A Synthetic Indifferentiability Analysis of Interleaved Double-Key Even-Mansour Ciphers [⋆]

Chun Guo[1,2], and Dongdai Lin[1⋆⋆]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] University of Chinese Academy of Sciences, China
{guochun,ddlin}@iie.ac.cn

Abstract. Iterated Even-Mansour scheme (IEM) is a generalization of the basic 1-round proposal (ASIACRYPT '91). The scheme can use one key, two keys, or completely independent keys.
Most of the published security proofs for IEM against relate-key and chosen-key attacks focus on the case where all the round-keys are derived from a single master key. Whereas results beyond this barrier are relevant to the cryptographic problem whether a secure blockcipher with key-size twice the block-size can be built by mixing two *relatively independent* keys into IEM and iterating sufficiently many rounds, and this strategy actually has been used in designing blockciphers for a long-time.
This work makes the first step towards breaking this barrier and considers IEM with Interleaved Double *independent* round-keys:

$$\mathrm{IDEM}_r((k_1,k_2),m) = k_i \oplus (P_r(\ldots k_1 \oplus P_2(k_2 \oplus P_1(k_1 \oplus m))\ldots)),$$

where $i = 2$ when r is odd, and $i = 1$ when r is even. As results, this work proves that 15 rounds can achieve (full) indifferentiability from an ideal cipher with $O(q^8/2^n)$ security bound. This work also proves that 7 rounds is sufficient and necessary to achieve sequential-indifferentiability (a notion introduced at TCC 2012) with $O(q^6/2^n)$ security bound, so that IDEM_7 is already correlation intractable and secure against any attack that exploits evasive relations between its input-output pairs.

Keywords: blockcipher, ideal cipher, indifferentiability, key-alternating cipher, Even-Mansour cipher, correlation intractability.

[⋆] ©IACR 2015. This is the revised full version of the article submitted by the authors to the IACR and to Springer-Verlag on September 6, 2015, which appears in the proceedings of ASIACRYPT 2015.
[⋆⋆] Corresponding author.

Table of Contents

1 Introduction

Blockciphers are arguably the most important primitives in cryptography. A blockcipher $\mathsf{BC}[\kappa, n] : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ maps a κ-bit key K and an n-bit input x to an n-bit output y. For each key K, the map $\mathsf{BC}[\kappa, n](K, \cdot)$ is a permutation, and is efficiently invertible.

Most of the existing blockcipher designs can be roughly split into two families, namely Feistel ciphers and substitution-permutation networks. The latter are known as the structure of AES, and can be generalized as *key-alternating ciphers* [DR02]/*iterated Even-Mansour ciphers* (IEM for short). An r-round IEM cipher IEM_r consists of r fixed n-bit permutations P_i separated by key addition

$$\mathrm{IEM}_r(K, m) = k_r \oplus P_r(\ldots k_2 \oplus P_2(k_1 \oplus P_1(k_0 \oplus m))\ldots).$$

The single round Even-Mansour (the case $r = 1$) was developed in 1991 [EM93] in an attempt to turn a single permutation into a family of permutations (blockcipher). IEM_1 has been proved pseudorandom when the underlying permutation is random and public while the keys are secret. Since then, a soar of studies on IEM has been witnessed (especially in the recent half decade), for instance, on minimization [DKS13,CLL$^+$14], on pseudorandomness [BKL$^+$12,Ste12,LPS12,CS14], on related-key (RK) security [FP15,CS15], and on attacks (notable examples include [DKS13,DDKS15,DDKS14]). The pseudorandomness results showed that IEM is provably secure in traditional single secret key settings.

Indifferentiability of IEM. The studies on *indifferentiability* and *sequential-indifferentiability* (*seq-indifferentiability*) of IEM are mainly motivated by further validating the SPN-based blockcipher design methodology by proving IEM secure against *known-key* and *chosen-key (CK) attacks*, in which the adversary knows and chooses keys and tries to exhibit non-randomness. Roughly speaking, indifferentiability of IEM means that IEM can be as secure as an *ideal cipher* [MRH04], whereas seq-indifferentiability of IEM implies that IEM is *correlation intractable* [CGH04], and there is no relation between the inputs and outputs of IEM that can be exploited by an attack (even a chosen-key one) [MPS12]. Here the ideal cipher $\mathbf{IC}[\kappa, n] : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is taken randomly from the set of $(2^n!)^{2^\kappa}$ possible choices of $\mathsf{BC}[\kappa, n]$. In this work, $\mathbf{IC}[2n, n]$ will be referred by $\mathbf{E}$.

As to (seq-)indifferentiability, we have been aware of four works: [ABD$^+$13], [LS13], [CS15], and [Ste15]. [ABD$^+$13] showed that IEM_5 is indifferentiable from $\mathbf{IC}[\kappa, n]$, if the round-key is derived from a preimage-aware key derivation function (KDF). On the other hand, [LS13] and [CS15] concentrated on *single-key EM* (SEM) in which the user-provided n-bit master key is directly used at each round: [LS13] proved that SEM_{12} (12-round SEM; similarly for SEM_4 and SEM_9) is indifferentiable, while [CS15] proved that SEM_4 is seq-indifferentiable. In [Ste15], Steinberger proved the indifferentiability of SEM_9. Results on SEM are closer to concrete designs, since they can be easily generalized to the case where each round-key is derived by an *efficiently invertible* permutation.

Problem: Even-Mansour with Two Keys. Existing works on provable security of IEM in RK and CK settings almost all focus on the SEM context: [LS13] (ASIACRYPT 2013), [FP15] (FSE 2015), [CS15] (EUROCRYPT 2015) (except for those considered random oracle modeled KDF, e.g. [ABD$^+$13]). This work makes the first step towards breaking this barrier and considers the following problem: *can we obtain an ideal cipher by mixing two independent keys into IEM and iterating enough rounds?* (a problem left open by Lampe and Seurin (LS) [LS13]).[3] This problem is far from being trivial because all the works on SEM (in RK and CK settings) crucially rely on the correlation between *all* round-keys, so that they cannot be directly generalized to double-key case. Also, the independence between round-keys may bring in weakness – the most extreme case is IEM with completely independent round-keys, which is vulnerable to trivial related-key attacks. This problem is also practical since the idea is really used in existing designs such as AES-256 [DR02], Serpent [ABK98], and LED-128 [GPPR11] – note that they (certainly) mix the keys into the state by lightweight and efficient operations and iterate, rather than use some very complex hash function to seal the $2n$ key bits first. The intuition is that by iterating enough rounds, such designs will be "secure"; but the fact that the diffusion of the $2n$ key bits is relatively slow brings in doubts (e.g. doubts on AES-256 [KHP07,BDK$^+$10]). The fact that among the three AES variants, AES-256 was the first that is theoretically broken [BK09] seems to support such doubts, and this attack raises a problem whether there exists a $\mathsf{BC}[2n, n]$ design behaving like $\mathbf{IC}[2n, n]$;[4] due to this, it is necessary to either validate (using a security proof) or negate (using a generic attack) this intuitive methodology.

To dig out a solution, note that using one key in the first $n/2$ rounds while using the other in the last $n/2$ rounds is trivially insecure [LS13]. Instead, a (seemingly) more promising approach to mixing two keys into IEM is the idea behind LED-128 [GPPR11], that is, interleaving the xoring of them: we name it *interleaved double-key Even-Mansour cipher* (IDEM for short; see Fig. 1 for an illustration). More formally, the r-round variant is written as follows:

$$\mathrm{IDEM}_r((k_1, k_2), m) = k_t \oplus P_r(\dots k_2 \oplus P_3(k_1 \oplus P_2(k_2 \oplus P_1(k_1 \oplus m)))),$$

where $t = 2$ when r is odd, and $t = 1$ when r is even. LS viewed IDEM as a promising solution to the problem mentioned before, and gave an extremely preliminary analysis, which led to the *conjecture* that 15 rounds is sufficient to achieve indifferentiability; but no concrete proof exists. Moreover, the provable security of IDEM with shorter rounds has not been considered yet.

[3] A trivial solution to building $\mathbf{IC}[2n, n]$ by IEM is hash-than-encrypt, which has been included in [ABD$^+$13]. It was also discussed in [CDMS10]. But this solution imposes strong burden on the key derivation and is far from practical designs.

[4] Please see [CDMS10], page 275: *as of 2009 it is unclear if we have a candidate block-cipher with key-size larger than block-size that behaves like an ideal cipher.*

Contributions. We give the first indifferentiability proof for 15-round IDEM. This is the first main result of this work. Interestingly, this matches LS's conjecture, but the proof is obtained by an approach quite different from they expected.

To obtain security guarantees for shorter round cases, we prove that IDEM_7 is seq-indifferentiable from $\mathbf{IC}[2n, n]$; therefore, IDEM_7 is also correlation intractable in the random permutation model [MPS12], and resists all attacks that exploit evasive relation between its inputs and outputs. We also find a sequential distinguisher against IDEM_6 (which is actually an easy extension of LS's attack against SEM_3 [LS13]), so that 7 rounds is also necessary. All the results are summarized by the following informal theorem.

Theorem. *For the construction* IDEM *based on completely independent random permutations, 6 rounds is not (seq-)indifferentiable; 7 rounds is seq-indifferentiable from* $\mathbf{IC}[2n, n]$ *with* $O(q^6/2^n)$ *security bound, and is also correlation intractable in the random permutation model; 15 rounds is indifferentiable from* $\mathbf{IC}[2n, n]$ *with* $O(q^8/2^n)$ *security bound.*

Due to the independence between the two n-bit round keys, at current time, we are not sure whether the results can be generalized to IEM with "very general" key schedules; however, for the first time, these results indeed validate the (seemingly long standing) *design principle* to some extent in the open-key model, i.e. a secure blockcipher $\mathsf{BC}[2n, n]$ can be built from key-alternating ciphers without using very complex KDFs, or even without any KDF. Especially, they show that the intuition behind the key schedule of LED-128 is sound. However, they certainly cannot provide direct security guarantee for LED-128 – in fact, as theoretical results, they do not guarantee the security of *ANY concrete blockcipher.* As already mentioned, whether there exist some designs that "behave like" $\mathbf{IC}[2n, n]$ have to be supported by more (cryptanalysis) works.

