192 bits it uses the mapping $k = (k_0 \| k_1) \rightarrow (k_0 \| k_0^{'} \| k_1)$ where $k_0^{'}$ is derived from k_0 by using a linear function L':

$$L'(k_0) = (k_0 \ggg 1) \oplus (k_0 \gg 63),$$

where $\gg$ denotes the right shift and $\ggg$ the rotation of a 64-bit word. While subkeys k_0 and $k_0^{'}$ are used as input and output whitening keys, the 64-bit key k_1 is used for the 12-round internal block cipher which is called PRINCE$_{core}$. For simplicity, we refer to it as the core of PRINCE or simply the core function and we denote it by $Pcore$. So every plaintext P is transformed into the corresponding ciphertext C by using the function $E_k(P) = k_0^{'} \oplus Pcore_{k_1}(P \oplus k_0)$ where $Pcore$ uses the key k_1 (see Fig.1).

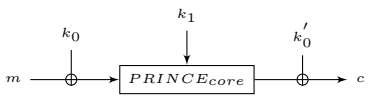

Fig. 1. Structure of PRINCE

The core function consists of a key k_1 addition, a round constant (RC_0) addition, five forward rounds, a middle round, five backward rounds and finally a round constant (RC_{11}) and a key k_1 addition. The full schedule of the core is shown in Fig. 2.

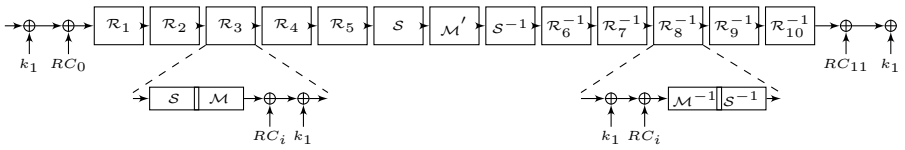

Fig. 2. Structure of the core of PRINCE

Each forward round of the core is composed by a 4-bit Sbox layer (S), a linear layer (64×64 matrix M), an addition of a round constant RC_i for $i \in \{1, \ldots, 5\}$ and the addition of the key k_1. The linear M layer is defined as $M = SR \circ M^{'}$ where SR is the following permutation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$\longrightarrow$

0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

The $M^{'}$ layer, which is only used in the middle rounds, can be seen as a mirror in the middle of the core as the 5 backward rounds are defined as the inverse of the 5 forward rounds.

In every RC_i-add step, a 64-bit round constant is XORed with the state. It should be noted that $RC_i \oplus RC_{11-i} = \alpha =$ `0xc0ac29b7c97c50dd` for all $0 \leq i \leq 11$. From this, but also from the fact that the matrix $M^{'}$ is an involution, we can perform the decryption function of PRINCE by simply performing the encryption procedure with inverse order of keys k_0 and $k_0^{'}$ and by using the key $k_1 \oplus \alpha$ instead of k_1. That means, that for any key $(k_0 \| k_0^{'} \| k_1)$, we have $D_{(k_0 \| k_0^{'} \| k_1)}(\cdot) = E_{(k_0^{'} \| k_0 \| k_1 \oplus \alpha)}(\cdot)$. This property is called the α-reflection property of PRINCE.

4.2 Attack on PRINCE in the multi-user setting

In the multi-user setting, we assume that we have L different users which are all using the block cipher PRINCE. Each user U_i with $0 \leq i < L$, chooses her key $k^{(i)} = k_0^{(i)} \| k_1^{(i)}$ at random and independently from all the other users. In order to attack PRINCE using the distinguished point method, we first construct a set of chains for every user using the function of PRINCE. For this, we use the function defined as follows:

$$F_{k_0^{(i)}, k_0^{'(i)}, k_1^{(i)}}(x) = x \oplus \ \text{PRINCE}_{k_0^{(i)}, k_0^{'(i)}, k_1^{(i)}}(x) \oplus \ \text{PRINCE}_{k_0^{(i)}, k_0^{'(i)}, k_1^{(i)}}(x \oplus \delta)$$

where δ is an arbitrary but fixed non zero constant. The key $k_0^{'(i)}$ vanishes from the equation and the function F thus takes the following form:

$$F_{k_1^{(i)}}(x) = x \oplus \ Pcore_{k_1^{(i)}}(x \oplus k_0^{(i)}) \oplus \ Pcore_{k_1^{(i)}}(x \oplus k_0^{(i)} \oplus \delta).$$

