How to Construct an Ideal Cipher from a Small Set of Public Permutations

Rodolphe Lampe[1] and Yannick Seurin[2]

[1] University of Versailles, France
[2] ANSSI, Paris, France
rodolphe.lampe@gmail.com,yannick.seurin@m4x.org

April 20, 2013

Abstract. We show how to construct an ideal cipher with n-bit blocks and n-bit keys (*i.e.* a set of 2^n public n-bit permutations) from a small constant number of n-bit random public permutations. The construction that we consider is the *single-key iterated Even-Mansour cipher*, which encrypts a plaintext $x \in \{0,1\}^n$ under a key $k \in \{0,1\}^n$ by alternatively xoring the key k and applying independent random public n-bit permutations $P_1, \ldots, P_r$ (this construction is also named a *key-alternating cipher*). We analyze this construction in the plain indifferentiability framework of Maurer, Renner, and Holenstein (TCC 2004), and show that twelve rounds are sufficient to achieve indifferentiability from an ideal cipher. We also show that four rounds are necessary by exhibiting attacks for three rounds or less.

Keywords: block cipher, ideal cipher, iterated Even-Mansour cipher, key-alternating cipher, indifferentiability.

Table of Contents

1 Introduction

1.1 Block Ciphers

Block ciphers are one of the most important classes of primitives in cryptography. They are mainly used to provide confidentiality and authenticity to communication channels or local data storage means, but also to construct hash functions and in other more advanced cryptographic tasks. Syntactically, a block cipher E with message space $\{0,1\}^n$ and key space $\{0,1\}^m$ is a mapping from $\{0,1\}^m \times \{0,1\}^n$ to $\{0,1\}^n$ such that for each key $k \in \{0,1\}^m$, $E(k,\cdot)$ is an (efficiently invertible) permutation. Block cipher designs (virtually all of which rely on the iteration of some key-dependent round function) can be roughly split into two families (with some rare exceptions such as IDEA):

1) Feistel networks [Fei73] and their generalizations, where the round function is given by $(x,y) \mapsto (y, x \oplus F(k_i, y))$, where x and y are the left and right $n/2$ bits of the state, and k_i is the round key; prominent examples include DES, Blowfish, KASUMI, and Camellia for "classical" Feistel networks, and CAST-256 and MARS for generalized Feistel networks;
2) substitution-permutation networks (SPNs), where one round generally consists of the composition of a round-key addition, a non-linear mixing layer, and a linear diffusion layer; notable examples include AES, SAFER, CRYPTON, SERPENT, PRESENT, and LED.

At an even higher design level, SPNs can be described (by collapsing the non-linear mixing layer and the linear diffusion layer at i-th round into a single n-bit permutation P_i) as successive applications of round-key additions and permutations P_i. Such a structure was named a *key-alternating cipher* by the designers of AES [DR01, DR02].

The traditional security notion for a block cipher is pseudorandomness, *i.e.* indistinguishability from a random permutation [LR86]: namely, no distinguisher with reasonable resources and having black-box access to a permutation (and also to its inverse in a more stringent variant of the security notion) should be able to distinguish whether it is interacting with the block cipher $E(k,\cdot)$ for a randomly chosen key k, or with a truly random permutation. In an asymptotic and more theoretical language, a family of block ciphers indexed by a security parameter meeting this security notion is called a pseudorandom permutation (PRP), or a strong pseudorandom permutation (SPRP) when the distinguisher has also access to the inverse permutation. The classical example of a construction for which we have some provable security results with respect to indistinguishability is the Feistel network. Starting from the seminal Luby-Rackoff paper [LR88] which showed that the Feistel construction with three rounds yields a PRP when its round functions are pseudorandom [GGM86], and followed by a paper by Patarin [Pat90] showing that four rounds yield a SPRP (which was stated in [LR88] without proof), a long series of works established refined results in the same vein, such as [Mau92, MP03, Vau03, Pat04] to name a few.

1.2 The Ideal Cipher Model

Though there are numerous examples where the standard pseudorandomness assumption is sufficient to prove (in a reductionist sense) the security of a cryptographic scheme (*e.g.* for building a symmetric encryption scheme [BDJR97] or a MAC scheme [BKR00]), there are also some settings where it might not be strong enough to derive a security proof. Indeed, in some situations, the adversary has more abilities than merely querying in a black-box

way an encryption/decryption oracle. For example, there are some cases where the attacker might have access to a more powerful "related-key" oracle [Bih94, BK03, AFPW11], *i.e.* it can ask encryption and decryption queries for keys that are related (in some limited and attack-dependent way) to the main key of the system.

In some contexts, the adversary might even be able to know or choose the key during the attack. This is notably the case when considering hash functions built from a block cipher, which is an old and well-established design strategy [LM92, PGV93]. Consider for example the Davies-Meyer construction, which turns a n-bit block and m-bit key block cipher E into a $(m+n)$-to-n-bit compression function $f(k\|x) = E(k,x) \oplus x$. An attacker trying to break the collision or pre-image resistance of this compression function is clearly in a more powerful position than a mere black-box distinguisher. In fact, the PRP assumption is provably not sufficient to show the security of the Davies-Meyer construction. More precisely, assuming that PRPs exist, then there exists a PRP such that the Davies-Meyer compression function obtained from it is insecure. For example, starting from a block cipher E (with key length equal to its block length) which is a PRP, one can modify it to a slightly different block cipher $\tilde{E}$ such that $\tilde{E}(k,k) = k$ for any key k. This does not affect the PRP property, but this clearly makes the Davies-Meyer compression function based on $\tilde{E}$ trivially vulnerable to pre-image and collision attacks. Even more disappointingly, Simon [Sim98] showed that there is no provably secure black-box construction of a collision resistant hash function family from a block cipher using only the standard PRP assumption.

Ideally, the ultimate security goal for a block cipher would be that it "behaves" as a random and independent permutation for each possible key. This naturally leads to the so-called *ideal cipher model* (ICM), the origin of which can be traced back to Shannon [Sha49]. In the ICM, a block cipher E with n-bit blocks and m-bit keys is drawn at random from the set of $(2^n!)^{2^m}$ possible block ciphers of this form, and made available through oracle queries (for both encryption and decryption) to all parties (including the adversary). This is very similar in spirit to the random oracle model (ROM) [FS86, BR93] used to model a perfect hash function. To the best of our knowledge, this model was first formally used in a security proof by Winternitz [Win84] and later by Merkle [Mer89] to show respectively the pre-image and collision resistance of the Davies-Meyer compression function. The ICM became increasingly popular after Black *et al.* [BRS02] used it to extensively analyze the security of the PGV block cipher-based compression functions [PGV93]. Since then, the ICM has been used to prove the security of a variety of other block cipher-based hash functions [Hir04, Hir06, Ste07, LSS11, Men12], of key length extension methods for block ciphers [KR96, Des00, BR06, GM09, GT12], of symmetric encryption schemes [JJV02], and even of some public-key protocols such as signature schemes [Gra02], ring signature schemes [RST01], public-key encryption [Jon02], and key exchange protocols [BPR00]. Despite these numerous successful applications, one must not lose from sight that the ICM only gives heuristic insurance just as the ROM [CGH98]. In particular, Black [Bla06] exhibited an (arguably artificial) block cipher-based hash function which is provably collision resistant in the ICM, but becomes insecure when the ideal cipher is instantiated with any concrete block cipher.

With the ICM at hand, the question now becomes: is it possible to argue that a given block cipher design is as close as possible to an ideal cipher? In the standard model, one immediately faces the problem that, unlike for pseudorandomness, it even seems hard to come with a satisfactory definition of what this formally means, without running into impossibility results (similarly to [CGH98] and [Bla06]) following from the fact that a concrete block cipher

has a short description, whereas an ideal cipher does not. This unfortunate state of affairs has not prevented cryptanalysts from *disproving* that a concrete block cipher behaves as an ideal cipher by exhibiting some non-random behavior, *i.e.* some non-trivial[3] relation between inputs and outputs of the block cipher that can be found faster than for an ideal cipher, in a setting where the key is random and given to the attacker (known-key attacks), or when the attacker can freely choose the key(s) (chosen-key attacks). A classical example is the complementation property of DES which, despite being often viewed as a "benign" undesirable property, implies that DES does not behave as an ideal cipher. For AES, no such non-random properties were known until Biryukov *et al.* [BKN09] showed that so-called q-multicollisions can be found faster for AES-256 than for an ideal cipher. Known-key and chosen-key attacks were first put forward as an important cryptanalysis goal by Knudsen ans Rijmen [KR07], and have since then become an active area of research [MPP09, GP10, SY11].

1.3 Indifferentiability

Though we cannot hope to formalize (not to say prove) that a concrete block cipher behaves as an ideal cipher in any reasonable sense in the standard model, it is possible to obtain positive results in idealized models, *i.e.* by viewing some subcomponent of the block cipher as perfectly random. This perfect subcomponent is made available to all parties as a public oracle, which makes this setting formally distinct from classical indistinguishability. In order to assess whether a cryptographic construction based on an ideal subcomponent is secure, one has to employ the formalism of *indifferentiability*, introduced by Maurer *et al.* [MRH04]. A construction $\mathcal{C}$ (*e.g.* a block cipher), based on some ideal primitive $\boldsymbol{F}$ (*e.g.* a random permutation), is said to be indifferentiable from some target ideal primitive $\boldsymbol{G}$ (*e.g.* an ideal cipher) if there exists an efficient simulator $\mathcal{S}$ (with black-box access to the primitive $\boldsymbol{G}$) such that the two systems $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$ and $(\boldsymbol{G}, \mathcal{S}^{\boldsymbol{G}})$ are indistinguishable. Informally, the goal of the simulator is to provide answers which are consistent with what a distinguisher can obtain from $\boldsymbol{G}$, without deviating too much from the distribution of answers of $\boldsymbol{F}$. An indifferentiability result can be interpreted as a way to make sure that the high-level design of the construction $\mathcal{C}$ has no structural defect. More importantly, a composition theorem [MRH04] asserts that if $\mathcal{C}^{\boldsymbol{F}}$ is indifferentiable from $\boldsymbol{G}$, then any cryptosystem proved secure when used with $\boldsymbol{G}$ remains secure when used with $\mathcal{C}^{\boldsymbol{F}}$, therefore allowing modular proofs of security in idealized models.[4]

Soon after its introduction, Coron *et al.* [CDMP05] used the indifferentiability framework to revisit the design of a hash function from an ideal cipher: namely they showed that a number of variants of the Merkle-Damgård domain extension method [Dam89, Mer89], used with an ideal cipher in Davies-Meyer mode, are indifferentiable from a random oracle. The converse direction, *i.e.* proving that it is possible to construct an ideal cipher from a random oracle, turned out to be harder to achieve. A first attempt to prove that the Feistel construction with public random round functions is indifferentiable from a random permutation (and hence from an ideal cipher by prepending the key to each input to the random round functions) was made by Coron *et al.* for six rounds [CPS08], and later by Seurin for ten rounds [Seu09], but serious flaws were found in both proofs [Kün09, HKT11]. The situation was corrected with a proof by

[3] We stress that because of the lack of a rigorous definition, the meaning of non-trivial here is somehow subjective.

[4] Care has to be taken with this composition result when the security definition for the cryptosystem puts some limitations on the adversary, such as an upper bound on its memory [RSS11, DGHM12]

Holenstein *et al.* [HKT11] that the 14-round Feistel construction with public random round functions is indifferentiable from a random permutation. This must be contrasted with the classical Luby-Rackoff result stating that the 4-round Feistel construction with pseudorandom round functions yield a SPRP.

1.4 Our Contribution

The indifferentiability result for the Feistel construction mentioned above is fundamentally about how to obtain a random permutation from a random (function) oracle. The step to obtain an ideal cipher (*i.e.* an exponential number of independent permutations) is trivially achieved through domain separation of the underlying primitive (namely by prepending the key to each call to the random function oracles). However, it does not tell us anything about how the key should be concretely mixed into the state. In a departure from this approach, we ask the following question: given a small number of objects with n-bit inputs (*e.g.* n-bit permutations $P_1, \ldots, P_r$), is there a way to "combine" them together with an m-bit key in order to obtain a construction which is close to an n-bit block and m-bit key ideal cipher, *i.e.* a set of 2^m independent permutations, without appealing to a trivial domain separation argument? This naturally prompts us to turn our attention towards the second class of designs, namely key-alternating ciphers.[5] More formally, we consider the construction of a block cipher with n-bit blocks and m-bit keys from r public n-bit permutations $P_1, \ldots, P_r$ defined as follows: derive $(r+1)$ n-bit round keys $(k_0, \ldots, k_r)$ from a master key K through some key derivation function, and encrypt the plaintext $x \in \{0,1\}^n$ by computing the ciphertext y defined as:

$$y = k_r \oplus P_r(k_{r-1} \oplus P_{r-1}(\cdots P_2(k_1 \oplus P_1(k_0 \oplus x))\cdots)) \ .$$

When $r = 1$ and two independent n-bit keys (k_0, k_1) are used, so that the ciphertext is simply $y = k_1 \oplus P_1(k_0 \oplus x)$, one obtains the so-called Even-Mansour cipher [EM97]. When P_1 is modeled as a public random permutation (that the adversary can query in a black-box way), Even and Mansour [EM97] showed that the resulting block cipher is a SPRP, with security ensured up to $\mathcal{O}(2^{n/2})$ distinguisher queries. The indistinguishability of the general construction for $r > 1$ with independent keys $(k_0, \ldots, k_r)$ was later studied for two rounds by Bogdanov *et al.* [BKL+12], for three rounds by Steinberger [Ste12], and for any number r of rounds (with non-tight security bounds) by Lampe *et al.* [LPS12]. Unsurprisingly, the number of adversarial queries up to which the key-alternating cipher is indistinguishable from a random permutation increases with the number of rounds. Following [LPS12], and to emphasize that we work in the random permutation model for $P_1, \ldots, P_r$, we will use the naming *r-round iterated Even-Mansour cipher* to designate the idealized key-alternating cipher where the permutations $P_1, \ldots, P_r$ are public and perfectly random permutations oracles.

In this paper, we consider the iterated Even-Mansour cipher from the point of view of indifferentiability, and ask whether this construction is indifferentiable from an ideal cipher for a sufficient number of rounds when the permutations $P_1, \ldots, P_r$ are public and random. A first simple observation is that the construction with $r+1$ independent n-bit keys $(k_0, \ldots, k_r)$

[5] One could certainly undertake the same study for Feistel-based block ciphers, but this seems more complicated.

(resulting in a total key space $\{0,1\}^m = \{0,1\}^{(r+1)n}$) is never indifferentiable (for any r) from an ideal cipher with n-bit blocks and $(r+1)n$-bit keys (this had already been informally observed by [BKL+12]). In a sense, independent keys offer too much freedom to the attacker, enabling to easily find related-key relations. There are two possible approaches to solve this problem. The first one is to derive the round keys $(k_0, \ldots, k_r)$ from the master key using some cryptographic function (modeled as a random oracle for the indifferentiability proof). This was considered in an earlier and independent work by Andreeva *et al.* [ABD+13] (see Section 1.6 below for a discussion of their result). The second possibility (not relying on any cryptographic assumption about the key derivation function) is to "correlate" the round keys. This is the approach we adopt: namely, we consider the iterated Even-Mansour cipher where the n-bit round keys $(k_0, \ldots, k_r)$ are obtained by applying efficiently invertible n-bit permutations $(\gamma_0, \ldots \gamma_r)$ to the n-bit master key k (see Figure 1 on page 12). As will appear clearly in view of its proof, the fact that the master key length is equal to the block length is crucial for our result. To insist on this particular point, we call this construction the *single-key* iterated Even-Mansour cipher. Our main result is the following one.

Theorem. *The 12-round single-key iterated Even-Mansour cipher with twelve independent random public n-bit permutations $(P_1, \ldots, P_{12})$ and any efficiently invertible (public) n-bit permutations $(\gamma_0, \ldots, \gamma_{12})$ for the key schedule is indifferentiable from an ideal cipher with n-bit blocks and n-bit keys.*

In fact, the key derivation permutations γ_i will not play any role in the proof, so that we will focus on the simple case where they are all equal to the identity. Additionally, we show that at least four rounds are necessary by describing attacks (using only a constant number of queries) for three rounds or less.

Together with the result of [ABD+13] discussed below, our main theorem validates the design strategy underlying SPNs and more generally key-alternating ciphers as a sound way to ensure security beyond pseudorandomness: it (theoretically) enables to achieve resistance against related-key, known-key and chosen-key attacks (that an ideal cipher can withstand). We stress that our result cannot be used as is to take *concrete* design decisions: first, our bounds (as is often the case with indifferentiability results) are extremely loose.[6] More importantly, the permutations P_i used in concrete block ciphers such as AES are often too simple to be deemed close to random permutations (not to say independent: they are often the same).

1.5 Our Techniques

The techniques used to prove our main theorem are very similar to the ones introduced in [CPS08, Seu09, HKT11] for the Feistel construction (while the formalism we adopt is very close to [HKT11]). We simply give a very cursory overview of the main ideas here (assuming all γ_i's are the identity). The simulator works by detecting and completing "partial chains" created by the queries of the distinguisher. Define the computation path for a plaintext x and a key k as the sequence of pairs (x_1, y_1), $\ldots$, (x_{12}, y_{12}) of corresponding input and output values for the simulated permutations $P_1, \ldots, P_{12}$. It must hold that the value y obtained through this computation path matches the value $\boldsymbol{E}(k, x)$ obtained from the ideal cipher,

[6] Since the proof is already quite involved, we favored simplicity rather than tightness, but the bounds can probably be improved at some places.

otherwise one could straightforwardly distinguish the "simulated" world from the "real" world. Hence, simply answering the distinguisher queries randomly will not work: the simulator must somehow "adapt" the computation path to match the ideal cipher $\boldsymbol{E}$. Observe now the following important property of the single-key iterated Even-Mansour cipher: given only two consecutive values y_i and x_{i+1} of the computation path (*i.e.* the output value of permutation P_i and the input value to permutation P_{i+1}), it is possible to deduce the corresponding key $k = y_i \oplus x_{i+1}$, and hence to move forward and backward along the path. Note that this property essentially relies on the fact that the master key length is equal to the block length of the permutations (would the master key be larger, then it could not be uniquely determined by y_i and x_{i+1}). Note also that this is the exact analogue of the property of the Feistel network that the input and output values to two consecutive round functions enable to uniquely move forward and backward inside the construction. With this in mind, the strategy of the simulator will be to detect *partial chains* in computation paths created by queries of the distinguisher to two consecutive permutations, and "complete" them by moving forward and backward inside the iterated Even-Mansour construction (randomly setting undefined permutation values encountered along the way, and making a call to the ideal cipher to "wrap around") until the input x_ℓ and the output y_ℓ for one particular permutation P_ℓ are obtained (but still undefined inside P_ℓ history). This permutation is then "adapted" by setting $P_\ell(x_\ell) := y_\ell$ so that the corresponding input and output for the simulated Even-Mansour cipher and for the ideal cipher match. A moment of thinking should make clear that the simulator cannot complete each and every partial chain created in its history, since this would create a "chain reaction" leading to an exponential running time and an exponential number of ideal cipher queries from the simulator. Hence, one must make a careful and parsimonious choice of "detection zones" for deciding which partial chains to complete. In addition, one must ensure that the simulator never overwrites an entry when adapting permutation P_ℓ, thereby rendering a previously completed chain inconsistent. How exactly this is done is very similar to the case of the Feistel construction [Seu09, HKT11], and we refer to Section 4.1 for a more detailed overview.

As a retrospective afterthought, we note that the Feistel and the iterated Even-Mansour indifferentiability results are not that far apart: they both tell how to construct a "big object" (which in both cases has some specific syntactic constraints which are relevant only from a cryptographic perspective) taking $2n$ bits of input (the left and right n-bit halves of the input in the case of the Feistel network, and the key and the plaintext in the case of the iterated Even-Mansour cipher) from smaller objects with only n bits of input (fourteen n-bit to n-bit functions for the Feistel network, and twelve n-bit permutations for the iterated Even-Mansour cipher).

1.6 Related Work

In a prior and independent work [ABD+13], Andreeva *et al.* proved a result which is close and complementary to ours: they showed that the iterated Even-Mansour construction with *five* rounds and a key derivation function *modeled as a random oracle* is indifferentiable from an ideal cipher. Though significantly reducing the number of rounds required for the proof to go through, and lifting the restriction that the master key length be equal to the block length of the permutations, their technique puts a strong burden on the key derivation function, which can hardly be seen as close to a random oracle in most concrete block ciphers. In fact, most

key schedules, such as the one of AES, are "lightweight" and invertible, which makes our result (where the key derivation function has no cryptographic role) more relevant to practice.