Techniques. To prove indifferentiability and seq-indifferentiability, one first builds a simulator to mimic the behaviors of all the underlying permutations. Taking IDEM_{15} as an example, consider a sequence of pairs of input and output (IO for short) $(x_1, y_1), \ldots, (x_{15}, y_{15})$ (called a *computation path/chain*) of the 15 permutations simulated by the simulator, which satisfies $y_i \oplus x_{i+1} = k_2$ when i is odd, and $y_i \oplus x_{i+1} = k_1$ when i is even. The simulator should ensure that each such chain simulated by it matches the ideal cipher $\mathbf{E}$, i.e. $\mathbf{E}((k_1, k_2), x_1 \oplus k_1) = y_{15} \oplus k_2$. The basic idea to reach this goal is Coron et al.'s *simulation via chain completion* technique [CHK+14], which has achieved success in (weaker) indifferentiability proofs for a variety of idealized blockciphers. It requires the simulator $\mathbf{S}$ to *detect* partial computation chains formed by the queries of the distinguisher, and *completes* the chains in advance by querying the ideal cipher $\mathbf{E}$, such that $\mathbf{S}$ is ready for answering queries in the future. To simulate answers that are consistent with $\mathbf{E}$, $\mathbf{S}$ has to use the answer from $\mathbf{E}$ to define some simulated answers; this action is called *adaptation.*

Detect Chains. To detect the so-called partial chains, note that the construction IDEM has the following property: given 4 values of 3 permutations y_i, x_{i+1}, y_{i+1}, and x_{i+2} (namely, an output of P_i, a pair of IO of P_{i+1}, and an input to P_{i+2}), the two associated keys can be derived as $k = y_i \oplus x_{i+1}$ and $k' = y_{i+1} \oplus x_{i+2}$, and it is possible to move forward and backward along the path. By this, at some place, using three rounds for chain detection is necessary – this idea has already appeared in [LS13].

Overall Strategy of the Simulators. As to the overall strategy, the simulator used to prove seq-indifferentiability of IDEM$_7$ is quite close to those for 6-round Feistel [MPS12] and SEM$_4$ [CS15]: the simulator *detects* partial chains at the *three middle round* of IDEM$_7$, completes them forward or backward, and finally adapts them at the first or last round – depending on concrete contexts.

On the other hand, the simulator used to prove the indifferentiability of IDEM$_{15}$ is motivated by Steinberger's illustration of indifferentiability proof for SEM$_9$ [Ste15]. The overall strategy requires detecting chains both at the two sides and at the middle – which is similar to several previous works (e.g. [CHK$^+$14]). The core idea in this part is a so-called "pureness" property which is different from [CHK$^+$14]: the simulator may fall into a recursive chain completion process; however, *during each such recursive completion process, all the partial chains are to be adapted at the same round*; as a consequence, *when a partial chain is to be completed, its extending is necessarily due to simulator defining new simulated answers to random ones rather than the adaptation of some other chains, so that the "endpoints" of this chain are always random.* Whereas in the context of IDEM, to uniquely specify a chain requires at least 3 values of 3 consecutive permutations, so that the adversary has more freedom to choose values and make different chains collide. With this in mind, we arrange two rounds to surround each adaptation zone to ensure different chains diverge in the adaptation zone; following an old convention [CHK$^+$14], we call them *buffer rounds.*[5] For a more detailed overview of the simulator, we refer to subsection 3.1.

In the indifferentiability proof for IDEM$_{15}$, we used an *active-chain-oriented bad events define strategy*, which is motivated by the analysis of IDEM$_7$: we directly define some bad events to be with respect to the chains that are active during the completion process. This helps us achieving the $O(q^8/2^n)$ indifferentiability security bound in spite of the complex character of IDEM. Albeit loose, this bound has been quite well-looking compared to similar (full) indifferentiability proofs for idealized blockciphers (the best non-flawed one(s) among them reached the level of $O(q^{10}/2^n)$ [ABD$^+$13]).

Summary: What are Inherited and What are Novel? Technically speaking, we inherit the simulation via chain completion technique, the randomness mapping argument, and the basic idea for simulator termination argument from [CHK$^+$14]; we also inherit (and adapt) the overall frameworks of Steinberger (which dates back to Seurin [Seu09]) and Cogliati et al. [CS15] (which dates back to Mandal et al. [MPS12]). Our novelties mainly lie in the proof for IDEM$_{15}$: first, we use a bad

[5] But our "buffer" rounds deviate from those in [CHK$^+$14], in the sense that the values in them **can be defined** when the simulator is completing other chains.

event to establish a slightly tighter bound on the size of the history ($O(q^2/2^n)$) and the simulator's complexity; second, we define the bad events to be so-called active-chain-oriented, so that the probability can be very low ($O(q^6/2^n)$). They two together enable to establish the $O(q^8/2^n)$ security bound.

Organization. Sect. 2 presents preliminaries. Sect. 3 contains the first main result – the indifferentiability of IDEM_{15}, and the proof sketch. Sect. 4 presents the (generalized) sequential distinguisher for IDEM_6. Sect. 5 contains the second main result – the seq-indifferentiability of IDEM_7. Sect. 6 concludes. Finally, the Appendices present some further discussions on the model IDEM as well as on improving results in this work.

2 Preliminaries and Notations

This work focuses on $\mathsf{BC}[2n, n]$, say, blockciphers with n-bit blocks and $2n$-bit keys. Throughout the remaining, the n-bit round-keys are denoted by lower-case letters, i.e. k_1 and k_2, while the $2n$-bit master key is interchangeably denoted by the capital letter K or the concatenation (k_1, k_2) (as the reader has seen).

An n-bit random permutation is a permutation that is uniformly selected from all $(2^n)!$ possible choices. In this work, the notation $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_j)$ is used to denote *a tuple of random permutations* ($j = 15$ in the context of IDEM_{15}, and $j = 7$ in the context of IDEM_7), and we let $\mathbf{P}$ provide a unified interface, i.e. $\mathbf{P}.\text{P}(i, \delta, z) := \{1, \ldots, j\} \times \{+, -\} \times \{0,1\}^n \to \{0,1\}^n$, i indicates the index, $\delta \in \{+, -\}$ indicates direct query or inverse query, and $z \in \{0,1\}^n$ is the queries value). On the other hand, the interface of the ideal cipher $\mathbf{E}$ is $\mathbf{E}.\text{E}(\delta, K, z) := \{+, -\} \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$.

Indifferentiability. The indifferentiability framework [MRH04] addresses the idealized construction in settings where the underlying parameters are exposed to the adversary. For concreteness, consider $\text{IDEM}_{15}^{\mathbf{P}}$: a distinguisher $\mathbf{D}^{\text{IDEM}_{15}^{\mathbf{P}}, \mathbf{P}}$ with oracle access to the cipher and the underlying primitives is trying to distinguish $\text{IDEM}_{15}^{\mathbf{P}}$ from $\mathbf{E}$. Then the formal definition is as follows.

Definition 1 (Indifferentiability). *The idealized blockcipher* $\text{IDEM}_{15}^{\mathbf{P}}$ *with oracle access to ideal primitives* $\mathbf{P}$ *is said to be statistically and strongly* $(q, \sigma, t, \varepsilon)$*-indifferentiable from an ideal cipher* $\mathbf{E}$ *if there exists a simulator* $\mathbf{S}^{\mathbf{E}}$ *s.t.* $\mathbf{S}$ *makes at most* σ *queries to* $\mathbf{E}$*, runs in time at most* t*, and for any distinguisher* $\mathbf{D}$ *which issues at most* q *queries, it holds*

$$\left| Pr[\mathbf{D}^{\text{IDEM}_{15}^{\mathbf{P}}, \mathbf{P}} = 1] - Pr[\mathbf{D}^{\mathbf{E}, \mathbf{S}^{\mathbf{E}}} = 1] \right| \leq \varepsilon$$

Such a result means that $\text{IDEM}_{15}^{\mathbf{P}}$ can safely replace $\mathbf{E}$ in most "natural" settings – although this belief does not necessarily hold when the resource of the adversary is limited [RSS11,DGHM13]. Since introduced, indifferentiability framework has been applied to various constructions, including variants of Merkle-Damgård, Feistel [CHK+14], Sponge [BDPVA08], and IEM [ABD+13,LS13].

To formally define seq-indifferentiability, we first specify a restricted distinguisher class, namely the *sequential distinguisher* (*seq-distinguisher*) [MPS12]. Consider $\text{IDEM}_7^{\mathbf{P}}$ and $D^{\text{IDEM}_7^{\mathbf{P}},\mathbf{P}}$. D is *sequential* if it issues queries in a specific order: (1) queries the underlying primitives $\mathbf{P}$ as it wishes; (2) queries $\text{IDEM}_7^{\mathbf{P}}$ as it wishes; (3) outputs, and cannot query $\mathbf{P}$ again. This order is illustrated in the italic numbers in Fig. 3. We then define the notion *total oracle query cost* of D, which equals the total number of queries received by $\mathbf{P}$ (from D or $\text{IDEM}_7^{\mathbf{P}}$) when D interacts with $(\text{IDEM}_7^{\mathbf{P}}, \mathbf{P})$ [MPS12]. Then, the definition of seq-indifferentiability can be obtained by tweaking the definition of (full) indifferentiability by restricting the distinguisher to the range of sequential ones, and replacing the query cost of the distinguisher by the *total oracle query cost*.