For every user U_i, we create one encryption ($\mathcal{E}$) chain and one decryption ($\mathcal{D}$) chain which are both based on the function F defined above. $\mathcal{E}$ uses the encryption function of PRINCE whereas $\mathcal{D}$ uses the decryption function. And so, for the user U_i, we define functions $\mathcal{E}$ and $\mathcal{D}$ as follows:

$$\mathcal{E}_{k_0^{(i)}, k_1^{(i)}}(x_j^{(i)}) = x_{j+1}^{(i)} = x_j^{(i)} \oplus Pcore_{k_1^{(i)}}(x_j^{(i)} \oplus k_0^{(i)}) \oplus Pcore_{k_1^{(i)}}(x_j^{(i)} \oplus k_0^{(i)} \oplus \delta)$$

$$\mathcal{D}_{k_0'^{(i)},k_1^{(i)}\oplus\alpha}(y_j^{(i)}) = y_{j+1}^{(i)}$$
$$= y_j^{(i)} \oplus Pcore_{k_1^{(i)}\oplus\alpha}(y_j^{(i)} \oplus k_0'^{(i)}) \oplus Pcore_{k_1^{(i)}\oplus\alpha}(y_j^{(i)} \oplus k_0'^{(i)} \oplus \delta).$$

Let us define:

$$f^{\mathcal{E}} = Pcore_{k_1^{(i)}}(x_j^{(i)} \oplus k_0^{(i)}) \oplus Pcore_{k_1^{(i)}}(x_j^{(i)} \oplus k_0^{(i)} \oplus \delta) \text{ and}$$

$$f^{\mathcal{D}} = Pcore_{k_1^{(i)}\oplus\alpha}(y_j^{(i)} \oplus k_0'^{(i)}) \oplus Pcore_{k_1^{(i)}\oplus\alpha}(y_j^{(i)} \oplus k_0'^{(i)} \oplus \delta).$$

We create encryption chains until $f^{\mathcal{E}}$ reaches a distinguished point (resp. decryption chains until $f^{\mathcal{D}}$ reaches a distinguished point). We search for a collision between the encryption and the decryption chain.

Let us consider two users, U_1 and U_2. Whenever the chains $\mathcal{E}_{k_0^{(1)},k_1^{(1)}}(x^{(1)})$ and $\mathcal{D}_{k_0^{(2)},k_1^{(2)}\oplus\alpha}(y^{(2)})$ arrive at the same distinguished point, we suspect that these two chains have become parallel. As the core of PRINCE is only parametrized by the key k_1, when we arrive at the same distinguished point we obtain a probable collision between keys $k_1^{(1)}$ and $k_1^{(2)} \oplus \alpha$ used in $Pcore$. However, we must verify that this is a real collision and not just a random incident. For this, we verify that next points of $f^{\mathcal{E}}$ and $f^{\mathcal{D}}$ after reaching a distinguished point, continue to remain equal. If we obtained a real collision we know that:

$$k_1^{(1)} = k_1^{(2)} \oplus \alpha.$$

This indicates that $x^{(1)} \oplus y^{(2)}$ is expected equal to $k_0^{(1)} \oplus k_0'^{(2)}$. It is obvious that since $k_1^{(1)} = k_1^{(2)} \oplus \alpha$ we will also have $k_1^{(1)} \oplus \alpha = k_1^{(2)}$. This indicates that we also know $k_0'^{(1)} \oplus k_0^{(2)}$.