Taken together, the two results indicate, not too surprisingly, that using a cryptographically strong key schedule, though not being necessary, enables to lower the number of rounds needed to obtain an ideal cipher (however this interpretation must be taken cautiously: it may well be that, say, the iterated Even-Mansour cipher with four rounds is indifferentiable from an ideal cipher, independently of the cryptographic strength of the key schedule).

Regarding the purely theoretical question of the minimal number of n-bit permutations needed to construct an n-bit block and n-bit key ideal cipher, the result of [ABD+13] is better since it enables to use only six independent permutations (using a independent random permutation P_0 to build a key derivation function $k \mapsto P_0(k) \oplus k$ which is indifferentiable from a random function).

1.7 Open Problems

The minimal number of rounds necessary in order for the single-key iterated Even-Mansour construction to achieve indifferentiability from an ideal cipher remains unclear. We know by our results of Section 3 that four rounds are necessary, and it may well be that this is also sufficient, but we think that substantially new techniques will be required to prove it (the situation is similar to the one of the Feistel construction: we know that six rounds are necessary in order to achieve indifferentiability from a random permutation [CPS08], but currently the best number of rounds proved to achieve indifferentiability is fourteen [HKT11]). In Appendix A, we make some observations about the obstacles that appear when trying to obtain a proof (with techniques similar to the ones used here) for the 8-round single-key iterated Even-Mansour construction.

As we already emphasized several times, our result crucially relies on the fact that the key length is equal to the block length of the cipher. An interesting open question (practically quite relevant to the understanding of the security of AES-256 and more generally of block ciphers with n-bit blocks and $2n$-bit keys) is whether it is possible to come with a simple construction of a block cipher with $2n$-bit keys based on a constant number of n-bit permutations which would be indifferentiable from an ideal cipher with n-bit blocks and $2n$-bit keys, without using a cryptographically strong assumption about the key derivation function.[7] We note that cascading two 12-round iterated Even-Mansour constructions with two independent keys does not work (in fact one can easily see that cascading two ideal ciphers with two independent n-bit keys is *not* indifferentiable from an ideal cipher with $2n$-bit keys). Interleaving the xoring of two n-bit keys k_1 and k_2 in the iterated Even-Mansour construction (with sufficiently many rounds) seems a more promising approach (this is for example what is done for the block cipher LED-128 [GPPR11] which has 64-bit blocks and 128-bit keys). As an extremely preliminary analysis, we note that in this construction, pairs of inputs and outputs (x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}) for three consecutive permutations P_i, P_{i+1}, and P_{i+2} are sufficient to recover the two corresponding n-bit keys (say $k_1 = y_i \oplus x_{i+1}$ and $k_2 = y_{i+1} \oplus x_{i+2}$), and hence uniquely specify the whole computation path inside the construction. Hence, a chain detection/completion strategy very similar to the one used in this paper might be able to work. It would require three detection rounds at the beginning, three detection rounds in the

[7] Indeed, if one allows the key derivation function to be modeled as a random oracle, the problem (for arbitrary key length) is solved by the 5-round iterated Even-Mansour construction [ABD+13].

middle, and three detection rounds at the end of the cipher, plus two adaptation rounds, and four buffer rounds surrounding the adaptation rounds, amounting to a total of fifteen rounds.

1.8 Organization

We start with some preliminaries and important definitions in Section 2. In Section 3, we describe attacks in the indifferentiability setting against the iterated Even-Mansour cipher with independent keys (for any number of rounds), and against the single-key iterated Even-Mansour cipher with three rounds or less. Finally, in Section 4, we prove that the single-key iterated Even-Mansour construction with twelve rounds is indifferentiable from an ideal cipher.

2 Preliminaries

2.1 Notation and Definitions

Given a finite non-empty set S, we write $s \leftarrow_{\$} S$ to mean that a value is sampled uniformly at random from S and assigned to s. The security parameter will be denoted n and will be identified with the block length of permutations in the Even-Mansour construction. We will write $f \in \texttt{poly}(n)$ to denote a polynomially bounded function and $f \in \texttt{negl}(n)$ to denote a negligible function. For $\delta \in \{+, -\}$, we denote $\bar{\delta}$ the opposite of δ.

In the following, we will use calligraphic fonts ($\mathcal{A}, \mathcal{B}, \ldots$) to denote interactive Turing machines, and typewriter fonts to denote `Procedures` attached to these machines. A distinguisher is an oracle Turing Machine $\mathcal{D}$ which takes as input a security parameter 1^n, has access to a set of oracles $O_1, \ldots, O_m$, and outputs a bit b, an experiment we denote $\mathcal{D}^{O_1,\ldots,O_m} = b$. We will always consider distinguishers that are deterministic and computationally unbounded, and restricted only with respect to the number of oracle queries they make.

An ideal primitive is a probability distribution on some set of functions, and will be denoted with bold fonts. In the corresponding *model*, a function is drawn at random from the corresponding distribution (say $\boldsymbol{F}$) and all parties (say $\mathcal{M}$) involved in the security experiment are given oracle access to the corresponding function, which we simply denote $\mathcal{M}^{\boldsymbol{F}}$. In the following we will consider the following two ideal primitives:

- a random permutation $\boldsymbol{P}_i$ on $\{0,1\}^n$, which is a permutation drawn at random from the set of all permutations on $\{0,1\}^n$, and which can be accessed in the two directions $\boldsymbol{P}_i(x)$ and $\boldsymbol{P}_i^{-1}(y)$; we will use the notation $\boldsymbol{P} = (\boldsymbol{P}_1, \ldots, \boldsymbol{P}_r)$ to denote a tuple of independent random permutations;
- an ideal cipher $\boldsymbol{E}$ with message space and key space $\{0,1\}^n$, which is drawn at random from the set of all block ciphers of this form, and which can be accessed in encryption, denoted $\boldsymbol{E}(k, x)$, and decryption, denoted $\boldsymbol{E}^{-1}(k, y)$.

2.2 Indifferentiability

We recall the usual definition of indifferentiability.

Definition 1. *Let $q, \sigma : \mathbb{N} \to \mathbb{N}$ and $\varepsilon : \mathbb{N} \to \mathbb{R}$ be three functions of the security parameter n. A Turing machine $\mathcal{C}$ with oracle access to an ideal primitive $\boldsymbol{F}$ is said to be statistically and strongly (q, σ, ε)-indifferentiable from an ideal primitive $\boldsymbol{G}$ if there exists an interactive*

Turing machine $\mathcal{S}$ with oracle access to $\boldsymbol{G}$ such that for any distinguisher $\mathcal{D}$ making at most q queries, $\mathcal{S}$ makes at most σ oracle queries, and the following holds:

$$\left|\Pr\left[\mathcal{D}^{\boldsymbol{G},\mathcal{S}^{\boldsymbol{G}}}=1\right]-\Pr\left[\mathcal{D}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right]\right|\leq\varepsilon\ .$$

$\mathcal{C}^{\boldsymbol{F}}$ is simply said to be statistically and strongly indifferentiable from $\boldsymbol{G}$ if for any $q\in\mathtt{poly}(n)$, the above definition is fulfilled with $\sigma\in\mathtt{poly}(n)$ and $\varepsilon\in\mathtt{negl}(n)$.

This definition does not refer to the running time of $\mathcal{S}$ and $\mathcal{D}$. When only polynomial-time algorithms are considered, indifferentiability is said to be *computational*. Weak indifferentiability is defined as above, but the order of quantifiers for the distinguisher and the simulator are switched (for all distinguisher, there is a simulator...).

In this paper, and similarly to [HKT11], we will slightly tweak the definition of strong indifferentiability as follows: we will describe a simulator which, for any distinguisher $\mathcal{D}$ making a polynomial number of queries, runs in polynomial time and makes at most σ queries *with overwhelming probability* (rather than probability one) in system $\mathcal{D}^{\boldsymbol{G},\mathcal{S}^{\boldsymbol{G}}}$. This is not a big concern since any such simulator $\mathcal{S}$ can be transformed into a simulator $\mathcal{S}'$ for weak indifferentiability (which is sufficient for the composition theorem of [MRH04] to hold) which takes the maximal number of queries q of $\mathcal{D}$ as input, and aborts when its number of queries becomes larger than σ (computed as a function of q), hence making at most σ queries with probability one.

2.3 The Iterated Even-Mansour Cipher

Fix an integer $r\geq 1$. Let $P=(P_1,\ldots,P_r)$ be a tuple of permutations on $\{0,1\}^n$. The r-round iterated Even-Mansour construction associated with P, denoted $\bar{\mathcal{C}}_r^P$, is the block cipher with message space $\{0,1\}^n$ and key space $(\{0,1\}^n)^{r+1}$ which maps a message x and a key $(k_0,\ldots,k_r)$ to the ciphertext defined by:

$$\bar{\mathcal{C}}_r^P((k_0,\ldots,k_r),x)=k_r\oplus P_r(k_{r-1}\oplus P_{r-1}(\cdots P_2(k_1\oplus P_1(k_0\oplus x))\cdots))\ .$$

Let $\gamma=(\gamma_0,\ldots,\gamma_r)$ be a tuple of efficiently invertible permutations on $\{0,1\}^n$. The *single-key* r-round iterated Even-Mansour construction associated with P and γ, denoted $\mathcal{C}_r^{P,\gamma}$, is the block cipher with message space $\{0,1\}^n$ and key space $\{0,1\}^n$ which maps a message x and a key k to the ciphertext defined by (see Figure 1):

$$\mathcal{C}_r^{P,\gamma}(k,x)=\gamma_r(k)\oplus P_r(\gamma_{r-1}(k)\oplus P_{r-1}(\cdots P_2(\gamma_1(k)\oplus P_1(\gamma_0(k)\oplus x))\cdots))\ .$$

In all the following, we will focus on the case where all permutations γ_i are the identity, and simply denote $\mathcal{C}_r^P$ the resulting cipher, namely:

$$\mathcal{C}_r^P(k,x)=k\oplus P_r(k\oplus P_{r-1}(\cdots P_2(k\oplus P_1(k\oplus x))\cdots))\ .$$

We stress that our main result (Theorem 2) holds for arbitrary permutations γ_i as long as they are efficiently invertible.

3 Attacks

In this section we present attacks against the iterated Even-Mansour cipher with independent keys, and against the single-key iterated Even-Mansour cipher for three rounds or less. To describe the corresponding distinguisher, we generically denote (E,P) the oracles to which it has access, where (E,P) is either $(\mathcal{C}_r^{\boldsymbol{P}},\boldsymbol{P})$ or $(\boldsymbol{E},\mathcal{S}^{\boldsymbol{E}})$.

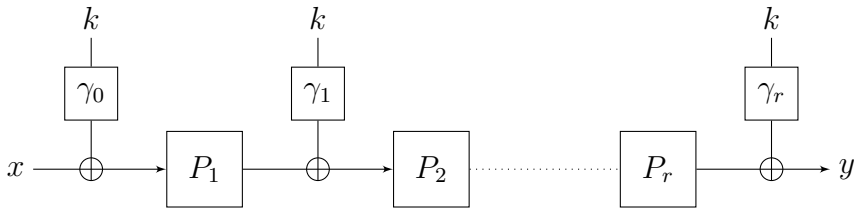

Fig. 1. The single-key iterated Even-Mansour cipher with r rounds $\mathcal{C}_r^{P,\gamma}$. We focus in this paper on the special case $\gamma_i = \mathrm{Id}$ for $i = 0, \ldots, r$.

3.1 Attack on Independent Keys

We start with a simple attack against $\bar{\mathcal{C}}_r^{\boldsymbol{P}}$, *i.e.* the iterated Even-Mansour cipher where the $r+1$ round keys $(k_0, \ldots, k_r)$ are independent, showing that this construction is not indifferentiable from an ideal cipher with $(r+1)n$-bit keys. This had already been observed in [BKL+12] (more precisely, the authors observed that there are trivial related-key attacks in this case).

Fix any non-zero constant $c \in \{0,1\}^n$. Consider the distinguisher $\mathcal{D}^{(E,P)}$ proceeding as follows:

(1) choose an arbitrary input x and an arbitrary key k_0, and define $x' := x \oplus c$ and $k_0' := k_0 \oplus c$;
(2) choose arbitrary keys $k_1, \ldots, k_r$; denote $k := (k_0, k_1, \ldots, k_r)$ and $k' := (k_0', k_1, \ldots, k_r)$;
(3) query $y := E(k, x)$ and $y' := E(k', x')$;
(4) if $y = y'$ output 1, otherwise output 0.

Note that this distinguisher does not make any query to $\boldsymbol{P}/\mathcal{S}$. Clearly, $\mathcal{D}$ always outputs 1 when interacting with $(\bar{\mathcal{C}}_r^{\boldsymbol{P}}, \boldsymbol{P})$. On the other hand, when interacting with $(\boldsymbol{E}, \mathcal{S}^{\boldsymbol{E}})$ for any simulator $\mathcal{S}$, we see that the probability that $\boldsymbol{E}(k, x) = \boldsymbol{E}(k', x')$ is exactly 2^{-n}. Hence the distinguishing advantage of $\mathcal{D}$ is exactly $1 - 2^{-n}$.

3.2 Attack on Three Rounds

Here, we show an attack on the single-key iterated Even-Mansour cipher with three rounds $\mathcal{C}_3^{\boldsymbol{P}}$, with $\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3)$, and $(\gamma_0, \ldots, \gamma_3) = (\mathrm{Id}, \ldots, \mathrm{Id})$ for the key derivation. This can easily be simplified into attacks for one and two rounds. An attack applicable to arbitrary efficiently invertible permutations $(\gamma_0, \ldots, \gamma_3)$ was independently described in [ABD+13]. Our attack has the additional advantage that is enables to find a so-called evasive relation [CGH98, MPS12] on the inputs and outputs of the cipher. Consider the following distinguisher $\mathcal{D}^{(E,P)}$ (see also Figure 2):

(1) choose arbitrary values $x_3, k, k' \in \{0,1\}^n$, with $k \neq k'$;
(2) compute $y_2 := x_3 \oplus k$ and $y_2' := x_3 \oplus k'$;
(3) query $x_2 := P_2^{-1}(y_2)$ and $x_2' := P_2^{-1}(y_2')$;
(4) compute $y_1 := x_2 \oplus k$ and $y_1' := x_2' \oplus k'$;
(5) compute $k'' := y_1 \oplus x_2'$ and $k''' := y_1' \oplus x_2$;
(6) if k, k', k'', and k''' are not pairwise distinct, output 0;
(7) query $x_1 := P_1^{-1}(y_1)$ and $x_1' := P_1^{-1}(y_1')$;
(8) compute $x := x_1 \oplus k$, $x' := x_1' \oplus k'$, $x'' := x_1 \oplus k''$, and $x''' := x_1' \oplus k'''$;

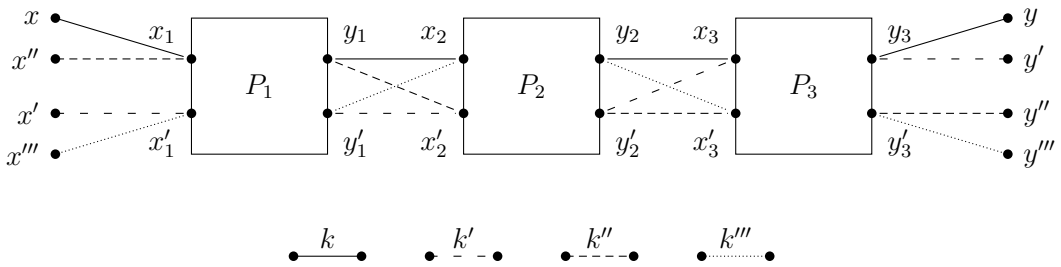

Fig. 2. The attack on the single-key iterated Even-Mansour cipher with three rounds $\mathcal{C}_3^{\boldsymbol{P}}$.

(9) query $y := E(k, x)$, $y' := E(k', x')$, $y'' := E(k'', x'')$ and $y''' := E(k''', x''')$;
(10) if $y \oplus y' \oplus y'' \oplus y''' = 0$ output 1, otherwise output 0.

This distinguisher implies the following theorem.

Theorem 1. *The 3-round single-key iterated Even-Mansour cipher $\mathcal{C}_3^{\boldsymbol{P}}$ using three independent random n-bit permutations (and $(\gamma_0, \ldots, \gamma_3) = (\mathrm{Id}, \ldots, \mathrm{Id})$ for the key derivation) is not indifferentiable from an ideal cipher with n-bit blocks and n-bit keys.*

Proof. We lower bound the distinguishing advantage of $\mathcal{D}$. First, we show that $\mathcal{D}$ outputs 1 with overwhelming probability when interacting with $(\mathcal{C}_3^{\boldsymbol{P}}, \boldsymbol{P})$. As a first step, observe that $k \neq k' \Rightarrow y_2 \neq y'_2 \Rightarrow x_2 \neq x'_2$. Since $k'' = k \oplus x_2 \oplus x'_2$ and $k''' = k' \oplus x_2 \oplus x'_2$, we also have $k'' \neq k'''$, $k'' \neq k$, and $k''' \neq k'$. Hence k, k', k'', and k''' are not pairwise distinct iff $x_2 \oplus x'_2 = k \oplus k'$, which happens with probability $1/(2^n - 1)$ since x_2 and x'_2 are uniformly random distinct values. Hence $\mathcal{D}$ outputs 0 with probability at most $1/(2^n - 1)$ at step (6). We show now that conditioned on $\mathcal{D}$ not outputting 0 at step (6), it always outputs 1. For this, note that by definition of y_1, y'_1, k'', and k''', we have:

$$\begin{aligned} k \oplus k' \oplus k'' \oplus k''' &= (y_1 \oplus x_2) \oplus (y'_1 \oplus x'_2) \oplus (y_1 \oplus x'_2) \oplus (y'_1 \oplus x_2) \\ &= 0 \ . \end{aligned}$$

Consider now the computation path inside the iterated Even-Mansour cipher for each input (k, x), (k', x'), (k'', x''), and (k''', x'''). Clearly, the input to P_3 for both inputs (k, x) and (k', x') is x_3. The input to P_3 for (k'', x'') and (k''', x''') are respectively $y'_2 \oplus k''$ and $y_2 \oplus k'''$. By definition of all intermediate values set by the distinguisher, we see that

$$k'' \oplus k''' = k \oplus k' = (x_3 \oplus y_2) \oplus (x_3 \oplus y'_2) = y_2 \oplus y'_2 \ ,$$

so that $y'_2 \oplus k'' = y_2 \oplus k'''$. Hence the input to P_3 for both inputs (k'', x'') and (k''', x''') is the common value $x'_3 := y'_2 \oplus k'' = y_2 \oplus k'''$. Hence, the four corresponding outputs are resp. $y = y_3 \oplus k$, $y' = y_3 \oplus k'$, $y'' = y'_3 \oplus k''$, and $y''' = y'_3 \oplus k'''$, where $y_3 = P_3(x_3)$ and $y'_3 = P_3(x'_3)$, which implies $y \oplus y' \oplus y'' \oplus y''' = k \oplus k' \oplus k'' \oplus k''' = 0$.

Consider now what happens when $\mathcal{D}$ interacts with $(\boldsymbol{E}, \mathcal{S}^{\boldsymbol{E}})$ for some efficient simulator $\mathcal{S}$ which makes at most σ queries when $\mathcal{D}$ makes at most q queries. Denote $\mathcal{M}$ the combination of $\mathcal{D}$ and $\mathcal{S}$. Note that $\mathcal{M}$ is an oracle Turing machine which makes at most $q' = q + \sigma$ queries in total to $\boldsymbol{E}$. Whenever $\mathcal{D}$ outputs 1, we see that $\mathcal{M}$ has successfully found four inputs

(k, x), (k', x'), (k'', x'') and (k''', x'''), where k, k', k'', and k''' are pairwise distinct, which, together with their corresponding outputs $y = \boldsymbol{E}(k, x)$, $y' = \boldsymbol{E}(k', x')$, $y'' = \boldsymbol{E}(k'', x'')$, and $y''' = \boldsymbol{E}(k''', x''')$ satisfy:

$$\begin{cases} k \oplus k' \oplus k'' \oplus k''' = 0 \\ x \oplus x' \oplus x'' \oplus x''' = 0 \\ y \oplus y' \oplus y'' \oplus y''' = 0 \end{cases} .$$

Note that the first two conditions holds with probability 1 by construction of the distinguisher. It remains to show that $\mathcal{M}$ has only a negligible probability to find such values. Consider the q' queries of $\mathcal{M}$ to $\boldsymbol{E}$ sequentially, and denote $\texttt{Bad}$ the event that such values can be found among the q' queries, and $\texttt{Bad}_i$ the event that such values can be found among the i first queries. We will upper bound $\Pr[\texttt{Bad}_i | \overline{\texttt{Bad}}_{i-1}]$. Consider the i-th query, and assume that this is an encryption query $y_i := \boldsymbol{E}(k_i, x_i)$ (the reasoning is similar for a decryption query). We can assume *wlog* that k_i is distinct from all keys appearing in previous queries, so that y_i is uniformly random. Then $\texttt{Bad}_i$ happens only if y_i takes one of at most $\binom{i-1}{3} \leq i^3$ values, hence with probability at most $i^3/2^n$. By summing over i, we see that $\texttt{Bad}$ happens with probability at most $\sum_{i=1}^{q'} i^3/2^n \leq q'^4/2^{n+1}$. Since q' is polynomial if $\mathcal{S}$ makes a polynomial number σ of queries, we see that $\mathcal{D}$ outputs 1 with negligible probability when interacting with $(\boldsymbol{E}, \mathcal{S}^{\boldsymbol{E}})$. Hence $\mathcal{D}$ has advantage negligibly close to 1, which concludes the proof. □

4 Indifferentiability for Twelve Rounds

In this section we prove the main result of this paper, which is the following theorem.