Definition 2 (Seq-indifferentiability). *The idealized blockcipher* $\text{IDEM}_7^{\mathbf{P}}$ *with oracle access to ideal primitives* $\mathbf{P}$ *is said to be statistically and strongly* $(q, \sigma, t, \varepsilon)$*-seq-indifferentiable from an ideal cipher* $\mathbf{E}$ *if there exists a simulator* $\mathcal{S}^{\mathbf{E}}$ *s.t.* $\mathcal{S}$ *makes at most* σ *queries to* $\mathbf{E}$*, runs in time at most* t*, and for any sequential distinguisher* D *of total oracle query cost at most* q*, it holds*

$$\left| Pr[D^{\text{IDEM}_7^{\mathbf{P}},\mathbf{P}} = 1] - Pr[D^{\mathbf{E},\mathcal{S}^{\mathbf{E}}} = 1] \right| \leq \varepsilon$$

Seq-indifferentiability – although weaker than indifferentiability [MRH04] – is already proved (by [MPS12,CS15]) sufficient to imply correlation intractability in the idealized model. The notion *correlation intractability* was introduced by Canetti et al. [CGH04] to capture the feature that there is no exploitable relation between the inputs and outputs of the target function ensembles. It was transposed to idealized models to guarantee similar feature on idealized constructions (e.g. $\text{IDEM}_7^{\mathbf{P}}$). To formally define this notion, we first give the definition (from [CS15]) of evasive relation.

Definition 3 (Evasive Relation). *A relation* $\mathcal{R}$ *over pairs of binary sequences is said* (q, ϵ)*-evasive with respect to an ideal cipher* $\mathbf{E}$ *with* n*-bit blocks, if for any oracle Turing machine* $\mathcal{M}$ *issuing at most* q *oracle queries, it holds*

$$Pr[(x_1, \ldots, x_m) \leftarrow \mathcal{M}^{\mathbf{E}}(1^n) : ((x_1, \ldots, x_m), (\mathbf{E}(x_1), \ldots, \mathbf{E}(x_m))) \in \mathcal{R}] \leq \epsilon.$$

Then is the correlation intractability itself.

Definition 4 (Correlation Intractability). *Let* $\mathcal{R}$ *be an* m*-ary relation. Then, an idealized blockcipher* $\text{IDEM}_7^{\mathbf{P}}$ *with oracle access to ideal primitives* $\mathbf{P}$ *is said to be* (q, ϵ)*-correlation intractable with respect to* $\mathcal{R}$*, if for any oracle Turing machine* $\mathcal{M}$ *issuing at most* q *oracle queries, it holds*

$$Pr[(x_1, \ldots, x_m) \leftarrow \mathcal{M}^{\mathbf{P}}(1^n) : ((x_1, \ldots, x_m), (\text{IDEM}_7^{\mathbf{P}}(x_1), \ldots, \text{IDEM}_7^{\mathbf{P}}(x_m))) \in \mathcal{R}] \leq \epsilon.$$

Seq-indifferentiability implies correlation intractability: (Theorem 4 from [CS15])

Theorem 1. *For an idealized blockcipher* $\mathrm{IDEM}_r^{\mathbf{P}}$ *which has oracle access to ideal primitives* $\mathbf{P}$ *and makes at most* r *queries to* $\mathbf{P}$ *on any input,*[6] *if* $\mathrm{IDEM}_r^{\mathbf{P}}$ *is* $(q+rm, \sigma, \epsilon)$*-seq-indifferentiable from* $\mathbf{E}$*, then for any* m*-ary relation* $\mathcal{R}$ *which is* $(\sigma+m, \epsilon_{\mathcal{R}})$*-evasive with respect to* $\mathbf{E}$*,* $\mathrm{IDEM}_r^{\mathbf{P}}$ *is* $(q, \epsilon+\epsilon_{\mathcal{R}})$*-correlation intractable with respect to* $\mathcal{R}$*.*

When the primitive implemented by the seq-indifferentiable construction is *stateless*, seq-indifferentiability implies *public indifferentiability* – indifferentiability from the target primitive in the setting where all the queries to it are *public* (a notion due to [YMO08,DRS09]).

3 Indifferentiability for 15-round IDEM

We prove the first main theorem of this paper in this section, which is:

Theorem 2. *The 15-round Even-Mansour cipher* IDEM_{15} *from fifteen independent random permutations* $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_{15})$ *and two* n*-bit keys* (k_1, k_2) *alternatively xored is strongly and statistically* $(q, \sigma, t, \varepsilon)$*-indifferentiable from an ideal cipher* $\mathbf{IC}[2n, n]$*, where* $\sigma = 2^{10} \cdot q^8$*,* $t = O(q^8)$*, and* $\varepsilon \leq \frac{2^{11} \cdot q^8}{2^n} + \frac{2^{14} \cdot q^6}{2^n} = O(\frac{q^8}{2^n})$*.*

As usual, we first present the simulator, then sketch the proof.

3.1 The Simulator

To build the simulator, we borrow a variant of the *explicit randomness* technique [CHK+14] from [CS15], that is, letting the simulator $\mathbf{S}$ query $\mathbf{P}$ as explicit randomness. We denote by $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ the simulator for IDEM_{15} which takes $\mathbf{P}$ as randomness source and interacts with $\mathbf{E}$. $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ provides an interface $\mathbf{S}.\mathrm{P}(i, \delta, z)$ ($i \in \{1, \ldots, 15\}$) which is exactly the same as $\mathbf{P}$. As argued [ABD+13,CS15], using such explicit randomness is actually equivalent to lazily sampling in advance before the experiment.

We now give a high-level overview of the simulator $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ (depicted in Fig. 1 (left)). $\mathbf{S}$ maintains a history for each simulated permutation under the form of fifteen sets $P_1, \ldots, P_{15}$. Each of the sets has entries in the form of (x, y) for $x, y \in \{0,1\}^n$. $\mathbf{S}$ will ensure that for any $z \in \{0,1\}^n$ and $i \in \{1, \ldots, 15\}$, there is *at most one* $z' \in \{0,1\}^n$ such that $(z, z') \in P_i$, and vice versa; *once such consistency cannot be kept,* $\mathbf{S}$ *aborts* (will be discussed later). By this, the sets $\{P\} = \{P_1, \ldots, P_{15}\}$ are expected to define fifteen *partial permutations*, and we denote by P_i^+ (P_i^-, resp.) the (time-dependent) set of all n-bit values x (y, resp.) satisfying that $\exists z \in \{0,1\}^n$ s.t. $(x, z) \in P_i$ ($(z, y) \in P_i$, resp.); denote by $P_i^+(x)$ ($P_i^-(y)$, resp.) the corresponding value of z (as mentioned *the uniqueness of* z *is ensured by* $\mathbf{S}$).

[6] Note that in this case, if the seq-distinguisher D makes q_p queries to $\mathbf{P}$ and q_e queries to IDEM_r, then the *total oracle query cost* of D is $q_p + r \cdot q_e$. For $\mathrm{IDEM}_7^{\mathbf{P}}$ studied in this work, $r = 7$.

Queries that have already appeared in the history will be instantly answered with the contents in $\{P\}$. Upon a new query $\mathbf{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(i,\delta,z)$, $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ queries $\mathbf{P}$ to draw $z' = \mathbf{P}.\mathrm{P}(i,\delta,z)$ as the answer and calls a procedure $\textsc{ForceVal}(z,z',\delta,i)$ to add z and z' to P_i – inside this procedure, if z' is found already in $P_i^{\overline{\delta}}$, $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ aborts due to the broken consistency (as mentioned). Then, if $(i,\delta) \in \{(2,+),(6,-),(10,+),(14,-)\}$ it satisfies the *chain detection conditions*, so that $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ enqueues and completes chains formed by previous queries to ensure that it is ready to simulate answers consistent with those of $\mathbf{E}$ in the future.

The cases (i,δ) equals $(2,+)$ and $(14,-)$ are similar: taking the former $\mathrm{P}(2,+,x_2)$ as an example, $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ considers all tuples $(x_1,y_1,x_{14},y_{14},x_{15},y_{15})$ such that $(x_j,y_j) \in P_j$ for $j \in \{1,14,15\}$, recovers two keys $k_2 := y_1 \oplus x_2$ and $k_1 := y_{14} \oplus x_{15}$, computes $y_0 := x_1 \oplus k_1$ and $x_{16} := y_{15} \oplus k_2$, checks whether $\mathbf{E}.\mathrm{E}(+,(k_1,k_2),y_0) = x_{16}$ via an inner procedure $\mathbf{S}.\textsc{Check}$ and enqueues a 5-tuple $(y_0,k_1,k_2,0,4)$ into a queue *ChainQueue* when this is the case. In this tuple, the 4-th value 0 informs $\mathbf{S}$ that the first value of the tuple is y_0, and the last value 4 describes that when completing the chain characterized by the tuple $(y_0,k_1,k_2,0)$, $\mathbf{S}$ should add the adapted pair to P_4 to ensure consistency with $\mathbf{E}$. The action towards answering new query $\mathrm{P}(14,-,y_{14})$ is symmetric: $\mathbf{S}$ considers all tuples $(x_1,y_1,x_2,y_2,x_{15},y_{15})$ such that $(x_j,y_j) \in P_j$ for $j \in \{1,2,15\}$, recovers the two keys, calls $\mathbf{S}.\textsc{Check}$ and enqueues $(y_0,k_1,k_2,0,12)$ into *ChainQueue* when $\textsc{Check}$ returns true. The chain represented by this 5-tuple will be adapted at P_{12}, which is different from the case $(i,\delta) = (2,+)$.