Thus, we have:

$$k_0^{(1)} \oplus k_0'^{(2)} = A \text{ and } k_0'^{(1)} \oplus k_0^{(2)} = B \quad (*).$$

Let $\{a_{63}, \ldots, a_0\}$ be the representation of the bits of $k_0^{(1)}$ and $\{b_{63}, \ldots, b_0\}$ the representation of bits of $k_0^{(2)}$. As, from the definition of PRINCE, $k_0' = (k_0 \ggg 1) \oplus (k_0 \gg 63)$, we have that:

$$k_0'^{(1)} = \{a_0, a_{63}, \ldots, a_2, a_1 \oplus a_{63}\} \text{ and } k_0'^{(2)} = \{b_0, b_{63}, \ldots, b_2, b_1 \oplus b_{63}\}.$$

From $(*)$, we construct the system:

$$\{a_{63}, \ldots, a_0\} \oplus \{b_0, b_{63}, \ldots, b_2, b_1 \oplus b_{63}\} = \{A_{63}, \ldots, A_0\}$$
$$\{b_{63}, \ldots, b_0\} \oplus \{a_0, a_{63}, \ldots, a_2, a_1 \oplus a_{63}\} = \{B_{63}, \ldots, B_0\}$$

As this is an inversible linear system, we can easily find $k_0^{(1)}$ and $k_0^{(2)}$. Note that once $k_0^{(i)}$ has been found, recovering $k_1^{(i)}$ can be done with an exhaustive search whose cost is 2^{64}.

Analysis of the attack. Once the computation of a chain is finished we have to store $(x_{\ell-1}, d, d+1)$ where d is the distinguished point, $x_{\ell-1}$ is the point before the chain reaches a distinguished point and $d+1$ is the point after the chain reached a distinguished point. We need to store $x_{\ell-1}$ as we have to test if the found collision is useful and we also need to store $d+1$ to test if it is a real collision. If not, the search must continue.

As mentioned, PRINCE uses a 128-bit key which is split into two 64-bit parts, *i.e.* $k = k_0 \| k_1$. The attack consists in identifying and recovering all key material of a pair of users i and j for

whom $k_1^{(i)} = k_1^{(j)} \oplus \alpha$. We expect to find a collision $k_1^{(i)} = k_1^{(j)} \oplus \alpha$ between two different users with high probability when the number of users will be at least 2^{32}. So the attack uses a set of 2^{32} users and for each one we create 2 chains (encryption and decryption chain). The cost per user is 2^{32} operations and the total cost for recovering the keys k_0 of 2 users is approximately 2^{64} operations. For recovering k_1, the cost of the exhaustive search is 2^{64}. So in total, we can deduce both k_0 and k_1 in 2^{65} operations.

4.3 Attack in the classical model

We show in this section that a classical attack that also uses the distinguished points technique can also be possible. For this, we will create encryption chains from the function $\mathcal{E}$ defined in section 4.2.

Precalculation. In the first phase of the attack, we aim to create encryption chains for every possible key $k_1^{(i)}$ with $0 \leq i < 2^{64}$. More specifically, for every possible $k_1^{(i)}$, we set $k_0^{(i)} = 0$ and we create for every (i) a chain $\mathcal{S}_i$ from the function $\mathcal{E}$ with length 2^{32}. We store all chains $\mathcal{S}_i$.

Attack. Now, our purpose is to find a collision with one of the chains created with the zero key $k_0^{(i)}$. For this, for a random starting point x_0 and for keys k_0 and k_1 we will calculate an encryption chain $\mathcal{T}$ from the function $\mathcal{E}$. The chain $\mathcal{T}$ will collide with high probability with one of the chains $\mathcal{S}_i$. As described in previous section 4.2, when we detect a collision between two distinguished points, we know that the chains had become parallel and so we obtain $k_0^{(i)} \oplus k_0$. As the key $k_0^{(i)} = 0$, we finally obtain the unknown k_0.

Analysis of the attack. For the precalculation phase, for every 2^{64} possible keys we calculate a chain with length 2^{32} and so our complexity is equal to 2^{96}. As we need to store all chains, the precalculation phase has also a cost of 2^{64} in memory. However, once the first phase is over, the attacker can perform the attack in only 2^{32} operations as she has to calculate only one chain. So, the total cost of the attack is 2^{96}. The proposed attack satisfies $DT = 2^{128}$ as $D = 2^{32}$ and $T = 2^{96}$. This attack does not improve the complexity of PRINCE given in [8] and [16]. However, in our case, T is not the on-line time complexity as it corresponds to the precalculation phase of the attack. Thus, in our attack, we have $DT_{on} = 2^{64}$.