Theorem 2. *For any q, the 12-round single-key iterated Even-Mansour cipher $\mathcal{C}_{12}^{\boldsymbol{P},\gamma}$ with twelve independent random n-bit permutations $\boldsymbol{P} = (\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{12})$, and fixed, efficiently invertible n-bit permutations $\gamma = (\gamma_0, \ldots, \gamma_{12})$ for the key schedule, is strongly and statistically (q, σ, ε)-indifferentiable from an ideal cipher $\boldsymbol{E}$ with n-bit blocks and n-bit keys, where:*

$$\sigma = 2^7 \times q^4 \quad \textit{and} \quad \varepsilon = \frac{2^{91} \times q^{12}}{2^n} .$$

To prove this, we will describe an efficient simulator $\mathcal{S}$, and show that the two systems $(\mathcal{C}_{12}^{\boldsymbol{P},\gamma}, \boldsymbol{P})$ and $(\boldsymbol{E}, \mathcal{S}^{\boldsymbol{E}})$ are indistinguishable. For simplicity we focus on the case where all γ_i's are the identity, but the generalization is straightforward.

Notational convention. In all this section, we will use the following useful notational convention: we will interchangeably denote the input to the ideal cipher or the iterated Even-Mansour cipher x or y_0, and the output y or x_{13}.

4.1 Informal Description of the Simulator

We start with a high-level view of the simulator (see also Figure 3). It offers an interface to the distinguisher for querying the simulated permutations, which formally takes the form of a public procedure $\texttt{Query}(i, \delta, z)$, where $i \in \{1, \ldots, 12\}$ names the permutation, $\delta \in \{+, -\}$ tells whether this is a direct or indirect query, and $z \in \{0,1\}^n$ is the actual value queried. The simulator maintains an history for the simulated permutations under the form of hash tables $P_1, \ldots, P_{12}$. Each such table maps entries $(\delta, z) \in \{+, -\} \times \{0,1\}^n$ to values $z' \in \{0,1\}^n$.

We denote P_i^+, resp. P_i^-, the (time-dependent) sets of strings $z \in \{0,1\}^n$ such that $P_i(+,z)$, resp. $P_i(-,z)$, is defined. When the simulator receives a query (i,δ,z), it looks in hash table P_i to see whether the corresponding answer $P_i(\delta,z)$ is already defined. When this is the case, it outputs the answer and waits for the next query. Otherwise, it draws a uniformly random answer z' and defines in hash table $P_i(\delta,z) := z'$, as well as the answer to the opposite query $P_i(\bar{\delta},z') := z$ (note that this last assignment may overwrite an entry in P_i).

Additionally, before outputting the answer z', and for some specific values of (i,δ), the simulator triggers a *chain detection* mechanism followed by a *chain completion* mechanism to ensure consistency of its answers with the ideal cipher $\boldsymbol{E}$. An essential point to notice about the iterated Even-Mansour cipher in order to understand these mechanisms is that given an output value y_i for permutation P_i and an input value x_{i+1} for permutation P_{i+1}, it is possible to compute the corresponding key $k = y_i \oplus x_{i+1}$, and therefore to move forward and backward in the construction up and down to the corresponding input x and output y to the cipher. Hence, any tuple (y_i, x_{i+1}, i) (a so-called *partial chain* later in the reasoning) defines a unique computation path inside the whole construction. This is the exact analogue of the property of the Feistel construction that the input values to two consecutive round functions uniquely define the computation path inside the Feistel network.

There are exactly six such values of (i,δ) for which the simulator performs additional steps: $(2,+)$, $(6,+)$, $(6,-)$, $(7,+)$, $(7,-)$, and $(11,-)$. The cases $(2,+)$ and $(11,-)$ are similar. When receiving a query $(2,+,x_2)$ for which the answer is undefined yet, the simulator, after having drawn a random answer y_2 to this query, considers all values $y_1 \in P_1^-$, computes the corresponding key $k := y_1 \oplus x_2$, and moves backward in the iterated Even-Mansour cipher by computing $x_1 := P_1(-,y_1)$, $y_0 := x_1 \oplus k$, $x_{13} := \boldsymbol{E}(k,y_0)$ (hence making a query to the ideal cipher), and $y_{12} := x_{13} \oplus k$, and checks whether $y_{12} \in P_{12}^-$. When this is the case, it enqueues in a queue QUEUE the tuple $(y_0, x_1, 0, 4)$. The first three elements $(y_0, x_1, 0)$ specify the partial chain that must be completed, while the last element $\ell = 4$ specifies which permutation will be adapted during completion of the chain to ensure consistency with $\boldsymbol{E}$. The behavior of the simulator when receiving a query $(11,-,y_{11})$ is symmetric: after having drawn a random answer x_{11}, for all $x_{12} \in P_{12}^+$, it moves forward in the iterated Even-Mansour cipher to check whether the corresponding value x_1 is in P_1^+, and if so enqueues the corresponding tuple $(y_0, x_1, 0, 9)$ (note that in this case adaptation will take place at permutation P_9).

The four remaining cases $(i,\delta) = (6,+)$, $(6,-)$, $(7,+)$, and $(7,-)$ are similar, except that there is no check: the simulator enqueues a tuple $(y_6, x_7, 6, \ell)$ for each newly generated pair $(y_6, x_7) \in P_6^- \times P_7^+$. If this was a query with $i = 6$, then adaptation will take place at $\ell = 4$, while if this was a query with $i = 7$, adaptation will take place at $\ell = 9$. Assume for a concrete example that the simulator receives a query $(6,+,x_6)$ whose answer is undefined yet. Then it draws a random answer $y_6 \leftarrow_{\$} \{0,1\}^n$, and enqueues $(y_6, x_7, 6, 4)$ for all $x_7 \in P_7^+$.

Immediately after having enqueued newly created chains (y_i, x_{i+1}, i, ℓ), the simulator starts completing the partial chains, by dequeuing tuples from QUEUE. For this, when dequeuing (y_i, x_{i+1}, i, ℓ), it computes the key $k := y_i \oplus x_{i+1}$, and moves forward and backward in the iterated Even-Mansour cipher, possibly defining missing permutations values $P_i(+,\cdot)$ or $P_i(-,\cdot)$, and making a query to $\boldsymbol{E}(k,\cdot)$ to "wrap around", until it reaches the input value x_ℓ for P_ℓ (when moving forward) and the corresponding output y_ℓ (when moving backward). It finally "adapts" permutation P_ℓ by setting $P_\ell(+,x_\ell) := y_\ell$ and $P_\ell(-,y_\ell) := x_\ell$ in order to ensure consistency of the entire chain with $\boldsymbol{E}$. It also adds chains that have been completed in a set COMPLETED in order to avoid completing them twice. While completing a chain and adding

possibly missing permutation values, the simulator uses the same chain detection mechanism as when receiving a direct query from the distinguisher. Hence new tuples may be enqueued while dequeuing and completing a chain, and the simulator keeps dequeuing tuples until the queue is empty. When this is the case, it returns the answer to the original query of the distinguisher.

As in the indifferentiability proof of the Feistel construction, there will be two crucial points to show: first, that the recursive chain completion mechanism terminates in polynomial time (except maybe with negligible probability); second, that the simulator can always adapt, *i.e.* that it never has (or only with negligible probability) to overwrite previously defined entries when adapting a chain, which would render previously completed chains inconsistent with the ideal cipher $\boldsymbol{E}$. Permutations P_3, P_5, P_8, and P_{10} (*i.e.* the permutations surrounding the two adaptation rounds P_4 and P_9) will play a key role while proving this last point: they will ensure that no bad collisions occur at the input or output of the two permutations used for adapting chains.

4.2 Formal Description of the Simulator

We now give a formal description of the simulator $\mathcal{S}$ in pseudocode. We will use the following notations. The simulator maintains hash tables $P_1, \ldots, P_{12}$ which are initially empty. Any such hash table contains entries $(\delta, z) \in \{+,-\} \times \{0,1\}^n$ associated with values $z' \in \{0,1\}^n$. We will note $(\delta, z) \in P_i$ to mean that entry (δ, z) is in the hash table, and denote $P_i(\delta, z)$ the associated value. For $\delta \in \{+,-\}$ we will denote P_i^{δ} the set of values $z \in \{0,1\}^n$ such that $(\delta, z) \in P_i$. We say that P_i *defines a partial permutation on* $\{0,1\}^n$ (at some point during the execution) if for all $x, y \in \{0,1\}^n$, $(+, x) \in P_i$ and $P_i(+, x) = y$ if and only if $(-, y) \in P_i$ and $P_i(-, y) = x$. We warn that hash tables P_i will not necessarily define partial permutations at any time during the simulation. Nevertheless, a large part of the indifferentiability proof will be to show that this holds with overwhelming probability.[8]

Later in the proof, we will consider a slightly different simulator $\mathcal{T}(\varphi)$, whose differences with $\mathcal{S}$ are captured by framed statements, which can be ignored for the moment.

Simulator $\mathcal{S}$: **Simulator $\mathcal{T}(\varphi)$:**

Variables:
  hash tables $P_1, \ldots, P_{12}$, initially empty
  queue QUEUE, initially empty
  set COMPLETED, initially empty

The simulator offers one public interface `Query` accessible by the distinguisher for simulating the permutations. It can be queried with a tuple (i, δ, z) where $i \in \{1, \ldots, 12\}$ names the permutation, $\delta \in \{+,-\}$ tells whether this is a direct or indirect query, and $z \in \{0,1\}^n$ is the actual value queried. This procedure first calls `InQuery` (a procedure described later which internally draws a random answer if needed, and fills the queue with newly created chains), and then completes the chains in QUEUE.

public procedure `Query`(i, δ, z):
  $z' :=$ `InQuery`(i, δ, z)

[8] Note that assignments in any table P_i will always occur in pairs, which will be considered as atomic events since P_i does transiently not define a partial permutation between the two assignments in the pair.

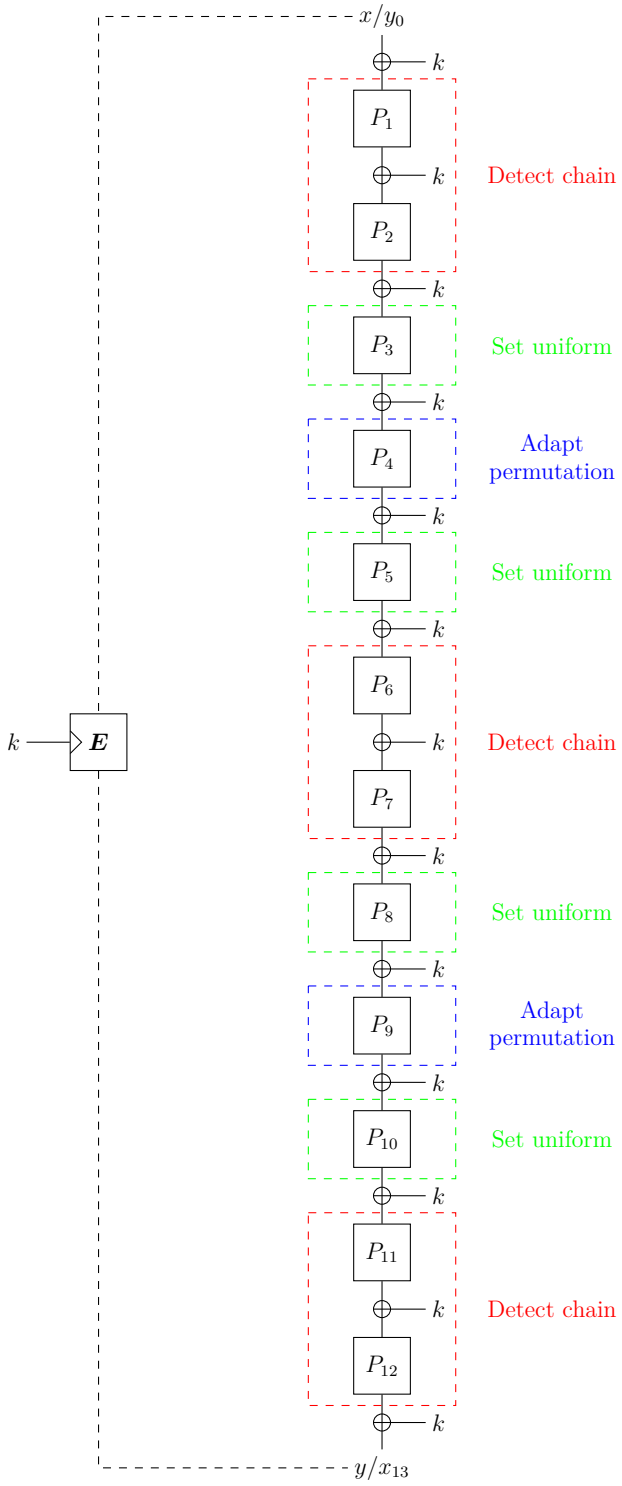

Fig. 3. Detection and adaptation zones used by the simulator.

8 **while** QUEUE $\neq \emptyset$ **do**
9 $(y_j, x_{j+1}, j, \ell) :=$ QUEUE.`Dequeue`()
10 **if** $(y_j, x_{j+1}, j) \notin$ COMPLETED **then**
11 $\backslash\backslash$ complete the partial chain
12 $(y_{\ell-2}, x_{\ell-1}) =$ `EvalForward`$(y_j, x_{j+1}, j, \ell - 1)$
13 $(y_{\ell+1}, x_{\ell+2}) =$ `EvalBackward`$(y_j, x_{j+1}, j, \ell + 1)$
14 `Adapt`$(y_{\ell-2}, x_{\ell-1}, y_{\ell+1}, x_{\ell+2}, \ell)$
15 $\backslash\backslash$ add corresponding partial chains to set COMPLETED
16 $(y_0, x_1) =$ `EvalBackward`$(y_j, x_{j+1}, j, 0)$
17 $(y_6, x_7) =$ `EvalForward`$(y_j, x_{j+1}, j, 7)$
18 COMPLETED := COMPLETED $\cup \{(y_0, x_1, 0), (y_6, x_7, 6)\}$
19 **return** z'

Procedure `Adapt` describes more specifically what happens around adaptations rounds $\ell = 4$ or $\ell = 9$. Describing this as an independent procedure will ease the discussion during the proof.

20 **private procedure** `Adapt`$(y_{\ell-2}, x_{\ell-1}, y_{\ell+1}, x_{\ell+2}, \ell)$:
21 $k := y_{\ell-2} \oplus x_{\ell-1}$
22 $y_{\ell-1} :=$ `InQuery`$(\ell - 1, +, x_{\ell-1})$
23 $x_\ell := y_{\ell-1} \oplus k$
24 $k' := y_{\ell+1} \oplus x_{\ell+2}$ $\backslash\backslash$ always equal to k by construction
25 $x_{\ell+1} :=$ `InQuery`$(\ell + 1, -, y_{\ell+1})$
26 $y_\ell := x_{\ell+1} \oplus k'$
27 `ForceVal`(x_ℓ, y_ℓ, ℓ)

Procedure `ForceVal` is used to force the value of permutation P_ℓ for a given input/output pair (x_ℓ, y_ℓ). A key point in the indifferentiability proof will be to show that entries are never overwritten in P_ℓ when this procedure is called.

28 **private procedure** `ForceVal`(x_ℓ, y_ℓ, ℓ):
29 $P_\ell(+, x_\ell) := y_\ell$ $\backslash\backslash$ may overwrite an entry
30 $P_\ell(-, y_\ell) := x_\ell$ $\backslash\backslash$ may overwrite an entry

Procedure `InQuery` internally draws a random value when the answer to a query is undefined, and then calls procedure `EnqueueNewChains`.

31 **private procedure** `InQuery`(i, δ, z):
32 **if** $(\delta, z) \notin P_i$ **then**
33 $z' \leftarrow_\$ \{0,1\}^n$ $\boxed{z' := \varphi_i(\delta, z)}$
34 $P_i(\delta, z) := z'$
35 $P_i(\bar{\delta}, z') := z$ $\backslash\backslash$ may overwrite an entry
36 **if** $(i, \delta) \in \{(2, +), (6, +), (6, -), (7, +), (7, -), (11, -)\}$ **then**
37 `EnqueueNewChains`(i, δ, z)
38 **return** $P_i(\delta, z)$

Procedure `EnqueueNewChains` enqueues newly created chains (if any) for specific values of (i, δ).

39 **private procedure** `EnqueueNewChains`(i, δ, z):

40 **if** $(i, \delta) = (2, +)$ **then**
41 **forall** $(y_1, x_2, y_{12}) \in P_1^- \times \{z\} \times P_{12}^-$ **do**
42 $k := y_1 \oplus x_2$
43 $x_1 := P_1(-, y_1)$
44 $y_0 := x_1 \oplus k$
45 $x_{13} := y_{12} \oplus k$
46 **if** $\boldsymbol{E}(k, y_0) = x_{13}$ **then** | **if** $\mathcal{F}.\texttt{Check}(k, y_0, x_{13})$ **then**
47 QUEUE.$\texttt{Enqueue}(y_0, x_1, 0, 4)$
48 **else if** $(i, \delta) = (11, -)$ **then**
49 **forall** $(x_1, y_{11}, x_{12}) \in P_1^+ \times \{z\} \times P_{12}^+$ **do**
50 $k := y_{11} \oplus x_{12}$
51 $x_{13} := P_{12}(+, x_{12}) \oplus k$
52 $y_0 := x_1 \oplus k$
53 **if** $\boldsymbol{E}^{-1}(k, x_{13}) = y_0$ **then** | **if** $\mathcal{F}.\texttt{Check}(k, y_0, x_{13})$ **then**
54 QUEUE.$\texttt{Enqueue}(y_0, x_1, 0, 9)$
55 **else if** $(i, \delta) = (6, +)$ **then**
56 **forall** $(y_6, x_7) \in \{P_6(+, z)\} \times P_7^+$ **do**
57 QUEUE.$\texttt{Enqueue}(y_6, x_7, 6, 4)$
58 **else if** $(i, \delta) = (6, -)$ **then**
59 **forall** $(y_6, x_7) \in \{z\} \times P_7^+$ **do**
60 QUEUE.$\texttt{Enqueue}(y_6, x_7, 6, 4)$
61 **else if** $(i, \delta) = (7, +)$ **then**
62 **forall** $(y_6, x_7) \in P_6^- \times \{z\}$ **do**
63 QUEUE.$\texttt{Enqueue}(y_6, x_7, 6, 9)$
64 **else if** $(i, \delta) = (7, -)$ **then**
65 **forall** $(y_6, x_7) \in P_6^- \times \{P_7(-, z)\}$ **do**
66 QUEUE.$\texttt{Enqueue}(y_6, x_7, 6, 9)$

$\texttt{EvalForward}(y_i, x_{i+1}, i, \ell)$ is a procedure which evaluates the Even-Mansour construction forward for a pair (y_i, x_{i+1}) and returns the pair $(y_{\ell-1}, x_\ell)$, where x_j is the input to permutation P_j for $j \in \{1, \ldots 12\}$ and x_{13} is the output of the Even-Mansour construction, and y_j is the output of permutation P_j for $j \in \{1, \ldots 12\}$ and y_0 is the input to the Even-Mansour construction. Similarly $\texttt{EvalBackward}(y_i, x_{i+1}, i, \ell)$ is a procedure which evaluates the Even-Mansour construction backward for a pair (y_i, x_{i+1}) and returns the pair $(y_\ell, x_{\ell+1})$.