The other two cases $\mathrm{P}(6,-,y_6)$ and $\mathrm{P}(10,+,x_{10})$ are similar by symmetry: in each case, $\mathbf{S}$ considers all tuples $(x_7,y_7,x_8,y_8,x_9,y_9)$ such that $(x_j,y_j) \in P_j$ for $j \in \{7,8,9\}$, computes $k_1 := y_8 \oplus x_9$ and $k_2 := y_7 \oplus x_8$, checks whether $x_7 \oplus k_1 = y_6 \wedge y_9 \oplus k_2 \in P_{10}^+$ (in case $\mathrm{P}(6,-,y_6)$) or $x_7 \oplus k_1 \in P_6^- \wedge y_9 \oplus k_2 = x_{10}$ (in case $\mathrm{P}(10,+,x_{10})$), and enqueues $(y_7,k_1,k_2,7,l)$ into *ChainQueue* when this is the case, where $l = 4$ in case $\mathrm{P}(6,-,y_6)$ and $l = 12$ in case $\mathrm{P}(10,+,x_{10})$.

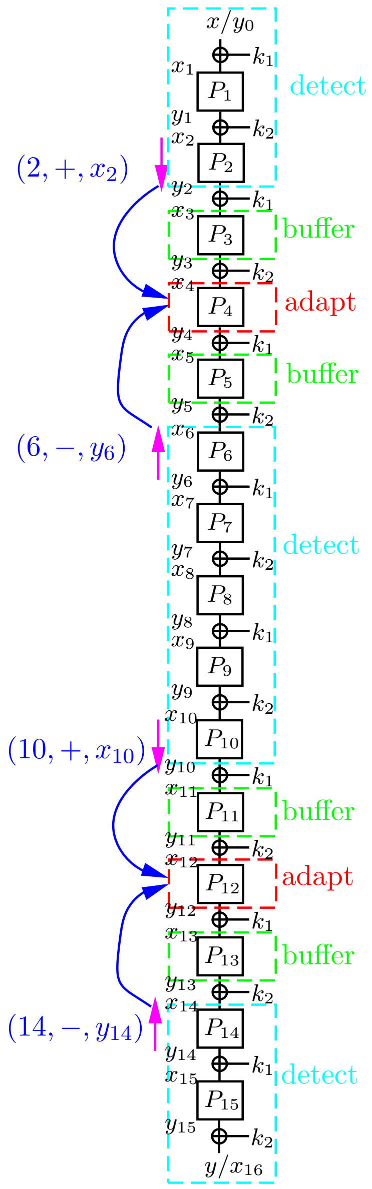
After enqueuing, $\mathbf{S}$ starts an execution of $\textsc{RecursiveCompletion}$, during which it continues taking the tuples out of *ChainQueue* and completing the associated partial chains till *ChainQueue* is empty again. More clearly, for each chain C dequeued from the queue, $\mathbf{S}$ evaluates in the IDEM_{15} computation path both forward and backward and queries $\mathbf{E}$ once to "wrap" around, until obtaining x_l (when moving forward) or y_l (when moving backward). $\mathbf{S}$ then calls $\textsc{ForceVal}(x_l,y_l,\perp,l)$ to add (x_l,y_l) to P_l as a newly defined pair of IO, so that the entire computation path is consistent with the answers of $\mathbf{E}$. Inside this call to $\textsc{ForceVal}$, if $x_l \in P_l^+$ or $y_l \in P_l^-$ before they are to be added, $\mathbf{S}$ aborts (also as mentioned).

During the completion of a chain, $\mathbf{S}$ adds new entries to P_i which are necessary for its evaluation. Such new values also trigger new chains to be enqueued when they satisfy the chain detection conditions mentioned before. For this, note that $\mathbf{S}$ *continues* dequeuing and completing chains till *ChainQueue* is empty again. To avoid re-completing the same chain, $\mathbf{S}$ maintains a set *CompSet* to keep a record of what it has completed, and a chain C dequeued from the queue will be completed only if $C \notin CompSet$. After all the works above are finished, $\mathbf{S}$ answers the original query with $P_i^\delta(z)$.

Note that throughout the process, the entries in $\mathbf{S}.\{P\}$ are never overwritten; once $\mathbf{S}$ finds itself unable to maintain consistency any more, $\mathbf{S}$ just aborts.

The pseudocode of $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ and a modified simulator $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$ (please see Sect. 3.3) is presented as follows. When a line has a boxed variant next to it, $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ uses the original code, whereas $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$ uses the boxed one.

```
1: **Simulator** $\mathbf{S}^{\mathbf{E},\mathbf{P}}$:                    $\boxed{\textbf{Simulator } \widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}:}$
2: **Variables**
3:   Sets $\{P\} = \{P_1, \ldots, P_{15}\}$ and $CompSet$, initially empty
4:   Queue $ChainQueue$, initially empty
5: **public procedure** P$(i, \delta, z)$
6:   $z' :=$ InnerP$(i, \delta, z)$ // Chains are enqueued in this step
7:   RecursiveCompletion()
8:   **return** $z'$
9: // The recursive completion process is extracted as an individual procedure.
10: **private procedure** RecursiveCompletion()
11:   **while** $ChainQueue \neq \emptyset$ **do**
12:     $(y_j, k_1, k_2, j, l) := ChainQueue$.Dequeue()
13:     **if** $(y_j, k_1, k_2, j) \notin CompSet$ **then**
14:       Complete$(y_j, k_1, k_2, j, l)$
15: // The "inner" permutation interface used by $\mathbf{S}$ itself.
16: **private procedure** InnerP$(i, \delta, z)$
17:   **if** $z \notin P_i^\delta$ **then**
18:     $z' := \mathbf{P}$.P$(i, \delta, z)$
19:     ForceVal$(z, z', \delta, i)$
20:     EnqueueChains$(i, \delta, z)$
21:   **return** $P_i^\delta(z)$
22: // Procedure that enqueues chains.
23: **private procedure** EnqueueChains$(i, \delta, z)$
24:   **if** $(i, \delta) = (2, +)$ **then**
25:     **forall** $((x_1, y_1), x_2, y_{14}, (x_{15}, y_{15})) \in P_1 \times \{z\} \times P_{14}^- \times P_{15}$ **do**
26:       $k_2 := y_1 \oplus x_2$
27:       $k_1 := y_{14} \oplus x_{15}$
28:       $y_0 := x_1 \oplus k_1$
29:       $x_{16} := y_{15} \oplus k_2$
30:       $flag :=$ Check$(y_0, x_{16}, (k_1, k_2))$        $\boxed{flag := \widetilde{E}^{\mathbf{E}}.\textsc{Check}(y_0, x_{16}, (k_1, k_2))}$
31:       **if** $flag = true$ **then**
32:         $ChainQueue$.Enqueue$(y_0, k_1, k_2, 0, 4)$
33:   **else if** $(i, \delta) = (14, -)$ **then**
34:     **forall** $((x_1, y_1), x_2, y_{14}, (x_{15}, y_{15})) \in P_1 \times P_2^+ \times \{z\} \times P_{15}$ **do**
35:       $k_2 := y_1 \oplus x_2$
36:       $k_1 := y_{14} \oplus x_{15}$
37:       $y_0 := x_1 \oplus k_1$
38:       $x_{16} := y_{15} \oplus k_2$
39:       $flag :=$ Check$(y_0, x_{16}, (k_1, k_2))$        $\boxed{flag := \widetilde{E}^{\mathbf{E}}.\textsc{Check}(y_0, x_{16}, (k_1, k_2))}$
40:       **if** $flag = true$ **then**
```