5 Conclusion

In this paper, we have presented new tradeoffs for public-key and symmetric-key cryptosystems in the multi-user setting. We have introduced some algorithmic tools for collision-based attacks using the distinguished point technique. The first tool allows to look for the discrete logarithm of L users in parallel using only a $\widetilde{O}(\sqrt{L})$ penalty using random graph process behaviour. The second tool allows to achieve key-recovery of Even-Mansour and related ciphers and is a novel lambda technique to find collisions when two different functions are involved. For the Even-Mansour cipher, we show new tradeoffs that partially solve an open problem due to Dunkelman *et al.* and we propose an analysis in the multi-user setting. Finally, for the PRINCE cipher, we show generic attacks that improve the best published results in the sense that our time complexity corresponds to a precomputation phase and not to an on-line phase. This last result could also be adapted to similar ciphers such as DESX and would also improve on the best previous attack.

References

1. Daniel J. Bernstein and Tanja Lange. Computing Small Discrete Logarithms Faster. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 317–338, 2012.
2. Daniel J. Bernstein and Tanja Lange. Non-uniform Cracks in the Concrete: The Power of Free Precomputation. In *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 II*, pages 321–340, 2013.
3. Eli Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.
4. Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved Time-Memory Trade-Offs with Multiple Data. In *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 110–127, 2005.
5. Alex Biryukov and Adi Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.
6. Alex Biryukov and David Wagner. Advanced Slide Attacks. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer, 2000.
7. Béla Bollobás. *Random Graphs*. Cambridge studies in advanced mathematics, 2nd edition, 2001.
8. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In *ASIACRYPT*, pages 208–225, 2012.
9. Anne Canteaut, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard. Multiple Differential Cryptanalysis of Round-Reduced PRINCE (Full version). *IACR Cryptology ePrint Archive*, 2014:89, 2014.
10. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-Middle: Improved MITM Attacks. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 222–240, 2013.
11. Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another Look at Tightness. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 293–319, 2011.
12. Kevin P Costello and Van H Vu. The rank of random graphs. *Random Structures & Algorithms*, 33(3):269–285, 2008.
13. Joan Daemen. Limitations of the Even-Mansour Construction. In *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, pages 495–498, 1991.
14. Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 336–354, 2012.
15. Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In Hideki Imai, RonaldL. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer Berlin Heidelberg, 1993.
16. Jérémy Jean, Ivica Nikolic, Thomas Peyrin, Lei Wang, and Shuang Wu. Security Analysis of PRINCE. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 92–111, 2013.
17. Antoine Joux. *Algorithmic Cryptanalysis*. Chapman & Hall / CRC Cryptography and Network Security Series, 2009.
18. Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). *J. Cryptology*, 14(1):17–35, 2001.

19. Fabian Kuhn and René Struik. Random Walks Revisited: Extensions of Pollard's Rho Algorithm for Computing Multiple Discrete Logarithms. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 212–229. Springer, 2001.
20. Rodolphe Lampe, Jacques Patarin, and Yannick Seurin. An Asymptotically Tight Security Analysis of the Iterated Even-Mansour Cipher. In *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*, pages 278–295, 2012.
21. Hyung Tae Lee, Jung Hee Cheon, and Jin Hong. Accelerating ID-based Encryption based on Trapdoor DL using Pre-computation. Cryptology ePrint Archive, Report 2011/187, 2011. `http://eprint.iacr.org/`.
22. Alfred Menezes. Another Look at Provable Security. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, page 8, 2012.
23. John M. Pollard. Kangaroos, Monopoly and Discrete Logarithms. *J. Cryptology*, 13(4):437–447, 2000.
24. Jean-Jacques Quisquater and Jean-Paul Delescaille. How Easy is Collision Search. New Results and Applications to DES. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 408–413. Springer, 1989.
25. Hadi Soleimany, Céline Blondeau, Xiaoli Yu, Wenling Wu, Kaisa Nyberg, Huiling Zhang, Lei Zhang, and Yanfeng Wang. Reflection Cryptanalysis of PRINCE-Like Ciphers. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 71–91, 2013.
26. Paul C. van Oorschot and Michael J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *J. Cryptology*, 12(1):1–28, 1999.

A Probabilistic Results

A.1 Random Graph Process

In this subsection, we recall the results on random graph processes that can be found in [7].