67 **private procedure** $\texttt{EvalForward}(y_i, x_{i+1}, i, \ell)$:
68 $k := y_i \oplus x_{i+1}$
69 **while** $i \neq \ell - 1$ **do**
70 **if** $i = 12$ **then**
71 $y_0 := \boldsymbol{E}^{-1}(k, x_{13})$ | $y_0 := \mathcal{F}.\texttt{Dec}(k, x_{13})$
72 $x_1 := y_0 \oplus k$
73 $i := 0$
74 **else**
75 $y_{i+1} := \texttt{InQuery}(i + 1, +, x_{i+1})$
76 $x_{i+2} := y_{i+1} \oplus k$
77 $i := i + 1$

78 **return** $(y_{\ell-1}, x_\ell)$

79 **private procedure** $\texttt{EvalBackward}(y_i, x_{i+1}, i, \ell)$:
80 $k := y_i \oplus x_{i+1}$
81 **while** $i \neq \ell$ **do**
82 **if** $i = 0$ **then**
83 $x_{13} := \boldsymbol{E}(k, y_0)$ $\boxed{x_{13} := \mathcal{F}.\texttt{Enc}(k, y_0)}$
84 $y_{12} := x_{13} \oplus k$
85 $i := 12$
86 **else**
87 $x_i := \texttt{InQuery}(i, -, y_i)$
88 $y_{i-1} := x_i \oplus k$
89 $i := i - 1$
90 **return** $(y_\ell, x_{\ell+1})$

This concludes the description of the simulator $\mathcal{S}$.

Remark 1. When considering the more general iterated Even-Mansour construction $\mathcal{C}_{12}^{\boldsymbol{P},\gamma}$ using efficiently invertible permutations γ_i for key derivation as defined in Section 2.3, the simulator can easily be modified as follows: each time the simulator computes a key as $k := y_i \oplus x_{i+1}$, we replace the corresponding line with $k := \gamma_i^{-1}(y_i \oplus x_{i+1})$. When the same key is used to move forward or backward in the construction, the corresponding round keys can then be derived by the simulator as $k_j := \gamma_j(k)$ for the adequate indexes j.

4.3 Intermediate Systems

Denote Σ_1 the system $(\boldsymbol{E}, \mathcal{S}^{\boldsymbol{E}})$ and Σ_4 the system $(\mathcal{C}_{12}^{\boldsymbol{P}}, \boldsymbol{P})$. We introduce two intermediate systems Σ_2 and Σ_3 that we describe below. See also Figure 4 for a pictorial representation of all systems.

4.3.1 Second System

In system $\Sigma_2(\eta, \varphi) = (\mathcal{F}(\eta), \mathcal{T}(\varphi)^{\mathcal{F}(\eta)})$, the simulator $\mathcal{S}$ is replaced by a slightly different one denoted $\mathcal{T}(\varphi)$ and whose differences with $\mathcal{S}$ are captured by framed statements in the pseudocode of Section 4.2, and the ideal cipher $\boldsymbol{E}$ is replaced by what we call a two-sided keyed random function $\mathcal{F}(\eta)$. We now explain the differences between the two systems. First, the randomness in Σ_2 is made explicit through tables $\varphi = (\varphi_1, \ldots, \varphi_{12})$ used by $\mathcal{T}$ and η used by $\mathcal{F}$. Each table φ_i maps entries $(\delta, z) \in \{+,-\} \times \{0,1\}^n$ to uniform and independent values in $\{0,1\}^n$, while η maps entries $(\delta, k, z) \in \{+,-\} \times \{0,1\}^n \times \{0,1\}^n$ to uniform and independent values in $\{0,1\}^n$. Whenever $\mathcal{S}$ sets a value $P_i(\delta, z)$ randomly at line 33, $\mathcal{T}$ uses the value $\varphi_i(\delta, z)$ instead (note that this is just a syntactic modification which does not change the distribution of the answers of the simulator).

$\mathcal{F}$ offers the usual interfaces of an ideal cipher which will be denoted $\mathcal{F}.\texttt{Enc}(\cdot,\cdot)$ and $\mathcal{F}.\texttt{Dec}(\cdot,\cdot)$, taking a key k and a value $z \in \{0,1\}^n$ as input and returning a value $z' \in \{0,1\}^n$, and an additional procedure $\mathcal{F}.\texttt{Check}$ (only queried by the simulator during the experiment). $\mathcal{F}$ is defined by the pseudocode given below. It uses a hash table E which contains entries $(\delta, k, z) \in \{+,-\} \times \{0,1\}^n \times \{0,1\}^n$ associated with values $z' \in \{0,1\}^n$. We note $(\delta, k, z) \in E$

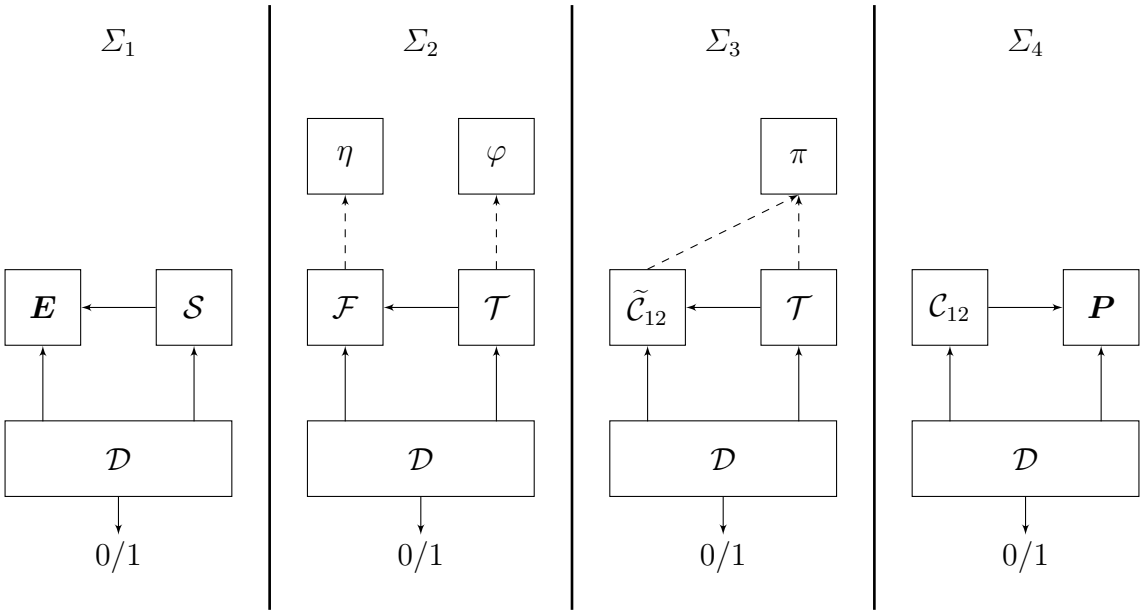

Fig. 4. Systems used in the indifferentiability proof.

to mean that entry (δ, k, z) is in the hash table and denote $E(\delta, k, z)$ the associated value. For $\delta \in \{+, -\}$, we denote E^δ the set of values (k, z) such that $(\delta, k, z) \in E$.

We say that E *defines a partial cipher* (at some point during the execution) if for all $k, x, y \in \{0,1\}^n$, $(+, k, x) \in E$ and $E(+, k, x) = y$ if and only if $(-, k, y) \in E$ and $E(-, k, y) = x$. We warn that E will not necessarily define a partial cipher at any time during the execution, however we will show that this holds with overwhelming probability.[9]

Two-sided keyed random function $\mathcal{F}(\eta)$:
Variables:
  Hashtable E (initially empty)

public procedure `Enc`(k, x):
  if $(+, k, x) \notin E$ **then**
    $y := \eta(+, k, x)$
    $E(+, k, x) := y$
    $E(-, k, y) := x$ \\ may overwrite an entry
  return $E(+, k, x)$

public procedure `Dec`(k, y):
  if $(-, k, y) \notin E$ **then**
    $x := \eta(-, k, y)$
    $E(-, k, y) := x$
    $E(+, k, x) := y$ \\ may overwrite an entry

[9] As for hash tables P_i, assignments in E will always occur in pair, which will be considered as atomic events since E does transiently not define a partial cipher between the two assignments in a pair.

```
17   return E(−, k, y)
18
19 public procedure Check(k, x, y):
20   if (+, k, x) ∈ E then return E(+, k, x) = y
21   if (−, k, y) ∈ E then return E(−, k, y) = x
22   return false
```

Said with words, procedure $\mathcal{F}$.Check enables to verify whether x is mapped through key k onto y without querying $\mathcal{F}$.Enc(k, x) or $\mathcal{F}$.Dec(k, y) and modifying the hash table E. Note that $\mathcal{F}(\eta)$ is stateful: its answers (for some fixed table η) depends on the order of the queries it receives.

4.3.2 Third System

In system $\Sigma_3(\pi) = (\widetilde{\mathcal{C}}_{12}(\pi), \mathcal{T}(\pi)^{\widetilde{\mathcal{C}}_{12}(\pi)})$, we make the following changes. We introduce tables $\pi = (\pi_1, \ldots, \pi_{12})$ which are uniformly random *permutation* tables, meaning that $\pi_i(+, x) = y$ if and only if $\pi_i(-, y) = x$. The two-sided keyed random function $\mathcal{F}$ is replaced by an Even-Mansour construction $\widetilde{\mathcal{C}}_{12}(\pi)$ defined by the pseudocode given below (note that this is simply the usual Even-Mansour construction associated with the tuple of permutations π, enhanced with a Check procedure). Also, the simulator $\mathcal{T}$ now uses the same tables π_i instead of uniform tables φ_i.

```
1  C̃12(π):
2  Variables:
3    Hashtable E′ (initially empty)
4
5  public procedure Enc(k, x)
6    if (+, k, x) ∉ E′ then
7      y0 := x
8      for i = 1 to 12 do
9        xi := yi−1 ⊕ k
10       yi := πi(+, xi)
11     y := y12 ⊕ k
12     E′(+, k, x) := y
13     E′(−, k, y) := x
14   return E′(+, k, x)
15
16 public procedure Dec(k, y)
17   if (−, k, y) ∉ E′ then
18     x13 := y
19     for i = 12 to 1 step −1 do
20       yi := xi+1 ⊕ k
21       xi := πi(−, yi)
22     x := x1 ⊕ k
23     E′(−, k, y) := x
24     E′(+, k, x) := y
25   return E′(−, k, y)
```

public procedure $\texttt{Check}(k, x, y)$
  if $(+, k, x) \in E'$ **then return** $E'(+, k, x) = y$
  if $(-, k, y) \in E'$ **then return** $E'(-, k, y) = x$
  return false

4.4 Stages of the Indifferentiability Proof

In order to prove our indifferentiability result, we fix a deterministic, computationally unbounded distinguisher $\mathcal{D}$ issuing at most q queries in total, and prove the following three inequalities:

$$\left|\Pr\left[\mathcal{D}^{\Sigma_1} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_2(\eta,\varphi)} = 1\right]\right| \leq \frac{2^{22} \times q^{12}}{2^n} \qquad \text{(Lemma 5)}$$

$$\left|\Pr\left[\mathcal{D}^{\Sigma_2(\eta,\varphi)} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_3(\pi)} = 1\right]\right| \leq \frac{2^{89} \times q^{12}}{2^n} \qquad \text{(Lemma 23)}$$

$$\left|\Pr\left[\mathcal{D}^{\Sigma_3(\pi)} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_4} = 1\right]\right| \leq \frac{2^{89} \times q^{12}}{2^n} \qquad \text{(Lemma 24)} \ ,$$

where the probabilities are over the random coins of $\mathcal{S}$ and the random draw of $\boldsymbol{E}$ in Σ_1, over the random draw of (η, φ) in Σ_2 (note that $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{F}$ are all deterministic), over the random draw of π in Σ_3, and over the random draw of $\boldsymbol{P}$ in Σ_4.

Additionally, we will show (Lemma 6) that the simulator $\mathcal{S}$ runs in polynomial time with overwhelming probability in system Σ_1. Combining these lemmas will yield Theorem 2, that we restate here for convenience.

Theorem. *For any q, the 12-round single-key iterated Even-Mansour cipher $\mathcal{C}_{12}^{\boldsymbol{P},\gamma}$ with twelve independent random n-bit permutations $\boldsymbol{P} = (\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{12})$, and fixed, efficiently invertible n-bit permutations $\gamma = (\gamma_0, \ldots, \gamma_{12})$ for the key schedule, is strongly and statistically (q, σ, ε)-indifferentiable from an ideal cipher $\boldsymbol{E}$ with n-bit blocks and n-bit keys, where:*

$$\sigma = 2^7 \times q^4 \quad \textit{and} \quad \varepsilon = \frac{2^{91} \times q^{12}}{2^n} \ .$$

Proof. This follows directly from Lemmas 5, 6, 23, and 24 which will be proven in the remaining of the paper. □

4.5 Complexity of the Simulator in the Second System

In this section, we show that the simulator $\mathcal{T}$ runs in polynomial time with probability 1 in system Σ_2. This will imply that $\mathcal{S}$ runs in polynomial time with overwhelming probability in system Σ_1 as we will show later. First, we prove that if the simulator $\mathcal{T}$ ever completes a partial chain $(y_0, x_1, 0)$, then the distinguisher must have made a corresponding call to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$.

Lemma 1. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ where the distinguisher makes at most q queries in total. Then the simulator dequeues at most q times a partial chain of the form $(y_0, x_1, 0, \ell)$ for which $(y_0, x_1, 0) \notin$ COMPLETED.*

Proof. First, note that if the same chain $(y_0, x_1, 0)$ is enqueued twice or more, it will be dequeued only once since it is put into COMPLETED once it has been completed the first time.

Let $(y_0, x_1, 0)$ and $(y'_0, x'_1, 0)$ be two distinct partial chains dequeued at some point in the execution such that $(y_0, x_1, 0) \notin$ COMPLETED and $(y'_0, x'_1, 0) \notin$ COMPLETED when they are dequeued. Let also $\texttt{Check}(k, y_0, x_{13})$ and $\texttt{Check}(k', y'_0, x'_{13})$ be the respective corresponding calls at line 46 or 53 of $\mathcal{T}$ which returned true and led to these chains being enqueued. Then $k = y_0 \oplus x_1$ and $k' = y'_0 \oplus x'_1$, so that these calls were necessarily for distinct triples of inputs. Hence, for each chain $(y_0, x_1, 0, \ell)$ which is dequeued at some point in the execution such that $(y_0, x_1, 0) \notin$ COMPLETED when it is dequeued, we can find a distinct 3-tuple (k, x, y) for which $\texttt{Check}(k, x, y)$ was true at the moment $(y_0, x_1, 0, \ell)$ was enqueued. We can now find a unique access to η during a call to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$ which corresponds to (k, x, y): either $y := \eta(+, k, x)$ if this was a query to $\mathcal{F}.\texttt{Enc}$, or $x := \eta(-, k, y)$ if this was a query to $\mathcal{F}.\texttt{Dec}$. This query to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$ was made either by the distinguisher or the simulator. We argue that this call cannot have been made by the simulator. The simulator issues such queries only when it completes a chain, and after this completion, it adds $(y_0, x_1, 0)$ to COMPLETED, and so it cannot have been that $(y_0, x_1, 0) \notin$ COMPLETED when it was dequeued. Thus, there is a distinct query of the distinguisher associated with each dequeue call of the form $(y_0, x_1, 0, \ell)$ for which $(y_0, x_1, 0) \notin$ COMPLETED, so there are at most q such dequeue calls. □

Lemma 2. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ where the distinguisher makes at most q queries in total. Then:*

- *the size of P_1^+, P_1^-, ..., P_{12}^+, P_{12}^-, E^+, and E^- is at most $6q^2$;*
- *the simulator makes at most $5q^2$ queries to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$, and at most $2 \times 6^3 \times q^6$ queries to $\mathcal{F}.\texttt{Check}$.*

Proof. We first show that $|P_6^+|$, $|P_6^-|$, $|P_7^+|$, and $|P_7^-|$ are at most $2q$. Pairs of assignments in P_6 or P_7 can only happen in two cases: either when the distinguisher makes a query to $\mathcal{T}.\texttt{Query}(i, \delta, z)$ with $i = 6$ or 7, or when the simulator completes a chain $(y_0, x_1, 0, \ell)$. There can be at most q calls to $\mathcal{T}.\texttt{Query}$ by the distinguisher, and according to Lemma 1, there are at most q chains of the form $(y_0, x_1, 0, \ell)$ which are completed, which implies the bound. This directly implies that the simulator completes at most $|P_6^-| \cdot |P_7^+| \leq 4q^2$ chains of the form $(y_6, x_7, 6, \ell)$.

For any $i \in \{1, \ldots, 12\}$, the size of P_i^+ and P_i^- can only be enlarged by at most 1 in the following cases: if the distinguisher calls $\mathcal{T}.\texttt{Query}(i, \delta, z)$, if a chain of the form $(y_0, x_1, 0, \ell)$ is completed, or if a chain $(y_6, x_7, 6, \ell)$ is completed. There are at most q events of the first kind, at most q events of the second kind by Lemma 1, and at most $4q^2$ events of the last kind, giving a total of $4q^2 + 2q \leq 6q^2$ possibilities.

The simulator queries $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$ only when it completes a chain. Hence the total number of queries to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$ made by the simulator is at most $q + 4q^2 \leq 5q^2$. Consequently the total number of queries to $\mathcal{F}.\texttt{Enc}$ or $\mathcal{F}.\texttt{Dec}$ made either by the distinguisher or the simulator is at most $q + 5q^2 \leq 6q^2$. Since the size of E^+ and E^- is enlarged at most by 1 by each query, one has that $|E^+|$ and $|E^-|$ are at most $6q^2$. Finally, the number of queries to $\mathcal{F}.\texttt{Check}$ made by the simulator is bounded by $|P_1^-| \times |P_2^+| \times |P_{12}^-| + |P_1^+| \times |P_{11}^-| \times |P_{12}^+| \leq 2 \times (6q^2)^3$. □

4.6 From the First to the Second System

We show that systems Σ_1 and Σ_2 are indistinguishable. For this, we first recall a result from [HKT11]. Consider the following two systems:

- an n-bit two-sided random function $\mathcal{R}$, which is defined exactly as the two-sided keyed random function $\mathcal{F}$, with a key space of size one (the key is therefore omitted from the inputs to $\mathcal{R}$)
- an n-bit random permutation $\widetilde{\boldsymbol{P}}$, enhanced with a procedure $\texttt{Check}(x, y)$ which returns **true** if and only if $\widetilde{\boldsymbol{P}}(x) = y$.

Then the following result states that these two systems are indistinguishable.

Lemma 3 (Lemma 3.2 of [HKT11]). *A distinguisher which issues at most q' queries to either a random permutation $\widetilde{\boldsymbol{P}}$ enhanced with a procedure* `Check` *as described above or a two-sided random function $\mathcal{R}$ has advantage at most $\frac{6q'^2}{2^n}$ in distinguishing the two systems.*

The following lemma is simply a "keyed" version of Lemma 3.

Lemma 4. *Let $\widetilde{\boldsymbol{E}}$ denote an ideal cipher enhanced with a procedure $\texttt{Check}(k, x, y)$ which returns **true** if and only if $\widetilde{\boldsymbol{E}}(k, x) = y$. Then a distinguisher which issues at most q' queries to either $\widetilde{\boldsymbol{E}}$ or a two-sided keyed random function $\mathcal{F}$ has advantage at most $\frac{6q'^2}{2^n}$ in distinguishing the two systems.*

Proof. We only sketch how to adapt the proof of Lemma 3 given in [HKT11]. Their analysis is based on the definition of bad events such that if these bad events do not occur, the two systems $\widetilde{\boldsymbol{P}}$ and $\mathcal{R}$ behave identically. The probability of these bad events is then upper bounded by $6q'^2/2^n$. In order to adapt the proof to our case, one can define the same bad events for each key k_i queried by the distinguisher. The probability that one of these bad events happens is then upper bounded by the sum over the keys queried by the distinguisher of $6(q'_i)^2/2^n$, where q'_i is the number of queries corresponding to key k_i. The result then follows from $\sum 6(q'_i)^2/2^n \leq 6(\sum q'_i)^2/2^n \leq 6q'^2/2^n$. □

Lemma 5. *For any distinguisher $\mathcal{D}$ which makes at most q queries in total, we have:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_1} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_2(\eta,\varphi)} = 1\right]\right| \leq \frac{2^{22} \times q^{12}}{2^n} \ .$$

Proof. Consider the following simple modification to the simulator $\mathcal{T}$: whenever $\mathcal{T}$ is about dequeuing a $(q+1)$-th partial chain of the form $(y_0, x_1, 0, \ell)$ for which $(y_0, x_1, 0) \notin \textsc{Completed}$, it aborts instead. Clearly Lemmas 1 and 2 still hold for this modified simulator. Assume now that in Σ_2 we replace $\mathcal{F}(\eta)$ with an ideal cipher $\widetilde{\boldsymbol{E}}$ enhanced with a `Check` procedure, and denote Σ'_1 the resulting system. Then the combination of $\mathcal{D}$ and $\mathcal{T}$ can be seen as a single distinguisher $\mathcal{D}'$ interacting either with an ideal cipher $\widetilde{\boldsymbol{E}}$ enhanced with a `Check` procedure for Σ'_1, or with a two-sided keyed random function $\mathcal{F}(\eta)$ for Σ_2. Moreover, by Lemma 2, this distinguisher $\mathcal{D}'$ makes at most $q + 5q^2 + 2 \times 6^3 \times q^6 \leq 2^9 \times q^6$ queries to either $\widetilde{\boldsymbol{E}}$ or $\mathcal{F}$. We can then use Lemma 4 with $q' = 2^9 \times q^6$ to get:

$$\left|\Pr\left[\mathcal{D}^{\Sigma'_1} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_2(\eta,\varphi)} = 1\right]\right| \leq \frac{2^{21} \times q^{12}}{2^n} \ .$$

Moreover, this implies that $\mathcal{T}$ aborts with probability at most $2^{21}q^{12}/2^n$ is Σ_1' since otherwise one would obtain a distinguisher contradicting the bound above (recall that $\mathcal{T}$ never aborts in Σ_2 by Lemma 1). Observe now that the only differences between Σ_1 and Σ_1' are the following ones:

- $\mathcal{T}$ queries φ whereas $\mathcal{S}$ draws random values by itself (but this does not change the distribution of the answers of the system);
- $\mathcal{T}$ queries $\mathcal{F}.\texttt{Check}$ whereas $\mathcal{S}$ queries $\boldsymbol{E}$ or $\boldsymbol{E}^{-1}$ and performs the check by itself (but again this does not change the distribution of the answers of the system);
- $\mathcal{T}$ may abort while $\mathcal{S}$ does not.