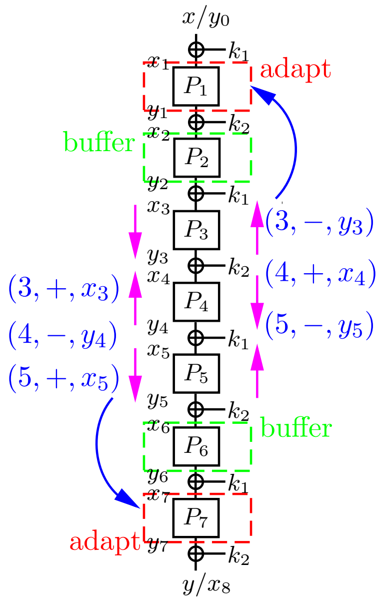


Fig. 1. (left) IDEM_{15} with the zones where the simulator detects chains and adapts them; (right) IDEM_7 and how the simulator for sequential indifferentiability works.

```
      $ChainQueue.\textsc{Enqueue}(y_0, k_1, k_2, 0, 12)$
  **else if** $(i, \delta) = (6, -) \vee (i, \delta) = (10, +)$ **then**
    **forall** $((x_7, y_7), (x_8, y_8), (x_9, y_9)) \in P_7 \times P_8 \times P_9$ **do**
      $k_1 := y_8 \oplus x_9$
      $k_2 := y_7 \oplus x_8$
      **if** $(i, \delta) = (6, -) \wedge z = x_7 \oplus k_1 \wedge y_9 \oplus k_2 \in P_{10}^{+}$ **then**
        $ChainQueue.\textsc{Enqueue}(y_7, k_1, k_2, 7, 4)$
      **else if** $z = y_9 \oplus k_2 \wedge x_7 \oplus k_1 \in P_6^{-}$ **then** // $(i, \delta) = (10, +)$
        $ChainQueue.\textsc{Enqueue}(y_7, k_1, k_2, 7, 12)$
**private procedure** $\textsc{Complete}(y_j, k_1, k_2, j, l)$
  $(y_{l-1}, k_1, k_2, l-1) := \textsc{Eval}\mathrm{FWD}(y_j, k_1, k_2, j, l-1)$
  $(y_l, k_1, k_2, l) := \textsc{Eval}\mathrm{BWD}(y_j, k_1, k_2, j, l)$
  $\textsc{ForceVal}(y_{l-1} \oplus k_2, y_l, \perp, l)$ // Always $k_2$, since $l \in \{4, 12\}$.
  $(y_0, k_1, k_2, 0) := \textsc{Eval}\mathrm{FWD}(y_j, k_1, k_2, j, 0)$
  $(y_7, k_1, k_2, 7) := \textsc{Eval}\mathrm{FWD}(y_0, k_1, k_2, 0, 7)$
  $CompSet := CompSet \cup \{(y_0, k_1, k_2, 0), (y_7, k_1, k_2, 7)\}$
// Procedure that adds entries to $\{P\}$.
**private procedure** $\textsc{ForceVal}(z, z', \delta, l)$
  // When $\delta = \perp$ then it's an adaptation
  **if** $z \in P_l^{\delta} \vee z' \in P_l^{\overline{\delta}}$ **then abort**
  **else if** $\delta = -$ **then** $P_l := P_l \cup \{(z', z)\}$
  **else** $P_l := P_l \cup \{(z, z')\}$ // $\delta = +$ or $\perp$
**private procedure** $\textsc{Check}(x, y, K)$ // $\widetilde{S}$ does not own such a procedure
  **return** $\mathbf{E}.\mathrm{E}(+, K, x) = y$
// Two procedures that help evaluate forward and backward respectively in IDEM.
**private procedure** $\textsc{Eval}\mathrm{FWD}(y_j, k_1, k_2, j, l)$
  **while** $j \neq l$ **do**
    **if** $j = 15$ **then**
      $x_{16} := y_{15} \oplus k_2$
      $y_0 := \mathbf{E}.\mathrm{E}(-, (k_1, k_2), x_{16})$      $\boxed{y_0 := \widetilde{E}^{\mathbf{E}}.\mathrm{E}(-, (k_1, k_2), x_{16})}$
      $j := 0$
    **else**
      **if** $j$ is odd **then**
        $y_{j+1} := \textsc{Inner}\mathrm{P}(j+1, +, y_j \oplus k_2)$
      **else**
        $y_{j+1} := \textsc{Inner}\mathrm{P}(j+1, +, y_j \oplus k_1)$
      $j := j + 1$
  **return** $(y_l, k_1, k_2, l)$
**private procedure** $\textsc{Eval}\mathrm{BWD}(y_j, k_1, k_2, j, l)$
  **while** $j \neq l$ **do**
    **if** $j = 0$ **then**
      $x_{16} := \mathbf{E}.\mathrm{E}(+, (k_1, k_2), y_0)$      $\boxed{x_{16} := \widetilde{E}^{\mathbf{E}}.\mathrm{E}(+, (k_1, k_2), y_0)}$
      $y_{15} := x_{16} \oplus k_2$
      $j := 15$
    **else**
      **if** $j$ is odd **then**
        $y_{j-1} := \textsc{Inner}\mathrm{P}(j, -, y_j) \oplus k_1$
      **else**
```

```
89:         $y_{j-1} := \textsc{InnerP}(j, -, y_j) \oplus k_2$
90:       $j := j - 1$
91:   **return** $(y_l, k_1, k_2, l)$
```

As in previous works, to show the indifferentiability, we have to achieve the following two sub-goals for any fixed, deterministic,[7] and computationally unbounded distinguisher $\mathbf{D}$:

(i) the simulated system $\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})$ and the real system $\Sigma_3(\mathrm{IDEM}_{15}^{\mathbf{P}}, \mathbf{P})$ are indistinguishable;
(ii) $\mathbf{S}$ works with polynomial complexity, except with negligible probability.

3.2 Distinguisher that Completes All Chains

As an almost standard step in indifferentiability proofs, we introduce *distinguisher which completes all chains*. More clearly, fix a deterministic distinguisher $\mathbf{D}$ which issues at most q queries, and consider a distinguisher $\overline{D}$ which first runs $\mathbf{D}$ and then emulates a call to $\textsc{EvalFWD}(x, k_1, k_2, 0, 15)$ (resp. $\textsc{Evalbwd}(y \oplus k_2, k_1, k_2, 15, 0)$) for all queries $\mathrm{E}(+, (k_1, k_2), x)$ (resp. $\mathrm{E}(-, (k_1, k_2), y)$) issued by $\mathbf{D}$ and outputs whatever $\mathbf{D}$ outputs: clearly $\overline{D}$ has exactly the same advantage as $\mathbf{D}$ in distinguishing Σ_1 and Σ_3, and all the rest arguments concentrate on $\overline{D}$.[8] Limiting $\mathbf{D}$ to deterministic ones is *wlog* since the advantage of a probabilistic distinguisher cannot exceed the corresponding deterministic version with the best random coins [ABD+13].

3.3 Intermediate System

We use an intermediate system $\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}})$, which consists of a modified ideal cipher $\widetilde{E}^{\mathbf{E}}$ and an also slightly modified simulator $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$. $\widetilde{E}^{\mathbf{E}}$ maintains a set ES to keep the history of queries it has received, which contains entries of the form $(x, y, K) \in \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{2n}$. $\widetilde{E}^{\mathbf{E}}$ provides an additional interface $\textsc{Check}$ to the simulator. Once being called on $\textsc{Check}(x, y, K)$, $\widetilde{E}^{\mathbf{E}}$ looks in ES to check whether $(x, y, K) \in ES$ and returns the answer. The modified ideal cipher $\widetilde{E}^{\mathbf{E}}$ is implemented as follows. On the other hand, the pseudocode of $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$ has been presented along with $\mathbf{S}^{\mathbf{E},\mathbf{P}}$, in subsection 3.1.

```
**Modified ideal cipher** $\widetilde{E}^{\mathbf{E}}$:
**Variables**
  Set $ES$, initially empty
**public procedure** $\mathrm{E}(\delta, K, z)$
  **if** $(K, z) \notin ES^{\delta}$ **then**
    $z' := \mathbf{E}.\mathrm{E}(\delta, K, z)$
```

[7] This is *wlog* since the advantage of any probabilistic distinguisher cannot exceed the advantage of the corresponding optimal deterministic version.

[8] Here we warn the readers that although the main body *seems* to focus on the original $\mathbf{D}$, the formal proof actually focus on the corresponding $\overline{D}$. But it is clear that all the bounds for $\overline{D}$ also hold for the original $\mathbf{D}$.

```
        if $\delta = +$ then $ES := ES \cup \{(z, z', K)\}$
        else $ES := ES \cup \{(z', z, K)\}$ // $\delta = -$
      return $ES^{\delta}(K, z)$
    public procedure CHECK$(x, y, K)$
      if $(x, y, K) \in ES$ then return true
      return false
```

To simplify notations and highlight the randomness source, in the following sections: $\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})$ may be referred by $\Sigma_1(\mathbf{E}, \mathbf{P})$, or even Σ_1 if the randomness source is not important; $\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}})$ may be referred by $\Sigma_2(\mathbf{E}, \mathbf{P})$, $\Sigma_2(\widetilde{E}, \widetilde{S})$, $\Sigma_2(\alpha)$ when $\alpha = (\mathbf{E}, \mathbf{P})$ or Σ_2; and $\Sigma_3(\mathrm{IDEM}_{15}^{\mathbf{P}}, \mathbf{P})$ may be referred by $\Sigma_3(\mathbf{P})$ and Σ_3. The three systems are depicted in Fig. 2.

Since all the entries of ES indeed come from (an ideal cipher) $\mathbf{E}$, ES always defines a partial cipher, and we use a notation system similar to that for $\{P\}$. More clearly, we denote by ES^+ the set of tuples (K, x) s.t. $\exists y : (x, y, K) \in ES$, and denote by $ES^+(K, x)$ the corresponding y. Similarly for ES^- and $ES^-(K, y)$. Finally, denote by $|\widetilde{E}.ES|$ the size of $\widetilde{E}.ES$.

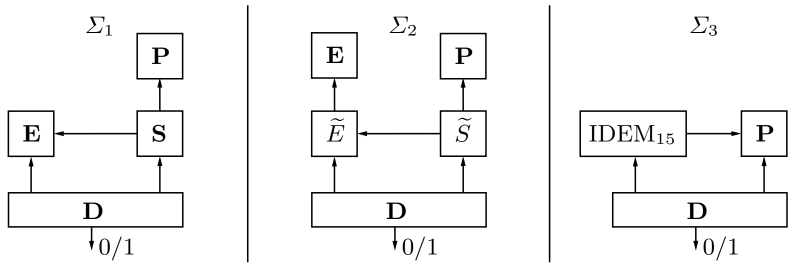


Fig. 2. Systems used in the indifferentiability proof for IDEM$_{15}$.

3.4 Bounding the Complexity of $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$ in Σ_2

In this section we show that the simulator $\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}}$ in Σ_2 works with polynomial complexity. The idea is inherited from [CHK+14], with a bit novelty.

The first step is exactly the same as [CHK+14]: the number of chains completed due to the detection zones at the sides (P_1,P_2,P_{14},P_{15}) is at most q.

Lemma 1. *During the execution $\overline{D}^{\Sigma_2}$, $\widetilde{S}$ dequeues at most q times a tuple of the form $(y_0, k_1, k_2, 0, l)$ for which $(y_0, k_1, k_2, 0) \notin CompSet$.*

Proof. By construction, $(y_0, k_1, k_2, 0, l)$ can be enqueued only if $(y_0, x_{16}, (k_1, k_2))$ has already been in ES for the corresponding x_{16}. Moreover, this entry cannot have been added due to $\widetilde{S}$, since $\widetilde{S}$ only queries $\widetilde{E}$ during the completion of a chain, so that $(y_0, k_1, k_2, 0) \in CompSet$ when it is dequeued. So the entry $(y_0, x_{16}, (k_1, k_2))$ can only be added due to $\overline{D}$ querying $\widetilde{E}$ (at most q times). □

The second step is to bound the size of each set of $\widetilde{S}$. This step slightly deviates from [CHK+14]: we prove that the complexity of $\widetilde{S}$ is at a *lower* level *with overwhelming probability*, rather than at a *higher* level in *all* possible cases.