In the Erdős-Rényi graph model $\mathcal{G}\{n, Pr(edge) = p\}$ $(0 < p < 1)$, all graphs have vertex set $V = \{1, 2, \ldots, n\}$ in which edges are chosen independently and with probability p. In other words, if G_0 is a graph with vertex set V and it has m edges, then

$$Pr(\{G_0\}) = Pr(G = G_0) = p^m q^{N-m},$$

where $q = 1 - p$ and $N = \binom{n}{2}$.

A random graph process on $V = \{1, 2, \ldots, n\}$ is a Markov chain $\tilde{G} = (G_t)_0^\infty$, whose states are graphs on V. The process starts from the empty graph and for $1 \leq t \leq N$ the graph G_t is obtained from G_{t-1} by adding an edge, all new edges begin equiprobable (we do not pick an edge that has been already chosen).

First of all, we give the result on the giant component

Theorem 1. *Let $c > 1$ be a constant, $t = \lfloor cn/2 \rfloor$ and $\omega(n) \to \infty$. Then, almost every G_t is the union of the giant component, the small unicyclic components and the small tree components. There are at most $\omega(n)$ vertices on the unicyclic components. The order of the giant component $L_1(G_t)$ satisfies*

$$|L_1(G_t) - (1 - t(c))n| \leq \omega(n)n^{1/2},$$

where

$$t(c) = \frac{1}{c} \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!} (ce^{-c})^k,$$

and for every fixed $i \geq 2$

$$\left| L_i(G_t) - (1/\alpha)\Big(\log n - \frac{5}{2} \log \log n \Big) \right| \leq \omega(n),$$

where $\alpha = c - 1 - \log c$.

If $t = O(n \log n)$, then the graph is almost certainly connected.

Theorem 2. *For $t = \lfloor 2n \ln n \rfloor$, we have*

$$\Pr[G_t \text{ is not connected}] < n^{-n/4}.$$

We can find the expected value of t that makes the graph connected, and then apply tail bounds to compute the probability for this happening for a particular t in the random graph process. First notice that $t \geq n - 1$ for a connected graph (a tree is a minimally-connected graph). As we add edges, we watch the number of connected components of the graph. Initially, the graph has n vertices and no edges, so there are n connected components. The first edge always connects two points, and gives us $n - 1$ connected components. The second edge also reduces the number of connected components to $n - 2$. The third may or may not reduce the number. We use epochs to model the different phases of the process (at each epoch, the number of connected components is the same). Let X_k be the number of random edges added while there are k connected components, until there are $k-1$ connected components. We have shown that $X_n = 1$ and $X_{n-1} = 1$. If we define

$$X = \sum_{k=2}^{n} X_k,$$

then X counts the total number of edges that we add until the graph is connected. Our goal is to compute $E(X)$. Now, define p_k to be the probability that an edge added while there are k components reduces the number of components. We cannot compute p_k exactly, but we can provide a lower bound. Assume that v is one edpoint of the edge we are adding. Then, there are at least $k-1$ other vertices to which we can connect v and reduce the number of components (these other vertices lie on the other components). In total there are $n-1$ other vertices to which we can connect v. So the probability that this edge reduces the number of components is $\geq (k-1)/(n-1)$. But this bound holds for any choice of v, so it also bounds p_k:

$$p_k \geq \frac{k-1}{n-1}.$$

Now, observe that X_k is a geometric random variable with success probability p_k. Its expected value is $1/p_k \leq (n-1)/(k-1)$. So we have

$$E(X) = \sum_{k=2}^{n} E(X_k) \leq \sum_{k=2}^{n} \frac{n-1}{k-1} = (n-1)H_{n-1},$$

where H_{n-1} is the $(n-1)^{st}$ harmonic number. In other words, an upper bound on $E(X)$ is about $n \ln n$.

The final step is to apply tail bounds on the probability of t being much larger than its mean using Chebyshev or Chernoff bounds.