Hence we see that unless $\mathcal{T}$ aborts in Σ_1', the two systems behave identically, so that:

$$\left|\Pr\left[\mathcal{D}^{\Sigma_1}=1\right]-\Pr\left[\mathcal{D}^{\Sigma_1'}=1\right]\right| \leq \frac{2^{21}\times q^{12}}{2^n} \enspace .$$

The results follows by combining the two inequalities above. □

As a side result, we obtain that the simulator $\mathcal{S}$ runs in polynomial time in system Σ_1 with overwhelming probability.

Lemma 6. *Assume that the distinguisher $\mathcal{D}$ makes at most q queries in total. Then with probability greater than $1-2^{21}\times q^{12}/2^n$ over an execution of $\mathcal{D}^{\Sigma_1}$, the simulator $\mathcal{S}$ makes at most $2^7\times q^4$ queries to $\boldsymbol{E}$ or $\boldsymbol{E}^{-1}$ (assuming $\mathcal{S}$ never repeats a query).*

Proof. We showed in the proof of Lemma 5 that $\mathcal{T}$ aborts over an execution of $\mathcal{D}^{\Sigma_1'}$ with probability at most $2^{21}q^{12}/2^n$. This directly implies that with probability at most $2^{21}q^{12}/2^n$ over an execution of $\mathcal{D}^{\Sigma_1}$, $\mathcal{S}$ dequeues at most q times a partial chain of the form $(y_0,x_1,0,\ell)$ for which $(y_0,x_1,0)\notin \textsc{Completed}$. Whenever this holds, this implies as in the proof of Lemma 2 that the size of P_1^+, P_1^-, …, P_{12}^+, P_{12}^-, E^+, and E^- maintained by $\mathcal{S}$ is at most $6q^2$. Hence, with probability at least $1-2^{21}\times q^{12}/2^n$, $\mathcal{S}$ makes at most $|P_1^-|\cdot|P_2^+|+|P_{11}^-|\cdot|P_{12}^+|\leq 2\times(6q^2)^2\leq 2^7\times q^4$ queries to $\boldsymbol{E}$ or $\boldsymbol{E}^{-1}$ over an execution of $\mathcal{D}^{\Sigma_1}$. □

4.7 From the Second to the Third System

This section contains the heart of the proof, and is largely devoted to the analysis of the second system: we first give some definitions about so-called *partial chains* in $\Sigma_2(\eta,\varphi)$ (Section 4.7.1), then we define a bad event that may happen in $\Sigma_2(\eta,\varphi)$ and show that it occurs only with negligible probability (Section 4.7.2), and then we show that for executions such that this bad event does not happen, the simulator $\mathcal{T}$ never overwrites entries during a call to `ForceVal` (Section 4.7.3). Finally, Section 4.7.4 contains a "randomness mapping" argument (similar to the one of [HKT11]) enabling to precisely compare executions of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ and $\mathcal{D}^{\Sigma_3(\pi)}$.

4.7.1 Partial Chains

A partial chain C is a tuple (y_i,x_{i+1},i) where $y_i,x_{i+1}\in\{0,1\}^n$ and $i\in\{0,\ldots,12\}$. The key associated with $C=(y_i,x_{i+1},i)$ is $k=y_i\oplus x_{i+1}$. Note that partial chains $(y_0,x_1,0)$ and $(y_{12},x_{13},12)$ are special: y_0 (resp. x_{13}) correspond to the direct input (resp. indirect input) to the iterated Even-Mansour cipher.

Fix hash tables $P_1, \ldots, P_{12}$, and E at some point in the execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$. For a chain $C = (y_i, x_{i+1}, i)$ we denote $C[1] = y_i$, $C[2] = x_{i+1}$, and $C[3] = i$. We define below two functions `next` and `prev`, taking as input a partial chain C, and returning the partial chain obtained by moving respectively one step forward or backward in the iterated Even-Mansour cipher, or the special symbol $\perp$ when this is not possible because some value is undefined in tables P_i or E, as well as functions $\texttt{val}_\ell^+$ and $\texttt{val}_\ell^-$, taking as input a partial chain C, and returning respectively the direct or inverse input to permutation P_ℓ corresponding to C, or the special symbol $\perp$ when it is undefined. Note that these are *functions* used for the reasoning, not procedures: they do not modify the tables.

function `next`(y_i, x_{i+1}, i):
  $k := y_i \oplus x_{i+1}$
  if $i < 12$ **then**
    if $(+, x_{i+1}) \notin P_{i+1}$ **then return** $\perp$
    $y_{i+1} = P_{i+1}(+, x_{i+1})$
    $x_{i+2} := y_{i+1} \oplus k$
    return $(y_{i+1}, x_{i+2}, i+1)$
  else if $i = 12$ **then**
    if $(-, k, x_{13}) \notin E$ **then return** $\perp$
    $y_0 := E(-, k, x_{13})$
    $x_1 := y_0 \oplus k$
    return $(y_0, x_1, 0)$

function `prev`(y_i, x_{i+1}, i):
  $k := y_i \oplus x_{i+1}$
  if $i > 0$ **then**
    if $(-, y_i) \notin P_i$ **then return** $\perp$
    $x_i := P_i(-, y_i)$
    $y_{i-1} := x_i \oplus k$
    return $(y_{i-1}, x_i, i-1)$
    else if $i = 0$ **then**
      if $(+, k, y_0) \notin E$ **then return** $\perp$
      $x_{13} := E(+, k, y_0)$
      $y_{12} := x_{13} \oplus k$
      return $(y_{12}, x_{13}, 12)$

function $\texttt{val}_\ell^+(C)$: $\quad \backslash\backslash \ell = 1, \ldots, 13$
  while $(C \neq \perp) \wedge (C[3] \neq \ell - 1)$ **do**
    $C := \texttt{next}(C)$
  if $C = \perp$ **then return** $\perp$
  return $C[2]$

function $\texttt{val}_\ell^-(C)$: $\quad \backslash\backslash \ell = 0, \ldots, 12$
  while $(C \neq \perp) \wedge (C[3] \neq \ell)$ **do**
    $C := \texttt{prev}(C)$
  if $C = \perp$ **then return** $\perp$
  return $C[1]$

We will also need the following two important definitions.

Definition 2. *Consider tables $P_1, \ldots, P_{12}$, and E at some point in the execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$. Two partial chains C and D are* equivalent *(denoted $C \equiv D$) if $C = D$ or if D can be obtained by applying* `next` *or* `prev` *finitely many times on C.*

This relation is clearly always reflexive and transitive, but not necessarily symmetric (as we will see later, it is symmetric as long as no entry is overwritten in any table P_i or E).

Definition 3. *Consider tables $P_1, \ldots, P_{12}$, and E at some point in the execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$. A partial chain $C = (y_i, x_{i+1}, i)$ is said to be* table-defined *if* $\texttt{next}(C) \neq \perp$ *and* $\texttt{prev}(C) \neq \perp$.

The following equivalences can be immediately checked:

- A partial chain $C = (y_i, x_{i+1}, i)$ with $i \in \{1, \ldots, 11\}$ is table-defined iff $(-, y_i) \in P_i$ and $(+, x_{i+1}) \in P_{i+1}$.
- A partial chain $C = (y_0, x_1, 0)$ is table-defined iff $(+, x_1) \in P_1$ and $(+, k, y_0) \in E$ with $k = y_0 \oplus x_1$.
- A partial chain $C = (y_{12}, x_{13}, 12)$ is table-defined iff $(-, y_{12}) \in P_{12}$ and $(-, k, x_{13}) \in E$ with $k = y_{12} \oplus x_{13}$.

4.7.2 Bad Events in the Second System

In all the following, a *pair of random forward assignments in table E* refers to any sequence of instructions $E(+, k, x) := y$, $E(-, k, y) := x$ happening at lines 8-9 of $\mathcal{F}$. A *pair of random backward assignments in table E* refers to any sequence of instructions $E(-, k, y) := x$, $E(+, k, x) := y$ happening at lines 15-16 of $\mathcal{F}$. Note that for a pair of random forward assignments, $y := \eta(+, k, x)$ is freshly taken from table η at line 7, while or a pair of random backward assignments, $x := \eta(-, k, y)$ is freshly taken from table η at line 14.

Similarly, a *pair of random forward assignments in table P_i* refers to any sequence of instructions $P_i(+, x_i) := y_i$, $P_i(-, y_i) := x_i$ happening at lines 34-35 of $\mathcal{T}(\varphi)$. A *pair of random backward assignments in table P_i* refers to any sequence of instructions $P_i(-, y_i) := x_i$, $P_i(+, x_i) := y_i$ happening at lines 34-35 of $\mathcal{T}(\varphi)$. Note that for a pair of random forward assignments, $y_i := \varphi_i(+, x_i)$ is freshly taken from φ at line 33, while for a pair of random backward assignments, $x_i := \varphi_i(-, y_i)$ is freshly taken from φ at line 33.

A *pair of random assignments* in table E or P_i refers indifferently to the forward or backward case. We define the history $\mathcal{H}$ for a pair of random assignments in E or in some P_i as the set of all n-bit strings appearing in the tables E and P_i just before the pair of assignments (more precisely, for any entry $E(\delta, k, z) = z'$, this includes k, z and z', while for any entry $P_i(\delta, z_i) = z_i'$ this includes z_i and z_i') as well as n-bit strings appearing in the *query* corresponding to the pair of assignments, namely the values k and x (resp. k and y) for a pair of random forward (resp. backward) assignments in E, or x_i (resp. y_i) for a pair of random forward (resp. backward) assignments in P_i.

We are now ready to define a bad event[10] that may happen during an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$, more precisely during a pair of random assignments in E or some table P_i.

[10] In a departure from [HKT11], we describe one single and simple event rather than three more complex events.

Definition 4. *We say that event* Bad *happens if for a pair of random assignments in E or in some P_i, the value freshly taken from η or φ_i is equal to the bitwise xor of five or less values of the history $\mathcal{H}$.*

The probability of Bad can be easily upper bounded.

Lemma 7. *When $\mathcal{D}$ makes at most q queries, the probability (over the random choice of η and φ) that event* Bad *occurs in $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ satisfies:*

$$\Pr\left[\mathtt{Bad}\right] \leq \frac{2^{43} \times q^{12}}{2^n} \ .$$

Proof. By Lemma 2, we know that there are at most $6q^2$ pairs of assignments in each tables E and P_i during all the execution. Each pair of assignments in E adds at most three values in the history, while each pair of assignments in P_i adds at most two values. Consequently, the size of the history is always at most $3 \times 6q^2 + 12 \times 2 \times 6q^2 = 162 \times q^2$. Hence, for any pair of random assignments in E or in some P_i, event Bad happens with probability at most $(162 \cdot q^2)^5/2^n$. Since there are at most $13 \times 6q^2$ pairs of random assignments in E or in the P_i's in total, we obtain:

$$\Pr[\mathtt{Bad}] \leq \frac{13 \times 6 \times 162^5 \times q^{12}}{2^n} \leq \frac{2^{43} \times q^{12}}{2^n} \ ,$$

as claimed. □

Definition 5. *A pair (η, φ) is said* good *(with respect to some fixed distinguisher $\mathcal{D}$) if event* Bad *does not occur during an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$.*

Remark 2. In the more general case where efficiently invertible permutations $(\gamma_0, \ldots, \gamma_{12})$ are used for the key derivation, the bad event becomes more cumbersome to describe. However, the same bound as in Lemma 7 holds as well.

4.7.3 Properties of Good Executions

Our goal in this part is to show (this will be Lemma 17) that during an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ), the simulator $\mathcal{T}$ never overwrites values in tables P_i during a call to ForceVal. For this we proceed with a series of intermediate lemmas. We start with two important definitions.

Definition 6. *A call to* $\mathtt{Adapt}(y_{\ell-2}, x_{\ell-1}, y_{\ell+1}, x_{\ell+2}, \ell)$ *is said* safe *if before the call, one has $x_{\ell-1} \notin P^+_{\ell-1}$ and $y_{\ell+1} \notin P^-_{\ell+1}$.*

Definition 7. *A call to* $\mathtt{ForceVal}(x_\ell, y_\ell, \ell)$ *is said* non-overwriting *if before the call, one has $x_\ell \notin P^+_\ell$ and $y_\ell \notin P^-_\ell$.*

We first state some basic properties of good pairs (η, φ) in Lemmas 8, 9, and 10 below. They are very similar to Lemmas 3.17, 3.20, 3.21, and 3.22 of [HKT11], but note that our Lemmas 9 and 10 hold assuming that all calls to ForceVal were non-overwriting up to some point in the execution.

Lemma 8. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Then the following properties hold:*

(a) No entry is ever overwritten during a random pair of assignments in E or in some P_i.
(b) For any partial chain C, if $\texttt{next}(C) = \perp$ before a pair of random assignments in E or in some P_i, then if C is table-defined after the pair of random assignments, it holds that $\texttt{next}(\texttt{next}(C)) = \perp$.
(c) For any partial chain C, if $\texttt{prev}(C) = \perp$ before a pair of random assignments in E or in some P_i, then if C is table-defined after the pair of random assignments, it holds that $\texttt{prev}(\texttt{prev}(C)) = \perp$.
(d) For any partial chain C, and each $\delta \in \{+, -\}$, there is at most one value $\ell \in \{0, \ldots, 13\}$ such that $\texttt{val}_\ell^\delta(C)$ may change during a pair of random assignments in E or in some P_i. More precisely, for a pair of random assignments in E, only $\texttt{val}_1^+(C)$ or $\texttt{val}_{12}^-(C)$ can change, while for a pair of random forward assignments $P_i(+, x_i) := y_i$, $P_i(-, y_i) := x_i$, or a pair of random backward assignments $P_i(-, y_i) := x_i$, $P_i(+, x_i) := y_i$, only $\texttt{val}_{i+1}^+(C)$ can change (in which case $\texttt{val}_i^+(C) = x_i$ before the pair of assignments) or $\texttt{val}_{i-1}^-(C)$ can change (in which case $\texttt{val}_i^-(C) = y_i$ before the pair of random assignments). Moreover, if $\texttt{val}_\ell^\delta(C)$ changes, then it necessarily changes from $\perp$ to some value in $\{0,1\}^n$, and if C is table-defined after the pair of assignments, then $\texttt{val}_\ell^\delta(C) \notin P_\ell^\delta$.
(e) For any partial chain C, if C is table-defined before a pair of random assignments in some P_i, then:
  – if this is a pair of random forward *assignments, $\texttt{val}_\ell^-(C)$ remains constant during the pair of assignments for any $\ell \in \{0, \ldots, 12\}$;*
  – if this is a pair of random backward *assignments, $\texttt{val}_\ell^+(C)$ remains constant during the pair of assignments for any $\ell \in \{1, \ldots, 13\}$.*

Proof. To show (a), observe that for an entry to be overwritten during a pair of random assignments in E or in some P_i, the value freshly taken from η or φ_i must already be in the history $\mathcal{H}$, contradicting the assumption that `Bad` does not happen.

We now show (b). Let $C = (y_i, x_{i+1}, i)$ be a partial chain, and assume that $\texttt{next}(C) = \perp$ before a pair of random assignments in E or in some P_i, and that C is table-defined and $\texttt{next}(\texttt{next}(C)) \neq \perp$ after the pair of assignments. We distinguish two cases depending on i. Assume first that $i = 12$, so that $C = (y_{12}, x_{13}, 12)$. Since $\texttt{next}(C) = \perp$ before the pair of assignments and $\texttt{next}(\texttt{next}(C)) \neq \perp$ after the pair of assignments, we see that it was necessarily a pair of assignments in E: either a pair of random forward assignments $E(+, k, y_0) := x_{13}$, $E(-, k, x_{13}) := y_0$, or a pair of random backward assignments $E(-, k, x_{13}) := y_0$, $E(+, k, y_0) := x_{13}$, with $k = y_{12} \oplus x_{13}$. Moreover, since C is table-defined after the pair of assignments, we have that $y_{12} \in P_{12}^-$ before the pair of assignments. Hence, if this was a pair of random forward assignments, the value freshly taken from η was $x_{13} := \eta(+, k, y_0)$, and we see that $x_{13} = k \oplus y_{12}$ is the xor of two values of the history $\mathcal{H}$, contradicting the assumption that `Bad` does not happen. If this was a pair of random backward assignments, then the value freshly taken from η was $y_0 := \eta(-, k, x_{13})$. Denote $x_1 = y_0 \oplus k$, so that $\texttt{next}(C) = (y_0, x_1, 0)$. Since by assumption $\texttt{next}(\texttt{next}(C)) \neq \perp$, $x_1 \in P_1^+$ even before the pair of random assignments. Hence we see that $y_0 = x_1 \oplus k$ is the xor of two values of the history $\mathcal{H}$, contradicting the assumption that `Bad` does not happen.

Assume now that $C = (y_i, x_{i+1}, i)$ with $i < 12$. Since $\texttt{next}(C) = \perp$ before the pair of assignments and $\texttt{next}(\texttt{next}(C)) \neq \perp$ after the pair of assignments, we see that it was necessarily a

pair of assignments in P_{i+1}: either a pair of random forward assignments $P_{i+1}(+, x_{i+1}) := y_{i+1}$, $P_{i+1}(-, y_{i+1}) := x_{i+1}$, or a pair of random backward assignments $P_{i+1}(-, y_{i+1}) := x_{i+1}$, $P_{i+1}(+, x_{i+1}) := y_{i+1}$. Moreover, since C is table-defined after the pair of assignments, we have that $y_i \in P_i^-$ before the pair of assignments. Denote $k = y_i \oplus x_{i+1}$, and $x_{i+2} = y_{i+1} \oplus k$. Then $\texttt{next}(C) = (y_{i+1}, x_{i+2}, i+1)$, and since $\texttt{next}(\texttt{next}(C)) \neq \bot$, we have $x_{i+2} \in P_{i+2}^+$ before the pair of assignments when $i < 11$, or $(-, k, x_{13}) \in E$ before the pair of assignments when $i = 11$. Hence if this was a pair of random forward assignments, the value freshly taken from φ_{i+1} was $y_{i+1} := \varphi_{i+1}(+, x_{i+1})$, and we see that $y_{i+1} = x_{i+2} \oplus y_i \oplus x_{i+1}$ is the xor of three values in the history $\mathcal{H}$, contradicting the assumption that `Bad` does not happen. If this was a pair of random backward assignments, the value freshly taken from φ_{i+1} was $x_{i+1} := \varphi_{i+1}(-, y_{i+1})$, and we see that $x_{i+1} = y_i \oplus y_{i+1} \oplus x_{i+2}$ is the xor of three values in the history $\mathcal{H}$, contradicting the assumption that `Bad` does not happen. This concludes the proof of (b).

The proof of (c) is similar to (b) by symmetry. The proof of (d) follows the same lines as the proof of (b) and (c) and is omitted.