Consider the sets $\{P\}$ of $\widetilde{S}$ standing at the end of $\overline{D}^{\Sigma_2}$. As the beginning, note that by Lemma 1, for $i \in \{6, \ldots, 10\}$, $|P_i| \leq 2q$. The reason is that for such values of i, $|P_i|$ can only be enlarged by at most 1 when $\overline{D}$ queries $\mathrm{P}(i, \delta, z)$ or when $\widetilde{S}$ completes a chain $(y_0, k_1, k_2, 0, l)$ (at most q times, by Lemma 1). It can be easily seen that for a certain value of i, each query of $\mathbf{D}$ corresponds to at most one query from $\overline{D}$ to $\mathrm{P}(i, \delta, z)$, so that $|P_i| \leq q + q$ for $i \in \{6, \ldots, 10\}$.

Till now, the chains detected by the middle detection zone can be bounded to $\prod_{i=7}^{9} |P_i| \leq 8q^3$. But we take a step further to bound it to $O(q^2)$ rather than $O(q^3)$.[9] For this, we introduce a bad event BadLockMid,[10] which happens if any of the three sub-events BadLockMidA, BadLockMidB, or BadLockMidC happens:

Definition 5. *After a random assignment in P_7, P_8, or P_9, the event* BadLockMidA *happens if there are two 3-tuples $((x_7, y_7), (x_8, y_8), (x_9, y_9)) \in P_7 \times P_8 \times P_9$ and $((x'_7, y'_7), (x'_8, y'_8), (x'_9, y'_9)) \in P_7 \times P_8 \times P_9$ such that the following three are simultaneously fulfilled:*

- *the two tuples are "totally different", i.e. $x_7 \neq x'_7 \wedge x_8 \neq x'_8 \wedge x_9 \neq x'_9$;*
- $x_7 \oplus y_8 \oplus x_9 = x'_7 \oplus y'_8 \oplus x'_9$;
- $y_7 \oplus x_8 \oplus y_9 = y'_7 \oplus x'_8 \oplus y'_9$.

There are at most $|P_7|^2 \cdot |P_8|^2 \cdot |P_9|^2 \leq (2q)^6$ such pairs of tuples. For each such pair of tuples, the probability that the last random assignment (before this pair exists) leads to such a situation is at most $\frac{1}{2^n - 2q}$, so that $Pr[\mathsf{BadLockMidA}] \leq \frac{2^7 \cdot q^6}{2^n}$ (assuming $2q < 2^n/2$). To illustrate more clearly, consider two such tuples which satisfy the above constraints, and *wlog* assume that the last random assignment adds (x_7, y_7) to P_7. Then, if this assignment is forward, we've $Pr[\mathbf{P}.\mathrm{P}(7, +, x_7) = y_7 = x_8 \oplus y_9 \oplus y'_7 \oplus x'_8 \oplus y'_9] \leq \frac{1}{2^n - 2q}$; if it is backward, we've $Pr[\mathbf{P}.\mathrm{P}(7, -, y_7) = x_7 = y_8 \oplus x_9 \oplus x'_7 \oplus y'_8 \oplus x'_9] \leq \frac{1}{2^n - 2q}$.

Definition 6. *For $(i, j) \in \{(7, 8), (8, 9)\}$, after a random assignment in P_7, P_8, or P_9, the event* BadLockMidB *happens if there are two pairs $((x_i, y_i), (x_j, y_j)) \in P_i \times P_j$ and $((x'_i, y'_i), (x'_j, y'_j)) \in P_i \times P_j$ such that the following three are simultaneously fulfilled:*

- *the two pairs are "totally different", i.e. $x_i \neq x'_i \wedge x_j \neq x'_j$;*
- $x_i \oplus y_j = x'_i \oplus y'_j$;
- $y_i \oplus x_j = y'_i \oplus x'_j$.

[9] This, in fact, comes from an intuition: although the three middle sets form $O(q^3)$ possible chains, the two sets (P_6 and P_{10}) surrounding them only have size of $O(q)$. How could as many as $O(q^3)$ chains share as few as $O(q)$ endpoints?

[10] *Bad events due to a Lock of values in the Middle rounds.*

Since $|P_7|, |P_8|, |P_9| \leq 2q$, there are at most $2 \cdot (2q)^4$ such pairs of pairs. For each such pair, the probability that the last random assignment leads to such a situation is at most $\frac{1}{2^n - 2q}$, so that $Pr[\mathsf{BadLockMidB}] \leq \frac{2^6 \cdot q^4}{2^n}$ (assuming $2q < 2^n/2$). The analysis is similar to $\mathsf{BadLockMidA}$.

Definition 7. *After a random assignment in P_7 or P_9, the event* $\mathsf{BadLockMidC}$ *happens if there are two pairs $((x_7, y_7), (x_9, y_9)) \in P_7 \times P_9$ and $((x'_7, y'_7), (x'_9, y'_9)) \in P_7 \times P_9$ such that the following three are simultaneously fulfilled:*

- *the two pairs are "totally different", i.e. $x_7 \neq x'_7 \wedge x_9 \neq x'_9$;*
- *$x_7 \oplus x_9 = x'_7 \oplus x'_9$;*
- *$y_7 \oplus y_9 = y'_7 \oplus y'_9$.*

There are at most $(2q)^4$ such pairs of pairs. For each such pair, the probability that the last random assignment leads to such a situation is at most $\frac{1}{2^n - 2q}$, so that $Pr[\mathsf{BadLockMidC}] \leq \frac{2^5 \cdot q^4}{2^n}$. Therefore, $Pr[\mathsf{BadLockMid}] \leq \frac{2^8 \cdot q^6}{2^n}$.

Armed with the event $\mathsf{BadLockMid}$, we are now able to bound the number of chains due to the middle detection zone to $O(q^2)$.

Lemma 2. *During $\overline{D}^{\Sigma_2}$, if* $\mathsf{BadLockMid}$ *does not happen, then it holds:*

(i) for any pair (y_6, x_{10}), there exists at most one tuple $((x_7, y_7), (x_8, y_8), (x_9, y_9)) \in P_7 \times P_8 \times P_9$ such that $x_7 \oplus y_8 \oplus x_9 = y_6$ and $y_7 \oplus x_8 \oplus y_9 = x_{10}$;
(ii) $\widetilde{S}$ enqueues at most $4q^2$ times a tuple of the form $(y_7, k_1, k_2, 7, l)$.

Proof. For proposition (i), assume otherwise, i.e. there exists another tuple $((x'_7, y'_7), (x'_8, y'_8), (x'_9, y'_9)) \in P_7 \times P_8 \times P_9$ s.t. $(x_7, x_8, x_9) \neq (x'_7, x'_8, x'_9)$ and:

- $x_7 \oplus y_8 \oplus x_9 = y_6 = x'_7 \oplus y'_8 \oplus x'_9$; and:
- $y_7 \oplus x_8 \oplus y_9 = x_{10} = y'_7 \oplus x'_8 \oplus y'_9$.

We argue that $\mathsf{BadLockMid}$ happened before this pair of tuples exists:

- if $x_7 \neq x'_7 \wedge x_8 \neq x'_8 \wedge x_9 \neq x'_9$, then $\mathsf{BadLockMidA}$ happens after the last random assignment before this pair is in $\{P\}$;
- if $x_i = x'_i$ for $i = 7$, 8, or 9, then either $\mathsf{BadLockMidB}$ (in case $i = 7$ or 9) or $\mathsf{BadLockMidC}$ (in case $i = 8$) happened. For instance, if $x_7 = x'_7$, then it necessarily be $x_8 \neq x'_8$ and $x_9 \neq x'_9$, otherwise the constraints cannot be fulfilled. Conditioned on $x_7 = x'_7$, it can be derived that $y_8 \oplus x_9 = y'_8 \oplus x'_9$ and $x_8 \oplus y_9 = x'_8 \oplus y'_9$, and $\mathsf{BadLockMidB}$ happened when $((x_8, y_8), (x_9, y_9))$ and $((x'_8, y'_8), (x'_9, y'_9))$ were added to $\{P\}$.

These discussions establish proposition (i). Then, proposition (ii) immediately follows from $|P_6| \cdot |P_{10}| \leq 4q^2$. □

Finally, the complexity of $\widetilde{S}$ is bounded as follows.

Lemma 3. *If* $\mathsf{BadLockMid}$ *does not happen during the execution $\overline{D}^{\Sigma_2}$, then at the end of $\overline{D}^{\Sigma_2}$ it holds:*

– *for $i \in \{1, 2, 14, 15\}$, $|P_i| \leq 5q^2$; for $i \in \{3, 4, 5, 11, 12, 13\}$, $|P_i| \leq 6q^2$; and $|\widetilde{E}.ES| \leq 5q^2$;*
– *$\widetilde{S}$ issues at most $(5q^2)^4$ queries to $\widetilde{E}$.*Check.

Proof. For $i \in \{1, 2, 14, 15\}$, $|P_i|$ can only be enlarged by at most 1 when: $\overline{D}$ queries $\mathrm{P}(i, \delta, z)$ ($\leq q$, already discussed), or $\widetilde{S}$ completes a chain $(y_7, k_1, k_2, 7)$ ($\leq 4q^2$ if $\neg$BadLockMid).[11] Hence in total the bound is $q+4q^2 \leq 5q^2$. Whereas for $i \in \{3, 4, 5, 11, 12, 13\}$, the completion of a chain $(y_0, k_1, k_2, 0)$ ($\leq q$, by Lemma 1) may also enlarge $|P_i|$, so that the bound is $6q^2$.