Using Chernoff bound, we can show that for $\delta < 2e-1$

$$\Pr[X > (1+\delta)E(X)] < exp(-E(X)\delta^2/4),$$

which in the special case of $\delta = 1$ leads to

$$\Pr[X > 2n \ln n] < n^{-n/4}.$$

A.2 Birthday Paradox with Girls and Boys

Theorem 3. *Let two sequences of size n_1 (resp. n_2), uniformly chosen at random and independently in the set $\{1, 2, \ldots, n\}$, then the expected number of collision is $n_1.n_2/n$.*

B A survey of Existing Attacks on Even-Mansour

In this section, we recall previous attacks on the Even-Mansour cryptosystem and the open problem of Dunkelman, Keller and Shamir as well as recast the attacks on Even-Mansour on collision-based algorithms.

Daemen's known-plaintext attack. The first attack uses two known plaintext/ciphertext pairs (M_0, C_0) and (M_1, C_1). The adversary performs an exhaustive search on the value K_1 and tests whether $\pi(M_0 \oplus K) \oplus \pi(M_1 \oplus K)$ is equal to $C_0 \oplus C_1$. If this is the case then, $K = K_1$ and since the block length is equal to the key size we expect to have a constant number of candidates. On average, this attack has a time complexity of $T = N/2$ where $n = |K_1| = |K_2|$ and requires two chosen plaintexts, $D = 2$.

Daemen's chosen-plaintext attack. The second attack is a T/M tradeoff of the previous attack. The adversary asks the encryption of D chosen plaintexts pairs (M_i, M_i') such that $M_i \oplus M_i' = \delta$ a fixed value and receives the corresponding ciphertext pair (C_0^i, C_1^i). Then, he store in a hash table the value M_i at the index $C_0^i \oplus C_1^i$. Finally, he computes for T values V the value $\Delta W = \pi(V) \oplus \pi(V \oplus \delta)$ and check whether this value is the index of a value in the hash table. According to the birthday paradox, on average we expect to have one collision with high probability as soon as $DT = O(N)$ between the values ΔW and the values $C_0^i \oplus C_1^i$. Such a collision gives us a solution for the key K_1 as $K_1 = M_i \oplus V$ or $K_1 = M_i \oplus V \oplus \delta$. The memory complexity of this attack is $\min(T, D)$. But, in practice we usually have $D \leq T$ since memory is a more scarce resource than time.

Biryukov-Wagner Slide attack. About ten years later, Biryukov and Wagner discovered the slide attack to break an arbitrarily number of rounds. Their technique can be adapted to the Even-Mansour cipher using a twist in the classical attack. Their attack has a complexity in $D = N^{1/2}$ known plaintexts and $T = N^{1/2}$ called to π. A slid pair is a pair of messages (P, P') such that

$$P \oplus P' = K_1.$$

Then, it is easy to see that we have the following condition for the slid pair:

$$\Pi(P) \oplus \pi(P) = \Pi(P') \oplus \pi(P'). \tag{1}$$

The idea of the attack is that in a set of $D = N^{1/2}$ plaintexts, we can construct N pairs and there is at least one slid pair on average among them. Consequently, we can check the condition (1) by searching a collision between the values $\Pi(P) \oplus \pi(P)$. A collision between the pair (P_i, P_j) will give us the two key candidates $K_1 = P_i \oplus P_j$ and $K_2 = \Pi(P_i) \oplus \Pi(P_j)$. The time complexity of this attack is $T = N^{1/2}$ and the memory requirement is $M = N^{1/2}$.

Slidex attack on the Two-Key Even-Mansour. The attack for the two-key EM scheme is a generalization of the advanced sliding attack of [6] where Dunkelman *et al.* introduce an additional degree of freedom Δ and is called the Slidex attack. In this attack, we assume that we have a slid pair which satisfies the property