We finally show (e). We consider the case of a pair of random forward assignments $P_i(+, x_i) := y_i$, $P_i(-, y_i) := x_i$ (the case of a pair of backward assignments is similar). For any $\ell \in \{0, \dots 12\}$, $\texttt{val}_\ell^-(C)$ can change during this pair of assignment only if $\texttt{val}_i^-(C) = y_i$. Note that it cannot be that $C = (y_i, x_{i+1}, i)$ for some x_{i+1} because of the assumption that C is table-defined before the pair of assignments. We distinguish two cases depending on i. Assume first that $i = 12$. Denote $(y_0, x_1, 0)$ the chain obtained by a sufficient number of applications of `prev` to C (possibly none), and $k = y_0 \oplus x_1$. Since C is table-defined before the pair of assignments and $\texttt{val}_{12}^-(C) = y_{12} \neq \bot$, it holds that $(+, k, y_0) \in E$ and $x_1 \in P_1^+$ before the pair of assignments. Denote $x_{13} = E(+, k, y_0)$. The value freshly taken from φ during the pair of forward assignments is y_{12}, and we see that $y_{12} = x_{13} \oplus (y_0 \oplus x_1)$ is the xor of three values in the history, contradicting the assumption that `Bad` does not happen. Assume now that $i < 12$, and denote $(y_{i+1}, x_{i+2}, i+1)$ the chain obtained by a sufficient number of applications of `prev` to C (possibly none), and $k = y_{i+1} \oplus x_{i+2}$. Since C is table-defined before the pair of assignments and $\texttt{val}_i^-(C) = y_i \neq \bot$, it holds that $y_{i+1} \in P_{i+1}^-$ and $x_{i+2} \in P_{i+2}^+$ when $i < 11$, or $(-, k, x_{13}) \in E$ when $i = 11$. Denote $x_{i+1} = P_{i+1}(-, y_{i+1})$. Then the value freshly taken from φ is y_i, and we see that $y_i = x_{i+1} \oplus (y_{i+1} \oplus x_{i+2})$ is the xor of three values in the history, again contradicting the assumption that `Bad` does not happen. □

Lemma 9. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Then at any point in the execution such that all calls to* `ForceVal` *were non-overwriting up to that point, the following properties hold:*

(a) For all partial chains C and D, $\texttt{next}(C) = D \Leftrightarrow \texttt{prev}(D) = C$.
(b) The relation $\equiv$ between partial chains is an equivalence relation.
(c) If two table-defined partial chains C and D are equivalent at this point in the execution, then there exists a sequence $C_1, \dots, C_r$ ($r \geq 1$) of table-defined partial chains such that:
  – ($C = C_1$ and $D = C_r$) or ($C = C_r$ and $D = C_1$);
  – $C_i = \texttt{next}(C_{i-1})$ and $C_{i-1} = \texttt{prev}(C_i)$.

Proof. Note that for a good pair (η, φ), as long as all calls to `ForceVal` are non-overwriting, no entries are ever overwritten in table E or in any table P_i by Lemma 8 (a). Hence all tables P_i define partial permutations (recall that this means that $(+, x_i) \in P_i$ and $P_i(+, x_i) = y_i$ if

and only if $(-, y_i) \in P_i$ and $P_i(-, y_i) = x_i$), and E defines a partial cipher (recall that this means that $(+, k, x) \in E$ and $E(+, k, x) = y$ if and only if $(-, k, y) \in E$ and $E(-, k, y) = x$). This implies statement (a).

The relation $\equiv$ is always reflexive and transitive by definition, and it follows from (a) that it is symmetric (and hence an equivalence relation) for a good pair (η, φ) as long as all calls to `ForceVal` are non-overwriting.

Finally, if $C \equiv D$, D can be obtained by applying `next` and `prev` finitely many times on C. Any shortest such sequence can only consist of applications of `next` or `prev` because of (a). Assume that a shortest sequence consists only of $r - 1$ ($r \geq 1$) applications of `next` (the reasoning is similar for `prev`). Then denoting $C_1 = C$, $C_2 = \texttt{next}(C_1)$, $\ldots$, $C_r = \texttt{next}(C_{r-1})$, we see that $C_r = D$, and that all partial chains C_i are table-defined since $\texttt{next}(C_i) \neq \bot$ and $\texttt{prev}(C_i) \neq \bot$ by (a). This shows (c). □

Lemma 10. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Let C and D be two table-defined chains at some point in the execution such that all calls to* `ForceVal` *were non-overwriting up to that point. Assume that after this point, there is a pair of random assignments in some table P_i or in E. Then $C \equiv D$ before the pair of assignments if and only if $C \equiv D$ after the pair of assignments.*

Proof. Assume that $C \equiv D$ before the pair of random assignments, and let $C_1, \ldots, C_r$ be a sequence whose existence is ensured by Lemma 9 (c). Since no entries are overwritten in E or in any P_i during the pair of random assignments by Lemma 8 (a), the same sequence $C_1, \ldots, C_r$ shows that $C \equiv D$ after the pair of random assignments.

Assume now that $C \equiv D$ after the pair of random assignments, and again let $C_1, \ldots, C_r$ be a sequence whose existence is ensured by Lemma 9 (c). Assume that $C_1 = C$ and $C_r = D$ (the reasoning is similar for the case $C_1 = D$ and $C_r = C$). Assume towards a contradiction that before the pair of random assignments, $r - 1$ applications of `next` to C do not yield D. It is not possible that this yields a partial chain $D' \neq \bot$ distinct from D since this would imply that the pair of random assignments overwrote an entry in a table, which does not happen for a good execution. This means that before the pair of random assignments, there is some j such that $\texttt{next}(C_j) = \bot$. Note that it is not possible that $C_j = C_r$ since $C_r = D$ is table-defined before the pair of random assignments. Hence we see that $\texttt{next}(\texttt{next}(C_j)) \neq \bot$ after the pair of random assignments, contradicting Lemma 8 (b). □

Two table-defined non-equivalent chains C and D may *collide* at round ℓ, meaning that $\texttt{val}^{\delta}_{\ell}(C) = \texttt{val}^{\delta}_{\ell}(D) \neq \bot$ for $\delta \in \{+, -\}$. However, this cannot happen unexpectedly for a good execution. This is captured by the following lemma.

Lemma 11. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Fix a point in the execution and assume that all calls to* `ForceVal` *were non-overwriting up to that point. Assume that immediately after this point, a pair of random assignments occurs in some table P_i. Then for any two partial chains C and D, any $\ell \in \{1, \ldots, 12\}$, and any $\delta \in \{+, -\}$, the following conditions cannot be simultaneously fulfilled:*

(1) before the pair of assignments, C and D are not equivalent;
(2) before the pair of assignments, $\texttt{val}^{\delta}_{\ell}(C) = \bot$ or $\texttt{val}^{\delta}_{\ell}(D) = \bot$;
(3) after the pair of assignments, C and D are table-defined;
(4) after the pair of assignments, $\texttt{val}^{\delta}_{\ell}(C) = \texttt{val}^{\delta}_{\ell}(D) \neq \bot$.

Proof. Assume towards a contradiction that all conditions happen during a pair of random assignments in table P_i for two partial chains C and D, some $\ell \in \{1, \ldots, 12\}$, and $\delta = +$ (the case $\delta = -$ is similar). Denote x_i and y_i the values involved in the pair of random assignments in P_i (we consider indifferently a forward or backward pair of assignments). Assume *wlog* that $\texttt{val}_\ell^+(C) = \perp$ before the pair of random assignments.

First, the fact that $\texttt{val}_\ell^+(C)$ changes from $\perp$ to a different value implies by Lemma 8 (d) that $i = \ell - 1$ (and hence $\ell \geq 2$) and $\texttt{val}_i^+(C) = x_i$ before the pair of random assignments. We write all indexes as functions of i in the remaining of the proof. In particular, we have that before the pair of assignments, $\texttt{val}_{i+1}^+(C) = \perp$, and after the pair of assignments, $\texttt{val}_{i+1}^+(C) = \texttt{val}_{i+1}^+(D) \neq \perp$.

We now distinguish two cases depending on $\texttt{val}_{i+1}^+(D)$ before the pair of assignments. First, assume that $\texttt{val}_{i+1}^+(D) = \perp$ before the pair of assignments. Exactly as for C, it must be that $\texttt{val}_i^+(D) = x_i$ before the pair of assignments. Denote $(y_{i-1}, x_i, i-1)$ the chain obtained by a sufficient number of application of `next` to C, and $(y'_{i-1}, x_i, i-1)$ the chain obtained by a sufficient number of application of `next` to D (possibly none in both cases). Then it must hold that $y_{i-1} \neq y'_{i-1}$. Otherwise, denoting $B = (y_{i-1}, x_i, i-1)$ one would have that $C \equiv B$ and $D \equiv B$ before the pair of assignments. Since we assumed that all calls to `ForceVal` were non-overwriting up to that point, by Lemma 9 (b) it holds that $C \equiv D$ before the pair of assignments, a contradiction with condition (1). Hence $y_{i-1} \neq y'_{i-1}$, so that $\texttt{val}_{i+1}^+(C) \neq \texttt{val}_{i+1}^+(D)$ after the pair of assignments, a contradiction with condition (4).

Assume now that $\texttt{val}_{i+1}^+(D) \neq \perp$ before the pair of assignments. Denote $(y_{i-1}, x_i, i-1)$ the chain obtained by a sufficient number of application of `next` to C, and $(y'_{i-1}, x'_i, i-1)$ the chain obtained by a sufficient number of application of `next` to D (possibly none in both cases). The same reasoning as above shows that one cannot have $x'_i = x_i$, hence we can assume $x'_i \neq x_i$. By condition (3) that D is table-defined after the pair of random assignments, D must be table-defined even before the pair of random assignments we are considering, so that $y'_{i-1} \in P_{i-1}^-$ and $x'_i \in P_i^+$ before the pair of random assignments. This allows us to consider the value $y'_i = P_i(+, x'_i)$ and we have $y'_i \in P_i^-$ before the pair of assignments. Similarly, $y_{i-1} \in P_{i-1}^-$ since C is table-defined after the pair of random assignments. After the pair of random assignments, we have $\texttt{val}_{i+1}^+(C) = \texttt{val}_{i+1}^+(D)$, which means $y_i \oplus y_{i-1} \oplus x_i = y'_i \oplus y'_{i-1} \oplus x'_i$, or equivalently $y_i \oplus x_i = y'_i \oplus y_{i-1} \oplus y'_{i-1} \oplus x'_i$, where y'_i, y_{i-1}, y'_{i-1}, and x'_i are all in the history $\mathcal{H}$. Hence, independently of whether this is a pair of forward or backward assignment, the value freshly taken from φ_i (either y_i for a pair of forward assignments or x_i for a pair of backward assignments) is the xor of five values of the history, a contradiction with the assumption that `Bad` does not happen.

Having excluded all possibilities, we conclude that the four conditions cannot be simultaneously fulfilled. □

We now state some properties of safe calls to `Adapt`.

Lemma 12. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Consider a safe call* $\texttt{Adapt}(y_{\ell-2}, x_{\ell-1}, y_{\ell+1}, x_{\ell+2}, \ell)$*, and assume that all previous calls to* `ForceVal` *were non-overwriting. Then the following properties hold:*

(a) The resulting call to $\texttt{ForceVal}(x_\ell, y_\ell, \ell)$ *is non-overwriting.*

(b) If a chain C is table-defined before the call to `Adapt` *and is not equivalent to the chain which is being completed, then for any $i \in \{1, \ldots, 12\}$, $\texttt{val}_i^+(C)$ and $\texttt{val}_i^-(C)$ remain constant during the call to* `ForceVal`*.*

(c) If two chains C and D are table-defined before the call to `Adapt`*, then $C \equiv D$ after the call to* `ForceVal` *if and only if $C \equiv D$ before the call to* `ForceVal`*.*

Proof. We first prove (a). Since $x_{\ell-1} \notin P^+_{\ell-1}$, a pair of random forward assignments occurs in $P_{\ell-1}$, and this cannot lead to $x_\ell \in P^+_\ell$ since this would contradict Lemma 8 (b) for the partial chain $(y_{\ell-2}, x_{\ell-1}, \ell-2)$. The reasoning is similar for y_ℓ, and this shows that the call to `ForceVal` is non-overwriting.

We then show (b). Consider a chain C which is table-defined before the call to `Adapt`. Denote $B = (y_{\ell-2}, x_{\ell-1}, \ell-2)$ and $A = (y_{\ell+1}, x_{\ell+2}, \ell+1)$. Note that when the call to `Adapt` occurs, $A \equiv B$ since A and B are both equivalent to the chain which is being completed. If C is equivalent to A and B before the call to `Adapt`, then C is equivalent to the chain which is being completed since $\equiv$ is an equivalence relation by Lemma 9 (b). Assume now that C is not equivalent to A and B. We show that for any $i \in \{1, \ldots, 12\}$, $\mathtt{val}^+_i(C)$ and $\mathtt{val}^-_i(C)$ remain constant during the call to `ForceVal`. Assume that $\mathtt{val}^+_i(C)$ does not remain constant during the call to `ForceVal` for some i (the reasoning for $\mathtt{val}^-_i(C)$ is similar). The value of $\mathtt{val}^+_i(C)$ can only change during $\mathtt{ForceVal}(x_\ell, y_\ell, \ell)$ if $\mathtt{val}^+_\ell(C) = x_\ell$. Consider the pair of random forward assignments to table $P_{\ell-1}$ just preceding the call to `ForceVal`. Then the following conditions are simultaneously fulfilled:

(1) before the pair of assignments, B and C are not equivalent (by assumption);
(2) before the pair of assignments, $\mathtt{val}^+_\ell(B) = \bot$ since $x_{\ell-1} \notin P^+_{\ell-1}$;
(3) after the pair of assignments, B and C are both table-defined;
(4) after the pair of assignments, $\mathtt{val}^+_\ell(B) = \mathtt{val}^+_\ell(C) = x_\ell$.

But this is impossible for a good execution by Lemma 11.

Finally, we show (c). Let C and D be two chains which are table-defined before the call to `Adapt`. Assume that $C \equiv D$ before the call to `ForceVal`. Let $C_1, \ldots, C_r$ be a sequence whose existence is ensured by Lemma 9 (c). Since the call to `ForceVal` is non-overwriting by (a), the same sequence still implies that $C \equiv D$ after the call to `ForceVal`. Assume now that $C \equiv D$ after the call to `ForceVal`. Again, let $C_1, \ldots, C_r$ be a sequence whose existence is ensured by Lemma 9 (c). Assume that $C_1 = C$ and $C_r = D$ (the reasoning is similar for the case $C_1 = D$ and $C_r = C$). Assume towards a contradiction that before the call to `ForceVal`, $r-1$ applications of `next` to C do not yield D. It is not possible that this yields a partial chain $D' \neq \bot$ distinct from D since the call to `ForceVal` is non-overwriting by (a). This means that before the pair of random assignments, there is some j such that $\mathtt{next}(C_j) = \bot$. Note that it is not possible that $C_j = C_r$ since $C_r = D$ is table-defined before the call to `Adapt`. We can now distinguish two cases. Assume first that C is equivalent to A and B before the call to `Adapt`. Then D is equivalent to A, B and C after the call to `ForceVal`, but this implies (since D was already table-defined before the call to `Adapt`) that D was already equivalent to A and B before the call to `Adapt`, and hence equivalent to C as well. The remaining case is if C is not equivalent to A and B before the call to `Adapt`. But then the fact that $\mathtt{next}(C_j) = \bot$ before the call to `ForceVal` and $\mathtt{next}(C_j) \neq \bot$ after the call implies that some value $\mathtt{val}^+_i(C)$ changes during the call to `ForceVal`, which contradicts (b). □

The following two lemmas will be helpful for the remaining of the reasoning.

Lemma 13. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Assume that at some point, a chain C is dequeued such that $C \notin$* Completed *and that all calls to* `Adapt` *before C*

is dequeued were safe. Then when C was enqueued, no equivalent chain had been previously enqueued.

Proof. Assume that this is false, and consider a chain C which is dequeued at some point such that $C \notin$ COMPLETED, all calls to `Adapt` before C is dequeued were safe, and at the moment C was enqueued, some chain D equivalent to C had been previously enqueued. Then by Lemmas 10 and 12 (c) and the assumption that all calls to `Adapt` are safe (and hence all calls to `ForceVal` are non-overwriting by Lemma 12 (a)) before C is dequeued, C and D remain equivalent until the moment where C is dequeued. Clearly, D has already been dequeued when C is dequeued. Since C was already equivalent to D at that time, we see that either $C = (y_0, x_1, 0)$ or $C = (y_6, x_7, 6)$, where (y_0, x_1) and (y_6, x_7) are the values returned respectively by $\texttt{EvalBackward}(D, 0)$ and $\texttt{EvalForward}(D, 7)$ at lines 16 and 17 of $\mathcal{T}$, so that C was added to the set COMPLETED upon completion of chain D. This is in contradiction with the assumption that $C \notin$ COMPLETED when C is dequeued. □

Lemma 14. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Assume that at some point, a chain C is dequeued, to be adapted at position ℓ, such that $C \notin$ COMPLETED. Then if $\texttt{val}^+_{\ell-1}(C) \notin P^+_{\ell-1}$ and $\texttt{val}^-_{\ell+1}(C) \notin P^-_{\ell+1}$ when C is dequeued, the resulting call to `Adapt` is safe.*

Proof. Denote $\texttt{Adapt}(y_{\ell-2}, x_{\ell-1}, y_{\ell+1}, x_{\ell+2}, \ell)$ the call resulting from the completion of C. We want to show that this call is safe, *i.e.* $x_{\ell-1} \notin P^+_{\ell-1}$ and $y_{\ell+1} \notin P^-_{\ell+1}$ before the call. We show this for $x_{\ell-1}$ (the reasoning is similar for $y_{\ell+1}$). Assume first that $\texttt{val}^+_{\ell-1}(C) \neq \perp$ when C is dequeued. Then clearly $x_{\ell-1} = \texttt{val}^+_{\ell-1}(C)$ and hence $x_{\ell-1} \notin P^+_{\ell-1}$ when `Adapt` is called. Assume now that $\texttt{val}^+_{\ell-1}(C) = \perp$ when C is dequeued. Then by Lemma 8 (d), $\texttt{val}^+_{\ell-1}(C)$ can only change from $\perp$ to some value in $\{0,1\}^n$ during the pair of random assignments in $P_{\ell-2}$ occurring while procedure $\texttt{EvalForward}(C, \ell-1)$ is executed, and since C is table-defined, $\texttt{val}^+_{\ell-1}(C) \notin P^+_{\ell-1}$ immediately after this pair of random assignments. Hence, $x_{\ell-1} = \texttt{val}^+_{\ell-1}(C) \notin P^+_{\ell-1}$ when `Adapt` is called. □

The following two lemmas contain the core of the reasoning.

Lemma 15. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Let C be a partial chain which is enqueued at some point in the execution to be adapted at position ℓ. Assume that when C is enqueued, no equivalent chain has been previously enqueued. Then before the call to $\texttt{InQuery}(i, \delta, z)$ which led to C being enqueued, one has $\texttt{val}^+_{\ell-1}(C) = \perp$ and $\texttt{val}^-_{\ell+1}(C) = \perp$.*

Proof. We consider only the case $\ell = 4$, the case $\ell = 9$ is similar by symmetry.

Consider first the case that $C = (y_0, x_1, 0)$ is enqueued due to a call to $\texttt{InQuery}(2, +, x_2)$. Then clearly before the pair of random assignments triggered by this call, $\texttt{val}^+_3(C) = \perp$ since $\texttt{val}^+_2(C) = x_2$ and $x_2 \notin P^+_2$. Assume now that $\texttt{val}^-_5(C) \neq \perp$ before the call, and denote $y_7 = \texttt{val}^-_7(C)$, $x_7 = P_7(-, y_7)$, $y_6 = \texttt{val}^-_6(C)$, and $x_6 = P_6(-, y_6)$. Then necessarily $(y_6, x_7, 6) \equiv C$ has been previously enqueued due to a call to `InQuery` with one of the four following tuple of arguments: $(6, +, x_6)$, $(6, -, y_6)$, $(7, +, x_7)$, or $(7, -, y_7)$. This contradicts the assumption that no equivalent chain has been previously enqueued.

Consider now the case that $C = (y_6, x_7, 6)$ is enqueued due to call to $\texttt{InQuery}(6, +, x_6)$. Then before the pair of random assignments triggered by this call, it holds that $\texttt{val}^-_5(C) = \perp$

since otherwise entry $P_6(-1, y_6)$ would be overwritten, contradicting Lemma 8 (a). Assume now that $\texttt{val}_3^+(C) \neq \perp$ before the call. Denote $k = y_6 \oplus x_7$. Defining $x_i = \texttt{val}_i^+(C)$ for $i = 11$, 12, 13, 1, and 2, we have that all these values must be different from $\perp$ since otherwise $\texttt{val}_3^+(C) = \perp$, and moreover all the following entries are set in the respective tables: $P_{11}(+, x_{11}) = y_{11}$, $P_{12}(+, x_{12}) = y_{12}$, $E(-, k, x_{13}) = y_0$, $P_1(+, x_1) = y_1$, and $= P_2(+, x_2) = y_2$, as well as the corresponding inverse entries $P_{11}(-, y_{11}) = x_{11}$, ..., $P_2(-, y_2) = x_2$. Consider the last pair of assignments to a table before all these values are defined. This cannot have been a pair of assignments to E, P_1 or P_{12} since otherwise $\texttt{val}_3^+(y_{11}, x_{12}, 11)$ would change from $\perp$ to some value $\neq \perp$ during the pair of assignments, contradicting Lemma 8 (d). This cannot have been a pair of random backward assignments to P_2 or a pair of random forward assignments to P_{11}. Indeed, in the first case, $\texttt{val}_3^+(y_0, x_1, 0)$ would change during the pair of assignments, while in the second case $\texttt{val}_{10}^-(y_{12}, x_{13}, 12)$ would change during the pair of assignments, contradicting Lemma 8 (e). The only remaining possibilities are a pair of random forward assignments to P_2 or a pair of random backward assignments to P_{11}, so that the simulator would have detected and enqueued $(y_0, x_1, 0) \equiv C$, a contradiction.