Then, by construction, $|ES|$ can only be enlarged by at most 1 when $\overline{D}$ queries $\widetilde{E}$ ($\leq q$ times) or $\widetilde{S}$ completes a chain $(y_7, k_1, k_2, 7)$ ($\leq 4q^2$ times if $\neg$BadLockMid). Hence in total the bound is $q + 4q^2 \leq 5q^2$.

Finally, $\widetilde{S}$ calls $\widetilde{E}$.Check at most $|P_1| \cdot |P_2| \cdot |P_{14}| \cdot |P_{15}| \leq (5q^2)^4$ times. □

3.5 Transition from $\varSigma_1$ to $\varSigma_2$

This section gives the transition from $\varSigma_1$ to $\varSigma_2$. It consists of three steps: first, specifying a bad event BadCheck1 in $\varSigma_1$; second, under the condition that neither of the two bad events BadCheck1 and BadLockMid happens, showing that the two systems have the same behaviors when they share the same randomness source; third, upper bounding the complexity of $\mathbf{S}$ in $\varSigma_1$.

Bad Event BadCheck1, and No BadCheck1 Implies Same Behaviors of $\varSigma_1$ and $\varSigma_2$. Consider two systems $\varSigma_1(\mathbf{E}, \mathbf{P})$ and $\varSigma_2(\mathbf{E}, \mathbf{P})$ which take randomness from the same source $(\mathbf{E}, \mathbf{P})$. The only essential difference between they two lies in procedure Check: in $\varSigma_2$, the return value of a call $\widetilde{E}.\textsc{Check}(x, y, K)$ depends on $\widetilde{E}.ES$, i.e. the history of queries to $\widetilde{E}$, whereas in $\varSigma_1$ the return value of $\mathbf{S}.\textsc{Check}(x, y, K)$ completely depends on $\mathbf{E}$. For this, we define an event BadCheck1:[12] consider a pair of random primitives $(\mathbf{E}, \mathbf{P})$, BadCheck1 happens during the execution $\overline{D}^{\varSigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$ if $\exists (x, y, K)$ s.t. all the following hold:

(i) $\mathbf{S}^{\mathbf{E},\mathbf{P}}$ makes a call to $\textsc{Check}(x, y, K)$.
(ii) $\mathbf{E}.\mathrm{E}(+, K, x) = y$.
(iii) Before the call in (i), neither $\mathbf{E}.\mathrm{E}(+, K, x)$ nor $\mathbf{E}.\mathrm{E}(-, K, y)$ has been issued.

Consider the *transcripts* of queries and random answers appeared in the two systems, where the queries include E, P, and Check. More clearly, such transcripts include the following queries:

(i) in $\overline{D}^{\varSigma_2}$: all the $\widetilde{E}$.E, $\widetilde{E}$.Check, and $\mathbf{P}$.P queries issued by $\overline{D}$ and $\widetilde{S}$;

[11] Note that when $\widetilde{S}$ completes a chain $(y_0, k_1, k_2, 0)$, it does not add *new* entries to $\{P_1, P_2, P_{14}, P_{15}\}$, nor ES.
[12] The number 1 indicates that the event is defined for $\varSigma_1$. Similarly for BadLockEP2, which will be introduced.

(ii) in $\overline{D}^{\Sigma_1}$: all the $\mathbf{S}$.Check and $\mathbf{P}$.P queries issued by $\overline{D}$ and $\mathbf{S}$, and all the queries $\mathbf{E}$.E issued outside the Check procedure by $\overline{D}$ and $\mathbf{S}$.

Differences can only occur around Check, so that for a fixed pair of random primitives $(\mathbf{E}, \mathbf{P})$, it holds:

(i) during $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$, if BadLockMid does not happen, then $\widetilde{E}^{\mathbf{E}}$.Check is called at most $(5q^2)^4$ times;

(ii) conditioned on (i), if the return values of the first $(5q^2)^4$ calls to Check equal correspondingly in $\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$ and $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$, the transcript obtained by $(\overline{D}, \mathbf{S})$ in Σ_1 is the same as the transcript obtained by $(\overline{D}, \widetilde{S})$ in Σ_2, and $\overline{D}$ gives the same output.

The idea is formally captured by the following lemma and proof.

Lemma 4. $|Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})} = 1] - Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})} = 1]| \le \frac{2^{11} \cdot q^8}{2^n}$.

Proof. Consider the pair $(\mathbf{E}, \mathbf{P})$, and denote by BadLockEP2 the event that *BadLockMid happens during the Σ_2 execution* $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$. Then by Lemma 3, conditioned on $\neg$BadLockEP2, $\widetilde{E}^{\mathbf{E}}$.Check is called at most $(5q^2)^4$ times.

Now, conditioned on $\neg$BadLockEP2, we further assume that during $\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$, BadCheck1 does not happen in the first $(5q^2)^4$ calls to $\mathbf{S}$.Check. Then by an induction, the transcripts generated during $\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$ and $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$ are the same: the answers of Check equal correspondingly; the answers of E and P clearly equal since they are due to the same randomness source. Since $\overline{D}$ is deterministic, $\overline{D}$ gives the same output in the two executions.

Now, assume that $\mathbf{E}$ is queried $q^* < 2^n/2$ times during $\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$. Then the probability that BadCheck1 occurs in the first $(5q^2)^4$ calls is clearly no more than $(5q^2)^4/(2^n - q^*) \le 2 \cdot 5^4 \cdot q^8/2^n$. By this, it holds

$$\begin{aligned} &|Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})} = 1] - Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})} = 1]| \\ \le &Pr_{\mathbf{E},\mathbf{P}}[\mathsf{BadCheck1} \text{ occurs during } \overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})} \mid \neg\mathsf{BadLockEP2}] \\ &+ Pr_{\mathbf{E},\mathbf{P}}[\mathsf{BadLockEP2}] \le \frac{2 \cdot 5^4 \cdot q^8}{2^n} + \frac{2^8 \cdot q^6}{2^n} \le \frac{2^{11} \cdot q^8}{2^n} \end{aligned}$$

as claimed. □

Complexity of S. Lemma 4 leads to the termination argument for $\mathbf{S}$ (in Σ_1).

Lemma 5. *During a Σ_1 execution $\overline{D}^{\Sigma_1(\mathbf{E}, \mathbf{S}^{\mathbf{E},\mathbf{P}})}$, with probability at least $1 - \frac{2^{11} \cdot q^8}{2^n}$, $\mathbf{S}$ issues no more than $2^{10} \cdot q^8$ queries to $\mathbf{E}$, and runs in time no more than $O(q^8)$.*

Proof. $\mathbf{S}$ issues at most $|P_1| \cdot |P_2| \cdot |P_{14}| \cdot |P_{15}|$ queries to $\mathbf{E}$. By Lemma 4, the probability of the event that *the transcripts in* $\overline{D}^{\Sigma_1(\mathbf{E},\mathbf{S}^{\mathbf{E},\mathbf{P}})}$ *and* $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}},\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}})}$ *are the same and* $\mathsf{BadLockMid}$ *does not happen in* $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}},\widetilde{S}^{\widetilde{E}^{\mathbf{E}},\mathbf{P}})}$ is at least $1 - \frac{2^{11} \cdot q^8}{2^n}$, so that in $\overline{D}^{\Sigma_1(\mathbf{E},\mathbf{S}^{\mathbf{E},\mathbf{P}})}$, the bounds given in Lemma 3 holds with probability at least $1 - \frac{2^{11} \cdot q^8}{2^n}$, and the bound on queries is $(5q^2)^4 \leq 2^{10} \cdot q^8$. Moreover, consider the loops in $\mathbf{S}$. It is not hard to check that the most frequently executed ones are the two **forall** loops at line 25 and 34, which are executed at most $O(q^6) \cdot O(q^2)$ times. This yields the bound $O(q^8)$ on running time. □

3.6 Non-Abortion Argument for $\widetilde{S}$ in Σ_2

The formal transition from Σ_2 to Σ_3 also consists of three steps: first, specifying several bad events in Σ_2; second, showing that the absence of these events constitute a sufficient condition for $\widetilde{S}$ not aborts; third, showing that if $\widetilde{S}$ does not abort then Σ_2 and Σ_3 are indistinguishable. For clearness, they are divided into two subsections: this one contains non-abortion argument for $\widetilde{S}$, and the next one (3.7) contains the third step.

For further discussions, we introduce some necessary notions first.

Necessary Notions and Functions. As mentioned in the main body, in this work, the *partial chains* are characterized by 4-tuples $C = (y_i, k_1, k_2, i) \in \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \times \{0,\dots,15\}$; we further denote $C[1] = y_i$, $C[2] = k_1$, $C[3] = k_2$, and $C[4] = i$. A merit of this definition is that the set of all possible partial chains is time-independent. After carefully checking, we think the most notable drawback of this definition is that it is quite hard to formalize the "table-defined" notion with respect to it. However, as the reader will see, this work uses the notion *chains in the history* (will be introduced later) instead.