$$P \oplus P' = K_1 \oplus \Delta$$

for some $\Delta \in \{0,1\}^n$. For this plaintext pair, we have

$$\Pi(P) = \pi(P \oplus K_1) \oplus K_2 = \pi(P' \oplus \Delta) \oplus K_2$$

$$\Pi(P') = \pi(P' \oplus K_1) \oplus K_2 = \pi(P \oplus \Delta) \oplus K_2$$

and so the following slid pair condition is satisfied

$$\Pi(P) \oplus \pi(P \oplus \Delta) = \Pi(P') \oplus \pi(P' \oplus \Delta).$$

This allows to mount an attack for any value $D \leq N$. First of all, the adversary calls $D = 2^{d/2}$ times the EM scheme to get the known plaintext encryptions $P_i, \Pi(P_i)$. Then, for each of 2^{n-d} arbitrary values for Δ he stores the value i in a hash table indexed by $\Pi(P_i) \oplus \pi(P_i \oplus \Delta)$ and searches for a slid pair in the hash table by checking the slid pair condition. Then each collision between (P_i, P_j) with Δ_k will give key candidates $K_1 = P_i \oplus P_j \oplus \Delta_k$ and $K_2 = \Pi(P_i) \oplus \pi(P_j \oplus \Delta_k)$. This is a known-plaintext attack which works for any value $D < N^{1/2}$. We will see later that one problem with this attack is that it is not possible to use precomputation.

C A reminder of distinguished points methods

Given a function f on a set S of size N, distinguished points methods allow to find collisions in a very flexible way. We first define a distinguished subset S_0 in S, using any efficiently testable property. The basic idea is, starting from a random point x_0 in S to construct chains of computations by evaluating the sequence $x_{i+1} = f(x_i)$ until we encounter a distinguished point, i.e. an element of S_0. The average length of such chains is $\ell = |S|/|S_0|$. Note that to avoid degenerate cases, it is useful to abort the computation of chains which do not reach a distinguished point after $c_0 \cdot \ell$ steps, where $c_0 > 1$ is a fixed constant. Indeed, the proportion of points which satisfy the distinguishing property is $1/\ell$. Then, the length of the chains is ℓ on average. If we compute chains of length $c_0 \cdot \ell$, then the proportion of chains that exceeds this value is $(1 - 1/\ell)^{c_0 \ell} \approx \exp(-c_0)$. Values such as $c_0 = 20$ are often recommended [26]. Since the number of aborted chains is 20 times larger than the average, the proportion of aborted work is approximately $20e^{-20} < 5 \cdot 10^{-8}$.

Once the computation of a chain is finished, we store a summary of the chain $(x_0, \ell_{x_0}, d_{x_0})$ containing the starting point x_0, the number of iterations ℓ_{x_0} and the distinguished endpoint d_{x_0}. The most important property of chains is that two chains which pass through a common point necessarily end at the same distinguished point. The converse is almost true, two chains that end at the same distinguished point are merging at some point (and thus yield a collision) unless one of the two chains is a subchain of the other.

When building chains, an essential safety measure is to avoid computing many times the same thing. For example, it is useless to aim at computing chains of length larger than $N^{1/2}$, because we expect such chains to cycle. Even if we make chains shorter, this can become a problem. Typically, when constructing B different chains of average length ℓ, we should always ensure that $B \cdot \ell^2$ do not grow beyond N [17]. Otherwise, we expect too many early mergings of the final chains into their predecessors. Collisions between 2-sets can be analyzed using Theorem 3 in Appendix.

D A reminder of Hellman's Time/Memory tradeoff

Hellman's algorithm is a method to invert a function $f : S \to S$ in time $T = N^{2/3}$ and memory $M = N^{2/3}$ with precomputation time $P = N$ where $N = |S|$. The idea is to use m starting points and iterate f t times for each starting point to compute the ending points. We store the m starting and ending points in a hash table indexed by the ending points. A table represents a covering of mt points among the N values. If we try to invert a value y covered in the table, we iterate f from y until $f^{(i)}(y)$ is in the set of ending points. Now, from the corresponding starting point, we iterate f until we reach y and the previous value gives the preimage. The on-line time complexity is t if we test membership in the set of ending points using a hash table. The memory is reduced to m if we omit the log factors.

The matrix stopping rule is $t \cdot mt \leq N$ since if we add a new line in the table, we will get a collision with one point of the m previous lines with probability $t \cdot (mt)/N$ according to Appendix A.2. Moreover, since colliding points will not add points to the coverage, a table cannot contain more than $mt^2 \leq N$ points. However, a table covers only mt points among the N values and then, from the matrix stopping rule, we cover at most a fraction $1/t$ of the N points. To increase this value, Hellman's idea is to use t independent tables with a related function f_i. Now, the memory is mt and the on-line time complexity is mt since we need to iterate m different functions and test membership. If we fix $t = m = N^{1/3}$, then we obtain the announced complexity and the precomputation needs to cover the whole space N.