Finally, assume that $C = (y_6, x_7, 6)$ is enqueued due to a call to $\texttt{InQuery}(6, -, y_6)$. Then clearly before the pair of random assignments triggered by this call, $\texttt{val}_5^-(C) = \perp$ since $y_6 \notin P_6^-$. The reasoning showing that $\texttt{val}_3^+(C) = \perp$ is similar to above. □

Lemma 16. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Then all calls to* `Adapt` *are safe.*

Proof. Assume that this is false, and consider the first call to `Adapt` which is not safe. Let C be the chain during completion of which this call to `Adapt` occurs. Then clearly $C \notin$ Completed when C is dequeued, which implies by Lemma 13 that when C was enqueued, no equivalent chain had been previously enqueued (for this, note that all calls to `Adapt` before C is dequeued were safe by assumption that we are considering the first call which is not safe). Hence, Lemma 15 implies that before the call to `InQuery` which led to C being enqueued, $\texttt{val}_{\ell-1}^+(C) = \perp$ and $\texttt{val}_{\ell+1}^-(C) = \perp$. We show that when C is dequeued, one has $\texttt{val}_{\ell-1}^+(C) \notin P_{\ell-1}^+$ and $\texttt{val}_{\ell+1}^-(C) \notin P_{\ell+1}^-$. This will imply the result by Lemma 14. Assume on the contrary that $\texttt{val}_{\ell-1}^+(C) \in P_{\ell-1}^+$ when C is dequeued (the reasoning is similar for $\texttt{val}_{\ell+1}^-(C)$). This implies in particular that $\texttt{val}_{\ell-1}^+(C)$ changed from $\perp$ to some value $\neq \perp$ after C was enqueued. Consider the last pair of assignments to a table before $\texttt{val}_{\ell-1}^+(C) \neq \perp$ holds. We show that immediately after this pair of assignments, $\texttt{val}_{\ell-1}^+(C) \notin P_{\ell-1}^+$. For this, we exclude all possibilities. First, this cannot have been a pair of random assignments to E or some P_i by Lemma 8 (d) since C is table-defined after this pair of assignments. Second, this cannot have been a previous call to `ForceVal`: indeed, C cannot be equivalent to the chain which is being completed when this call to `ForceVal` occurs (since when C is enqueued, no equivalent chain has been previously enqueued), so that by Lemma 12 (b), $\texttt{val}_i^\delta(C)$ remains constant for all i and δ.

Denote $x_{\ell-1}$ the value of $\texttt{val}_{\ell-1}^+(C)$ immediately after $\texttt{val}_{\ell-1}^+(C) \neq \perp$ holds. Since $x_{\ell-1} \notin P_{\ell-1}^+$ immediately after $\texttt{val}_{\ell-1}^+(C) \neq \perp$ holds, and $x_{\ell-1} \in P_{\ell-1}^+$ when C is dequeued, this necessarily means that this value was added in $P_{\ell-1}$ during completion of another chain D. This chain D was enqueued before C was enqueued, and dequeued before C was dequeued, and while D was completed, a pair of random assignments added $(+, x_{\ell-1})$ to $P_{\ell-1}$. Note that this cannot have been a pair of random *backward* assignments, since otherwise $\texttt{val}_\ell^+(C)$ would

change during the pair of assignments, contradicting Lemma 8 (e). Hence this was a pair of random *forward* assignments, and at the time this pair of assignments occurs, $\mathtt{val}^+_{\ell-1}(C) = \mathtt{val}^+_{\ell-1}(D) = x_{\ell-1}$. Consider the last pair of assignments to a table before $\mathtt{val}^+_{\ell-1}(C) = \mathtt{val}^+_{\ell-1}(D) \neq \bot$ holds. We exclude each possibility in turn.

First, this cannot have been a pair of random assignments to table E since this does not modify the value of $\mathtt{val}^\delta_3(\cdot)$ nor $\mathtt{val}^\delta_9(\cdot)$ for a good execution by Lemma 8 (d).

Second, this cannot have been a pair of random assignments in some table P_i. Indeed, the following conditions would be fulfilled:

(1) Before the pair of assignments, C and D are not equivalent. Indeed, when C is enqueued, no equivalent chain has been previously enqueued, and D cannot become equivalent to C later by Lemmas 10 and 12 (c).
(2) Before the pair of assignments, either $\mathtt{val}^+_{\ell-1}(C) = \bot$ or $\mathtt{val}^+_{\ell-1}(D) = \bot$. Indeed, if $\mathtt{val}^+_{\ell-1}(C) \neq \bot$, $\mathtt{val}^+_{\ell-1}(D) \neq \bot$, and $\mathtt{val}^+_{\ell-1}(C) \neq \mathtt{val}^+_{\ell-1}(D)$, then a pair of random assignments in some table P_i cannot change these values and make them collide by Lemma 8 (a).
(3) Both C and D are table-defined after this pair of assignments. Indeed, this pair of random assignments must be the one or posterior to the one which leads to C being enqueued, and D is already enqueued at this time, hence table-defined.
(4) After the pair of assignments, $\mathtt{val}^+_{\ell-1}(C) = \mathtt{val}^+_{\ell-1}(D) \neq \bot$.

Yet Lemma 11 exactly forbids that these conditions be simultaneously fulfilled for a good pair (η, φ).

Finally, this cannot have been because of a previous call to `ForceVal` by the same reasoning as before: C cannot be equivalent to the chain which is being completed when this call to `ForceVal` occurs (since when C is enqueued, no equivalent chain has been previously enqueued), so that by Lemma 12 (b) $\mathtt{val}^\delta_i(C)$ remains constant for all i and δ. The same reasoning shows that $\mathtt{val}^\delta_i(D)$ cannot change during a previous call to `ForceVal`.

We have excluded all possibilities, which shows that $\mathtt{val}^+_{\ell-1}(C) \notin P^+_{\ell-1}$ when C is dequeued. This concludes the proof. □

We are now ready to state the ultimate goal of this section, which is the following crucial lemma.

Lemma 17. *Consider an execution of $\mathcal{D}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Then all calls to* `ForceVal` *are non-overwriting.*

Proof. This is an immediate consequence of Lemmas 12 (a) and 16. □

4.7.4 Randomness Mapping

This section contains the final argument for proving the indistinguishability of the two systems $\Sigma_2(\eta, \varphi)$ and $\Sigma_3(\pi)$.

For any distinguisher $\mathcal{D}$, we define the distinguisher $\overline{\mathcal{D}}$ which runs $\mathcal{D}$ and then emulates a call to $\mathtt{EvalForward}(x, x \oplus k, 0, 13)$ (resp. $\mathtt{EvalBackward}(k \oplus y, y, 12, 0)$) for all queries $\mathcal{F}.\mathtt{Enc}(k, x)$ (resp. $\mathcal{F}.\mathtt{Dec}(k, y)$) made by $\mathcal{D}$, and outputs whatever $\mathcal{D}$ outputs. We say that $\overline{\mathcal{D}}$ is the distinguisher which *completes all chains* corresponding to $\mathcal{D}$. Note that $\overline{\mathcal{D}}$ makes at

most $13q$ queries if $\mathcal{D}$ makes at most q queries, and that $\overline{\mathcal{D}}$ has exactly the same advantage as $\mathcal{D}$ in distinguishing Σ_2 from Σ_3.

Recall that the definition of a good pair (η, φ) holds with respect to some fixed distinguisher. In all the following, when we mention a good pair (η, φ), this always means *with respect to* $\overline{\mathcal{D}}$.

We first state two simple lemmas regarding an execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ).

Lemma 18. *Consider an execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Assume that during the execution $\mathcal{F}.\mathtt{Enc}(k, x)$, resp. $\mathcal{F}.\mathtt{Dec}(k, y)$ is queried. Then, at the end of the execution, it holds that $E(+, k, x) = \mathtt{val}^{+}_{13}(x, x \oplus k, 0)$, resp. $E(-, k, y) = \mathtt{val}^{-}_{0}(y \oplus k, y, 12)$.*

Proof. We consider the case of a query to $\mathcal{F}.\mathtt{Enc}(k, x)$ (the case of a query to `Dec` is handled similarly). Assume first that the query was made by the simulator. Then this was while completing a chain, and clearly the equality holds right after completion of the chain. Since no entry is ever overwritten in tables E or P_i for a good pair (η, φ), the equality also holds at the end of the execution.

Assume now that the query was made by the distinguisher $\overline{\mathcal{D}}$. Since it completes all chains, at some point in the execution it must emulate a call to $\mathtt{EvalForward}(x, x \oplus k, 0, 13)$. In particular it must query the corresponding values $(6, +, x_6)$ and $(7, +, x_7)$ at some point. Denote y_6 and y_7 the corresponding answers. Then at the end of the execution we necessarily have that $P_6(+, x_6)$, $P_6(-, y_6)$, $P_7(+, x_7)$, and $P_7(-, y_7)$ are all defined in the respective hash tables. Thus, the partial chain $(y_6, x_7, 6)$ was necessarily enqueued at some point, and dequeued later, and hence completed at some point. As previously, the equality holds right after the completion, and also at the end of the execution. □

Lemma 19. *Consider an execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ with a good pair (η, φ). Then the number of calls to* `ForceVal` *made by the simulator is equal to the number of queries to η made by $\mathcal{F}$.*

Proof. Since no entry is overwritten in table E, the number of queries to η is exactly half the number of entries in E at the end of the execution. We describe a bijective mapping between calls to `ForceVal` and pairs of entries in E. Namely, to each call to `ForceVal` we associate the unique entry in E (and its corresponding inverse entry) that was read during during the call to `EvalForward` or `EvalBackward` just preceding the call to `ForceVal`. This mapping in one-to-one: indeed if for two distinct calls to `ForceVal` the same pair of entries is read, necessarily the second call to `ForceVal` overwrites an entry, contradicting Lemma 17.

This mapping is also onto: for each pair of entries in E we can identify a corresponding call to `ForceVal`. If the pair of entries was added due to a query to `Enc`/`Dec` made by the simulator, then we consider the call to `ForceVal` immediately following this query. If the pair of entries was added due to a query to `Enc`/`Dec` made by $\overline{\mathcal{D}}$, then the distinguisher will eventually emulate `EvalForward` or `EvalBackward` for the corresponding query, querying (or getting as answer) values y_6 and x_7. At the end of the emulation, consider the first chain which was enqueued during the execution and which was equivalent to $(y_6, x_7, 6)$ when it was enqueued. Then the pair of entries in E was read just before the call to `Adapt` made while completing this chain. □

In all the following, a sequence of *partial* permutation tables $\pi' = (\pi'_1, \ldots, \pi'_{12})$ is a sequence of functions $\pi'_i : \{+, -\} \times \{0,1\}^n \to \{0,1\}^n \cup \{*\}$ such that $\pi'_i(+, x) = y \neq *$ if and only if

$\pi'_i(-,y) = x \neq *$. A sequence of permutation tables $\pi = (\pi_1, \ldots, \pi_{12})$ is a sequence of functions $\pi_i : \{+,-\} \times \{0,1\}^n \to \{0,1\}^n$ such that $\pi_i(+,x) = y$ if and only if $\pi_i(-,y) = x$. We say that a sequence of permutation tables π *matches* a sequence of partial permutation tables π', denoted $\pi \cong \pi'$, if π and π' agree on all entries such that $\pi'_i(\delta, z) \neq *$.

Definition 8 (Randomness mapping). *We define the mapping Λ from pairs (η, φ) to partial tables $\pi' = (\pi'_1, \ldots, \pi'_{12})$, where $\pi'_i : \{+,-\} \times \{0,1\}^n \to \{0,1\}^n \cup \{*\}$, as follows. Run the experiment $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$; if $z' = \varphi_i(\delta, z)$ is ever read during the experiment, then define $\pi'_i(\delta, z) := z'$ and $\pi'_i(\bar{\delta}, z') := z$; otherwise, if a call to $\texttt{ForceVal}(x_\ell, y_\ell, \ell)$ occurs, then define $\pi'_\ell(+, x_\ell) := y_\ell$ and $\pi'_\ell(-, y_\ell) := x_\ell$ for the first such call. Finally, fill the remaining entries with the special symbol $*$.*

Lemma 20. *If (η, φ) is good (with respect to $\overline{\mathcal{D}}$), then $\Lambda(\eta, \varphi)$ is a sequence of partial permutation tables.*

Proof. Note that for a good pair (η, φ), $\Lambda(\eta, \varphi)$ is equal by definition to tables $(P_1, \ldots, P_{12})$ completed with symbols $*$ for entries which are left undefined at the end of the execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$. Hence the result immediately follows from the fact that for a good pair (η, φ), no entry is ever overwritten in tables P_i (Lemmas 8 (a) and 17). □

In the following, we say that a sequence of partial permutation tables π' is good if there exists a good pair (η, φ) such that $\Lambda(\eta, \varphi) = \pi'$. For a good π', we let $|\pi'_i|$ denote *half* the number of entries different from $*$ in table π'_i, and we let $|\pi'| = |\pi'_1| + \ldots + |\pi'_{12}|$.

Lemma 21. *Assume that $\mathcal{D}$ makes at most q queries. Let π' be a good sequence of partial permutation tables (with respect to $\overline{\mathcal{D}}$), and let (η', φ') be a good pair such that $\Lambda(\eta', \varphi') = \pi'$. Then the following properties hold:*

(a) The following conditions on a pair (η, φ) are equivalent:
  (i) (η, φ) is good and $\Lambda(\eta, \varphi) = \pi'$;
  (ii) for any sequence of permutation tables π such that $\pi \cong \pi'$, the transcript of the sequence of queries and answers to $(\mathcal{F}(\eta), \varphi)$ in $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ is the same as the transcript of queries and answers to $(\widetilde{\mathcal{C}}_{12}(\pi), \pi)$ in $\overline{\mathcal{D}}^{\Sigma_3(\pi)}$;
  (iii) the transcript of the sequence of queries and answers to (η, φ) in $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ is the same as the transcript of the sequence of queries and answers to (η', φ') in $\overline{\mathcal{D}}^{\Sigma_2(\eta',\varphi')}$.

(b) For any good pair (η, φ) such that $\Lambda(\eta, \varphi) = \pi'$ and any sequence of permutation tables π such that $\pi \cong \pi'$, the outputs of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ and $\overline{\mathcal{D}}^{\Sigma_3(\pi)}$ are equal.

(c) $|\pi'|$ is at most $12 \times 6 \times (13q)^2$.

Proof. We start by showing (a). Assume that (i) holds, *i.e.* consider a good pair (η, φ) such that $\Lambda(\eta, \varphi) = \pi'$. We show that (ii) holds. For this, we fix some $\pi \cong \pi'$, and we proceed by induction on the sequence of queries of $\overline{\mathcal{D}} \cup \mathcal{T}$ to $(\mathcal{F}(\eta), \varphi)$ or $(\widetilde{\mathcal{C}}_{12}(\pi), \pi)$. Assume that the sequence of queries and answers is the same in both systems up to some point in the execution, and consider the next query (which is the same in both systems since $\overline{\mathcal{D}}$ and $\mathcal{T}$ are both deterministic). Consider first the case of a query to $\varphi_i(\delta, z)/\pi_i(\delta, z)$. Then the answer is the same in both systems by definition of the mapping Λ. Consider now the case of a query to $\mathcal{F}.\texttt{Enc}(k, x)/\widetilde{\mathcal{C}}_{12}.\texttt{Enc}(k, x)$ (the case of a query to $\texttt{Dec}$ is handled similarly). By Lemma 18, the

answer to this query in the execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ is exactly what is obtained when evaluating the Even-Mansour construction forward at the end of the execution. Each answer to calls $P_i(+,x_i)$ in this evaluation was set either by a call to $\varphi_i(+,x_i)$, or a call $\varphi_i(-,y_i)$ which returned x_i, or during a call to `ForceVal`. In all cases the value of $\pi_i(+,x_i)$ must agree by definition of the randomness mapping and by the fact that no entry is overwritten for a good (η,φ). Hence, the answer to the query to $\widetilde{\mathcal{C}}_{12}.\texttt{Enc}(k,x)$ is the same in $\overline{\mathcal{D}}^{\Sigma_3(\pi)}$. Finally, it remains to consider queries to `Check`. For these queries, it is clear that the answer is the same in both systems since it is determined by previous queries and answers to `Enc` and `Dec`.

We then show that (ii) implies (iii). Assume that property (ii) holds for (η,φ). By assumption, (η',φ') is good and $\Lambda(\eta',\varphi')=\pi'$, so that property (ii) holds for (η',φ') by what was just proved above. The result then easily follows since the answer to any query to φ/φ' must agree with π', while the answer to any query to η/η' must agree with the answer returned by $\widetilde{\mathcal{C}}_{12}(\pi')$.

Finally, we show that (iii) implies (i). Since (η',φ') is good by assumption, assuming that the sequences of queries and answers to (η,φ) and to (η',φ') are the same implies that (η,φ) is good as well. Moreover, this also implies that $\Lambda(\eta,\varphi)=\Lambda(\eta',\varphi')=\pi'$.

We then show (b). Let (η,φ) be a good pair such that $\Lambda(\eta,\varphi)=\pi'$ and π such that $\pi\cong\pi'$. Because of (a), we know that the sequence of queries and answers of $\overline{\mathcal{D}}\cup\mathcal{T}$ is the same in $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ and $\overline{\mathcal{D}}^{\Sigma_3(\pi)}$. This clearly implies that the sequence of queries and answers of $\overline{\mathcal{D}}$ alone is the same in both systems, and hence also its output.

Finally, it remains to show (c). Since the total number of queries of $\overline{\mathcal{D}}$ is at most $13q$, it follows from Lemma 2 that the size of P_i^+ and P_i^- at the end of the execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta',\varphi')}$ is at most $6\times(13q)^2$. Since entries which are different from $*$ in $\pi'=\Lambda(\eta',\varphi')$ are exactly entries which are defined in tables P_i at the end of the execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta',\varphi')}$, the results follows (recall that $|\pi'_i|$ is by definition half the number of entries different from $*$ in π'_i). □

Lemma 22. *Let π' be a good sequence of partial permutation tables. Then:*

$$\left(1-\frac{|\pi'|^2}{2^n}\right)\Pr\left[\pi\cong\pi'\right]\leq\Pr\left[(\eta,\varphi)\ \textit{is good}\ \wedge\Lambda(\eta,\varphi)=\pi'\right]\leq\Pr\left[\pi\cong\pi'\right]\ ,$$

where the probabilities are over a uniformly random sequence of permutation tables π and a uniformly random pair (η,φ).

Proof. Fix a good sequence of partial permutation tables π'. We first show that:

$$\Pr\left[(\eta,\varphi)\text{ is good }\wedge\Lambda(\eta,\varphi)=\pi'\right]=2^{-n|\pi'|}\ .$$

Fix some good pair (η',φ') such that $\Lambda(\eta',\varphi')=\pi'$. By Lemma 21 (a), a pair (η,φ) is good and $\Lambda(\eta,\varphi)=\pi'$ if and only if the transcript of the sequence of queries and answers to (η,φ) in $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ is the same as the transcript of the sequence of queries and answers to (η',φ') in $\overline{\mathcal{D}}^{\Sigma_2(\eta',\varphi')}$. Each answer is identical in both systems with probability 2^{-n} over the random choice of (η,φ). Note moreover that $|\pi'|$ is exactly the sum of the number of queries to φ' plus the number of calls to `ForceVal` in the execution of $\overline{\mathcal{D}}^{\Sigma_2(\eta',\varphi')}$, and by Lemma 19 the number of calls to `ForceVal` is equal to the number of queries to η'. The claim follows.

Clearly, we also have:

$$\Pr\left[\pi \cong \pi'\right] = \prod_{i=1}^{12} \frac{1}{2^n(2^n-1)\dots(2^n-|\pi'_i|+1)} \enspace .$$

The upper bound then follows easily, while the lower bound comes from:

$$\begin{aligned}\prod_{i=1}^{12} 2^n(2^n-1)\dots(2^n-|\pi'_i|+1) &= 2^{n|\pi'|}\prod_{i=1}^{12}\left(1-\frac{1}{2^n}\right)\cdots\left(1-\frac{|\pi'_i|-1}{2^n}\right)\\ &\geq 2^{n|\pi'|}\left(1-\frac{|\pi'_1|^2+\ldots+|\pi'_{12}|^2}{2^n}\right)\\ &\geq 2^{n|\pi'|}\left(1-\frac{|\pi'|^2}{2^n}\right) \enspace .\end{aligned}$$

This concludes the proof. □

Lemma 23. *For any distinguisher $\mathcal{D}$ which makes at most q queries in total, we have:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_2(\eta,\varphi)}=1\right]-\Pr\left[\mathcal{D}^{\Sigma_3(\pi)}=1\right]\right| \leq \frac{2^{89}\times q^{12}}{2^n} \enspace .$$

Proof. Consider the distinguisher $\overline{\mathcal{D}}$ associated with $\mathcal{D}$, and recall that its advantage is the same as the one of $\mathcal{D}$. For conciseness, we define:

$$\begin{aligned}\varepsilon &= \left|\Pr\left[\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}=1\right]-\Pr\left[\overline{\mathcal{D}}^{\Sigma_3(\pi)}=1\right]\right|\\ \varepsilon' &= \Pr\left[\pi \text{ is good } \wedge\, \overline{\mathcal{D}}^{\Sigma_3(\pi)}=1\right]-\Pr\left[(\eta,\varphi) \text{ is good } \wedge\, \overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}=1\right] \enspace .\end{aligned}$$

First, we note that:

$$\varepsilon \leq |\varepsilon'| + \Pr\left[(\eta,\varphi) \text{ is not good}\right] + \Pr\left[\pi \text{ is not good}\right] \enspace .$$

Since $\overline{\mathcal{D}}$ makes at most $13q$ queries, we have by Lemma 7 that:

$$\Pr\left[(\eta,\varphi) \text{ is not good}\right] \leq \frac{2^{43}\times(13q)^{12}}{2^n} \enspace .$$

Denote Θ the set of good partial permutation tables π'. Then:

$$\begin{aligned}\Pr\left[\pi \text{ is not good}\right] &= 1-\sum_{\pi'\in\Theta}\Pr\left[\pi\cong\pi'\right]\\ &\leq 1-\sum_{\pi'\in\Theta}\Pr\left[(\eta,\varphi) \text{ is good } \wedge\, \Lambda(\eta,\varphi)=\pi'\right] \qquad \text{(Lemma 22)}\\ &= \Pr\left[(\eta,\varphi) \text{ is not good}\right] \leq \frac{2^{43}\times(13q)^{12}}{2^n} \enspace .\end{aligned}$$

By Lemma 21 (b), for any good π', the output of $\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ is the same for any (η,φ) such that $\Lambda(\eta,\varphi)=\pi'$ Hence we can consider the set Θ_1 defined as the set of good π' such that

$\overline{\mathcal{D}}^{\Sigma_2(\eta,\varphi)}$ outputs 1 for any good pre-image of π'. Again by Lemma 21 (b), for any $\pi' \in \Theta_1$ and any π such that $\pi \cong \pi'$, $\overline{\mathcal{D}}^{\Sigma_3(\pi)}$ outputs 1, so that:

$$\varepsilon' = \sum_{\pi' \in \Theta_1} \Pr\left[\pi \cong \pi'\right] - \Pr\left[(\eta,\varphi) \text{ is good } \wedge \Lambda(\eta,\varphi) = \pi'\right] \enspace .$$

Note that by Lemma 22, $\varepsilon' \geq 0$, so that:

$$\begin{aligned} |\varepsilon'| = \varepsilon' &\leq \sum_{\pi' \in \Theta_1} \frac{|\pi'|^2}{2^n} \Pr\left[\pi \cong \pi'\right] && \text{(Lemma 22)} \\ &\leq \frac{(12 \times 6 \times (13q)^2)^2}{2^n} && \text{(Lemma 21 } (c)\text{)} \enspace . \end{aligned}$$

Combining all previous inequalities, we finally obtain that:

$$\varepsilon \leq \frac{(12 \times 6 \times (13q)^2)^2}{2^n} + 2 \times \frac{2^{43} \times (13q)^{12}}{2^n} \leq \frac{2^{89} \times q^{12}}{2^n} \enspace ,$$

which concludes the proof. □

4.8 From the Third to the Fourth System

Finally, we upper bound the distance between systems $\Sigma_3(\pi)$ and Σ_4.

Lemma 24. *For any distinguisher $\mathcal{D}$ which makes at most q queries in total, we have:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_3(\pi)} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_4} = 1\right]\right| \leq \frac{2^{89} \times q^{12}}{2^n} \enspace .$$

Proof. First, note that because of Lemma 23, the distinguisher must eventually output an answer when interacting with $\Sigma_3(\pi)$ with probability at least $1 - 2^{89} \times q^{12}/2^n$. Suppose now that in system $\Sigma_3(\pi)$, $\mathcal{T}(\pi)$ eventually answers a query $\texttt{Query}(i,\delta,z)$ asked by the distinguisher. Then necessarily the answer is $\pi_i(\delta,z)$. Indeed, $\mathcal{T}$ sets this value either by accessing directly $\pi_i(\delta,z)$, in which case the claim is clear, or during a call to `ForceVal`. In the later case, the answer is obtained by evaluating the Even-Mansour construction forward or backward, wrapping up with a call to $\widetilde{\mathcal{C}}_{12}(\pi).\texttt{Enc}$ or $\widetilde{\mathcal{C}}_{12}(\pi).\texttt{Dec}$. Since $\widetilde{\mathcal{C}}_{12}$ uses tables π_i as well, the answer to query $\texttt{Query}(i,\delta,z)$ must be $\pi_i(\delta,z)$ as well. Consequently, for any π such that $\mathcal{D}^{\Sigma_3(\pi)}$ eventually outputs an answer, $\mathcal{D}^{\Sigma_4}$ returns the same answer when the random permutations $\boldsymbol{P}_i$ in Σ_4 take their randomness in tables π_i. This implies the result. □

References

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the Indifferentiability of Key-Alternating Ciphers. IACR Cryptology ePrint Archive Report 2013/061, 2013. Available at http://eprint.iacr.org/2013/061.

[AFPW11] Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On Cipher-Dependent Related-Key Attacks in the Ideal-Cipher Model. In Antoine Joux, editor, *Fast Software Encryption - FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 128–145. Springer, 2011.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Symposium on Foundations of Computer Science - FOCS '97*, pages 394–403. IEEE Computer Society, 1997.

[Bih94] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *Journal of Cryptology*, 7(4):229–246, 1994.

[BK03] Mihir Bellare and Tadayoshi Kohno. A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.

[BKL+12] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations - (Extended Abstract). In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2012.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*. Springer, 2009. To appear.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[Bla06] John Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. In Matthew J.B. Robshaw, editor, *Fast Software Encryption - FSE '06*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2006.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR06] Mihir Bellare and Thomas Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2006.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *Symposium on Theory of Computing - STOC '98*, pages 209–218. ACM, 1998. Full version available at http://arxiv.org/abs/cs.CR/0010019.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[Dam89] Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[Des00] Anand Desai. The Security of All-or-Nothing Encryption: Protecting against Exhaustive Key Search. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2000.

[DGHM12] Grégory Demay, Peter Gazi, Martin Hirt, and Ueli Maurer. Resource-Restricted Indifferentiability. IACR Cryptology ePrint Archive Report 2012/613, 2012. Available at http://eprint.iacr.org/2012/613.

[DR01] Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *Cryptography and Coding 2001*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[EM97] Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology*, 10(3):151–162, 1997.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GM09] Peter Gazi and Ueli M. Maurer. Cascade Encryption Revisited. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2009.

[GP10] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption - FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[Gra02] Louis Granboulan. Short Signatures in the Random Oracle Model. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 2002.

[GT12] Peter Gazi and Stefano Tessaro. Efficient and Optimally Secure Key-Length Extension for Block Ciphers via Randomized Cascading. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2012.

[Hir04] Shoichi Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2004.

[Hir06] Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In Matthew J.B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The Equivalence of the Random Oracle Model and the Ideal Cipher Model, Revisited. In Lance Fortnow and Salil P. Vadhan, editors, *Symposium on Theory of Computing - STOC 2011*, pages 89–98. ACM, 2011. Full version available at http://arxiv.org/abs/1011.1264.

[JJV02] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2002.

[Jon02] Jakob Jonsson. An OAEP Variant With a Tight Security Proof. IACR Cryptology ePrint Archive Report 2002/034, 2002. Available at http://eprint.iacr.org/2002/034.

[KR96] Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 1996.

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2007.

[Kün09] Robin Künzler. Are the random oracle and the ideal cipher models equivalent? Master's thesis, ETH Zurich, Switzerland, 2009.

[LM92] Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.

[LPS12] Rodolphe Lampe, Jacques Patarin, and Yannick Seurin. An Asymptotically Tight Security Analysis of the Iterated Even-Mansour Cipher. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 278–295. Springer, 2012.

[LR86] Michael Luby and Charles Rackoff. Pseudo-random Permutation Generators and Cryptographic Composition. In *Symposium on Theory of Computing - STOC '86*, pages 356–363. ACM, 1986.

[LR88] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[LSS11] Jooyoung Lee, Martijn Stam, and John P. Steinberger. The Collision Security of Tandem-DM in the Ideal Cipher Model. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 561–577. Springer, 2011.

[Mau92] Ueli M. Maurer. A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generator. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1992.

[Men12] Bart Mennink. Optimal Collision Security in Double Block Length Hashing with Single Length Key. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 526–543. Springer, 2012.

[Mer89] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[MP03] Ueli M. Maurer and Krzysztof Pietrzak. The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 544–561. Springer, 2003.

[MPP09] Marine Minier, Raphael C.-W. Phan, and Benjamin Pousse. Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 60–76. Springer, 2009.

[MPS12] Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the Public Indifferentiability and Correlation Intractability of the 6-Round Feistel Construction. In Ronald Cramer, editor, *Theory of Cryptography Conference - TCC 2012*, volume 7194 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2012. Full version available at http://eprint.iacr.org/2011/496.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *Theory of Cryptography Conference- TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[Pat90] Jacques Patarin. Pseudorandom Permutations Based on the DES Scheme. In Gérard D. Cohen and Pascale Charpin, editors, *EUROCODE '90*, volume 514 of *Lecture Notes in Computer Science*, pages 193–204. Springer, 1990.

[Pat04] Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2004.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[Seu09] Yannick Seurin. *Primitives et protocoles cryptographiques à sécurité prouvée*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, France, 2009.

[Seu11] Yannick Seurin. A Note on the Indifferentiability of the 10-Round Feistel Construction, March 2011. Unpublished note available from the author.

[Sha49] Claude Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[Sim98] Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998.

[Ste07] John P. Steinberger. The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2007.

[Ste12] John Steinberger. Improved Security Bounds for Key-Alternating Ciphers via Hellinger Distance. IACR Cryptology ePrint Archive Report 2012/481, 2012. Available at http://eprint.iacr.org/2012/481.

[SY11] Yu Sasaki and Kan Yasuda. Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes. In Antoine Joux, editor, *Fast Software Encryption - FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 397–415. Springer, 2011.

[Vau03] Serge Vaudenay. Decorrelation: A Theory for Block Cipher Security. *Journal of Cryptology*, 16(4):249–286, 2003.

[Win84] Robert S. Winternitz. A Secure One-Way Hash Function Built from DES. In *IEEE Symposium on Security and Privacy*, pages 88–90, 1984.

A Removing the "Set Uniform" Rounds

In this section, we point out the obstacles that appear when removing the four "Set Uniform" rounds surrounding the two adaptation rounds. Namely, consider the 8-round single-key Even-Mansour construction depicted on Figure 5, and a simulator that works exactly as the one of Section 4.2 (we do not formally define this simulator, which can be derived from the one of Section 4.2 by adequately renumbering permutations).

For this new simulator, we can show the analogue of Lemma 15: namely, just before the call to $\texttt{InQuery}(i, \delta, z)$ which led to some chain C being enqueued to be adapted at position $\ell = 3$ or $\ell = 6$, one has $\texttt{val}_\ell^+(C) = \perp$ and $\texttt{val}_\ell^-(C) = \perp$. The difficulty for proving that when C is dequeued, $\texttt{val}_\ell^+(C) \notin P_\ell^+$ and $\texttt{val}_\ell^-(C) \notin P_\ell^-$ is that it seems quite hard to show that $\texttt{val}_\ell^\delta(C)$ cannot collision with $\texttt{val}_\ell^\delta(D)$ for another chain D which has been enqueued before C. The reason for this is that $\texttt{val}_\ell^\delta(C)$ does not necessarily remain constant during calls to `ForceVal` occurring after C has been enqueued (*i.e.* the analogue of Lemma 12 (b) does not hold). We give an simple example of this kind of behavior.

Consider a distinguisher making the following sequence of queries:

1. choose arbitrary values $x_3, k, k' \in \{0,1\}^n$ with $k \neq k'$;
2. evaluate the chain corresponding to x_3 and k forward with the following queries:

$$\begin{aligned} y_3 &:= P_3(x_3) \\ x_4 &:= y_3 \oplus k \\ y_4 &:= P_4(x_4) \\ x_5 &:= y_4 \oplus k \\ y_5 &:= P_5(x_5) \end{aligned}$$

3. evaluate the chain corresponding to x_3 and k' backward with the following queries:

$$\begin{aligned} y'_2 &:= x_3 \oplus k' & \qquad x'_9 &:= E(k', y'_0) \\ x'_2 &:= P_2^{-1}(y_2) & \qquad y'_8 &:= x'_9 \oplus k' \\ y'_1 &:= x'_2 \oplus k' & \qquad x'_8 &:= P_8^{-1}(y'_8) \\ x'_1 &:= P_1^{-1}(y'_1) & \qquad y'_7 &:= x'_8 \oplus k' \\ y'_0 &:= x'_1 \oplus k' & \qquad x'_7 &:= P_7^{-1}(y'_7) \end{aligned}$$

We now analyze the internal behavior of the simulator when receiving this sequence of queries $(P_3(x_3), \ldots, P_7^{-1}(y'_7))$. It simply draws the first three answers y_3, y_4 and y_5 uniformly at random. Moreover, after query $P_5(x_5)$, it detects and enqueues the partial chain $(y_4, x_5, 4, 6)$, and dequeues it immediately, internally setting permutation values that are irrelevant to the analysis. Then, it defines answers $x'_2, \ldots, x'_7$ to subsequent queries uniformly at random. After the last query $P_7^{-1}(y'_7)$, it detects and enqueues $(y'_0, x'_1, 0, 6)$, and dequeues it immediately. When completing this chain, the simulator internally defines $x'_4 := y_3 \oplus k'$, draws $y'_4 \leftarrow_{\$} \{0,1\}^n$, sets $P_4(x_4) := y'_4$, and enqueues the partial chain $(y'_4, x_5, 4, 3)$. It then resumes completion of chain $(y'_0, x'_1, 0, 6)$ by defining $x'_5 := y'_4 \oplus k'$, draws $y'_5 \leftarrow_{\$} \{0,1\}^n$, sets $P_5(x'_5) := y'_5$, and enqueues $(y_4, x'_5, 4, 6)$ (as well as $(y'_4, x'_5, 4, 6)$, but this chain is equivalent to the chain being completed and does not play any role in the analysis). It then finishes to complete $(y'_0, x'_1, 0, 6)$, and continues dequeuing partial chains. Denote $D = (y'_4, x_5, 4)$ and $C = (y_4, x'_5, 4)$. At this point, we note that $\mathtt{val}_3^-(D) = \mathtt{val}_3^-(C) \neq \perp$. Indeed, $\mathtt{val}_3^-(D) = x'_4 \oplus y'_4 \oplus x_5$ and $\mathtt{val}_3^-(C) = x_4 \oplus y_4 \oplus x'_5$, and this two values are equal since by construction:

$$x_4 \oplus y_4 \oplus x_5 = x_4 \oplus k = y_3 = x'_4 \oplus k' = x'_4 \oplus y'_4 \oplus x'_5 \ .$$

In the following, we denote y'_3 this common value $\mathtt{val}_3^-(D) = \mathtt{val}_3^-(C)$ The next chain to be dequeued is D, to be adapted at $\ell = 3$. The simulator evaluates the chain forward until $x'_3 := \mathtt{val}_3^+(D)$ is defined, and then makes a call to $\mathtt{ForceVal}(x'_3, y'_3, 3)$ to adapt the chain. This call to `ForceVal` has the following properties:

- before the call to $\mathtt{ForceVal}(x'_3, y'_3, 3)$, C is table-defined (and enqueued) and not equivalent to D;
- before the call to $\mathtt{ForceVal}(x'_3, y'_3, 3)$, $\mathtt{val}_2^-(C) = \perp$ since $y'_3 \notin P_3^-$;
- after the call to $\mathtt{ForceVal}(x'_3, y'_3, 3)$, $\mathtt{val}_2^-(C) \neq \perp$.

Hence we see that the analogue of Lemma 12 (b) does not hold.

We stress that we do not know whether the simulator adapted for eight rounds can be attacked, we just pointed out which difficulties arise when trying to straightforwardly transpose the proof. The situation for the Feistel construction is quite different. Indeed, in that case, when removing the four buffer rounds in order to adapt the simulator of [HKT11] for fourteen rounds to ten rounds, there is actually an attack [Seu11]. For the iterated Even-Mansour cipher, it remains an open problem to either modify the distinguisher described above to obtain a real attack (*e.g.* by having the simulator overwrite a value at some point), showing that this simulator cannot be used to prove indifferentiability for eight rounds, or to find a more complex proof circumventing this difficulty.

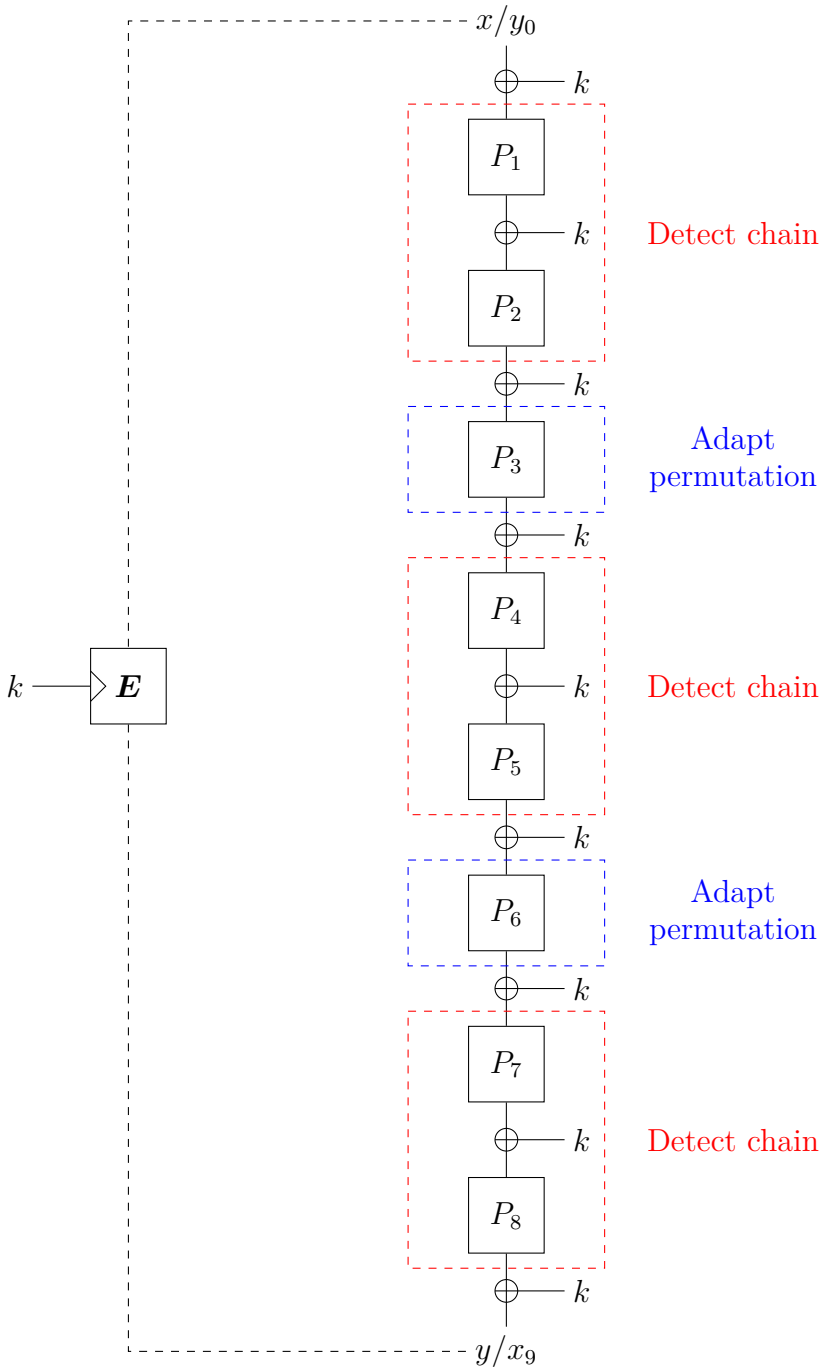

Fig. 5. Detection and adaptation zones used by the simulator for 8 rounds.