Then, consider a snapshot of the sets $\widetilde{E}.ES$ and $\widetilde{S}.\{P\}$ at some point in the execution $\overline{D}^{\Sigma_2}$: we borrow helper functions val_l^+ and val_l^- from [LS13] to help probe in the computation path defined in these sets. More clearly, val_l^+ and val_l^- move forward and backward respectively along the path formed by the entries of the sets, and return the input and output value of P_l respectively, or $\perp$, if the value is not computable due to the lack of some necessary entries. They are implemented as follows.

```
function $val_l^+(C)$
  // for $l = 1, \dots, 16$, returns $x_l$, if possible
  $j := C[4] + 1$
  if $C[4]$ is odd then $z := C[1] \oplus C[3]$
  else $z := C[1] \oplus C[2]$
  while $j \neq l$ do
    if $j \leq 15$ then
      if $z \notin P_j^+$ then return $\perp$
      if $j$ is odd then
        $z := P_j^+(z) \oplus C[3]$
      else // $j$ is even
        $z := P_j^+(z) \oplus C[2]$
      $j := j + 1$
    else // $j = 16$
      if $((C[2], C[3]), z) \notin ES^-$ then return $\perp$
      $z := ES^-((C[2], C[3]), z) \oplus C[2]$
      $j := 1$
  return $z$
function $val_l^-(C)$
  // for $l = 0, \dots, 15$, returns $y_l$, if possible
  $j := C[4]$
  $z := C[1]$
  while $j \neq l$ do
```

```
if $j \geq 1$ then
  if $z \notin P_j^-$ then return $\perp$
  if $j$ is odd then
    $z := P_j^-(z) \oplus C[2]$
  else // $j$ is even
    $z := P_j^-(z) \oplus C[3]$
    $j := j - 1$
else // $j = 0$
  if $((C[2], C[3]), z) \notin ES^+$ then return $\perp$
  $z := ES^+((C[2], C[3]), z) \oplus C[3]$
  $j := 15$
return $z$
```

The notion *equivalent* partial chains of [CHK+14] is also imported.

Definition 8. *Two partial chains* $C = (y_i, k_1, k_2, i)$ *and* $D = (y_j, k_1, k_2, j)$ *which share the same associated keys are* equivalent *(denoted* $C \equiv D$*) if* $val_j^-(C) = y_j$ *or* $val_i^-(D) = y_i$*.*[13]

By construction, each P_i always defines a partial permutation while ES always defines a partial blockcipher, so that the relation $\equiv$ is an equivalence relation.

History of Partial Chains. As mentioned in the main body, the bad events in $\overline{D}^{\Sigma_2}$ are due to the random answers from $(\mathbf{E}, \mathbf{P})$ hitting some values of the "active" chains. For this, we first formally define the notion "active", under the term *history for partial chains* $\mathcal{CH}$ for a random assignment (indifferently in ES or in $\{P\}$). Just before the random assignment, the *history for external partial chains* $\mathcal{ECH}$ is a set which includes all the tuples $(y_0, k_1, k_2, 0)$ such that $((k_1, k_2), y_0) \in ES^+$, while the *history for middle partial chains* $\mathcal{MCH}$ is a set which includes all the tuples $(y_7, k_1, k_2, 7)$ such that $y_7 \in P_7^-$, $x_8 = y_7 \oplus k_2 \in P_8^+$, and $x_9 = P_8^+(x_8) \oplus k_1 \in P_9^+$. Then $\mathcal{CH} = \mathcal{ECH} \cup \mathcal{MCH}$.[14]

Sufficient Condition for Non-abortion: Bad Events. At the beginning, note that all the discussions in the rest part of this subsection are made under the condition that *the event* BadLockMid *does not happen.* Under this condition, the bounds in Lemma 3 hold, and we have $|\mathcal{ECH}| \leq 5q^2$, $|\mathcal{MCH}| \leq 8q^3$, $|\mathcal{CH}| \leq 13q^3$.

The Random Values from $\mathbf{P}$ *Hit Adapted Values.* Some entries in P_4 and P_{12} are due to adaptation ($\textsc{forceval}(x, y, \perp, l)$). They may not be consistent with the values due to $\mathbf{P}$, so that during a later execution of $\textsc{Inner}\mathrm{P}(l, \delta, z)$, it might be that $\mathbf{P}.\mathrm{P}(l, \delta, z) \in P_l^{\overline{\delta}}$ and $\widetilde{S}$ aborts. The first event BadHitAdapt is due to this situation.

Definition 9. *The event* BadHitAdapt *occurs if when* $\widetilde{S}$ *queries* $\mathbf{P}$*, the obtained random value* $z' = \mathbf{P}.\mathrm{P}(i, \delta, z)$ *has already been in* $P_i^{\overline{\delta}}$*.*

Since $|P_4|$, $|P_{12}| \leq 6q^2$, the overall probability that BadHitAdapt occurs is at most $|P_4| \cdot \frac{|P_4|}{2^n - |P_4|} + |P_{12}| \cdot \frac{|P_{12}|}{2^n - |P_{12}|} \leq \frac{72q^4}{2^n - 6q^2}$.

[13] Note that if $C = D$ then both $y_i = val_i^-(D)$ and $y_j = val_j^-(C)$ hold.

[14] The middle chain set can also be defined as the 5-tuples formed by the entries in the five middle rounds. Conditioned on $\neg$BadLockMid, the size of such a set is only $O(q^2)$. However, this modification does not essentially improve the overall security bound, because at current time, the bottleneck is the $O(q^8)$ quantity brought in by Lemma 4 rather than the probability of the bad events in this part.

The Random Value from $\mathbf{E}$ *Unexpectedly Hits Entries in* $\{P\}$*.* An answer from $\mathbf{E}$ is bad if it hits any entry in P_1 or P_{15}. This is captured by BadE.

Definition 10. *The event* BadE *occurs if when* $\widetilde{E}$ *makes a query* $(+,(k_1,k_2),y_0)$ *to* $\mathbf{E}.\mathrm{E}$*, the answer satisfies* $\mathbf{E}.\mathrm{E}(+,(k_1,k_2),y_0)\oplus k_2 \in P_{15}^-$*; or, when* $\widetilde{E}$ *queries* $(-,(k_1,k_2),x_{16})$*, the answer satisfies* $\mathbf{E}.\mathrm{E}(-,(k_1,k_2),x_{16})\oplus k_1 \in P_1^+$*.*

Since both $|P_1|$ and $|P_{15}|$ are at most $5q^2$, $Pr[\mathsf{BadE}] \le |ES|\cdot\frac{5q^2}{2^n-|ES|} \le \frac{25q^4}{2^n-5q^2}$. Assume that BadE does not happen. Then it can be easily checked that for any chain C and any $i\le 14$, if $C\in\mathcal{CH}\wedge val_i^-(C)=\perp$ before a random forward assignment in ES, then $val_i^-(C)$ remains $\perp$ after this assignment. Similarly, for any chain C and any $i\ge 2$, if $C\in\mathcal{CH}\wedge val_i^+(C)=\perp$ before a random backward assignment in ES, then $val_i^+(C)$ remains $\perp$ after this assignment.

The Random Values from $\mathbf{P}$ *Unexpectedly Extend Active Chains.* Consider a chain $C\in\mathcal{CH}$. We expect that after a "random assignment with direction δ" in $\{P\}$, at most one value $val_i^{\delta}(C)$ changes from $\perp$ to non-empty (BadP), while no value $val_i^{\overline{\delta}}(C)$ changes from $\perp$ to non-empty ($\mathsf{BadInvP}$).

Definition 11. *The event* BadP *happens if one of the following happens in* $\overline{D}^{\Sigma_2}$*:*

- *(due to a random forward assignment) when* $\widetilde{S}$ *queries* $\mathbf{P}.\mathrm{P}(i,+,x_i)$*, there is a chain* $C=(y_j,k_1,k_2,j)\in\mathcal{CH}$ *such that* $val_i^+(C)=x_i$*, and after the subsequent random forward assignment in* P_i *it holds* $val_{i+1}^+(C)\in P_{i+1}^+$ *(in case* $i+1\le 15$*) or* $((k_1,k_2),val_{i+1}^+(C))\in ES^-$ *(in case* $i+1=16$*).*
- *(due to a random backward assignment) when* $\widetilde{S}$ *queries* $\mathbf{P}.\mathrm{P}(i,-,y_i)$*, there is a chain* $C=(y_j,k_1,k_2,j)\in\mathcal{CH}$ *such that* $val_i^-(C)=y_i$*, and after the subsequent random backward assignment in* P_i *it holds* $val_{i-1}^-(C)\in P_{i-1}^-$ *(in case* $i-1\ge 1$*) or* $((k_1,k_2),val_{i-1}^-(C))\in ES^+$ *(in case* $i-1=0$*).*

Since $|\mathcal{CH}|\le 13q^3$, there are *at most* $13q^3\cdot 15$ random assignments that are able to lead to BadP (indifferently forward or backward ones), each with probability at most $\frac{6q^2}{2^n-6q^2}$ since $|ES|,|P_i|\le 6q^2$ for any i.[15] Hence $Pr[\mathsf{BadP}]\le 13q^3\cdot 15\cdot\frac{6q^2}{2^n-6q^2}\le\frac{1170q^5}{2^n-6q^2}$.

Definition 12. *The event* $\mathsf{BadInvP}$ *happens if one of the following happens in* $\overline{D}^{\Sigma_2}$*:*

- *(due to a random forward assignment) when* $\widetilde{S}$ *queries* $\mathbf{P}.\mathrm{P}(i,+,x_i)$*, there is a chain* $C=(y_j,k_1,k_2,j)\in\mathcal{CH}$ *such that* $val_i^-(C)=\mathbf{P}.\mathrm{P}(i,+,x_i)$*.*
- *(due to a random backward assignment) when* $\widetilde{S}$ *queries* $\mathbf{P}.\mathrm{P}(i,-,y_i)$*, there is a chain* $C=(y_j,k_1,k_2,j)\in\mathcal{CH}$ *such that* $val_i^+(C)=\mathbf{P}.\mathrm{P}(i,-,y_i)$*.*

[15] For clearness, consider an example: for a forward one which adds $(x_1,\mathbf{P}.\mathrm{P}(1,+,x_1))$ to P_1 and $C=(y_j,k_1,k_2,j)\in\mathcal{CH}$, if $val_1^+(C)=x_1$, then the probability is $Pr[\mathbf{P}.\mathrm{P}(1,+,x_1)\oplus k_2\in P_2^+]\le\frac{|P_2|}{2^n-|P_1|}\le\frac{6q^2}{2^n-6q^2}$. The claim that $|P_i|\le 6q^2$ for any i follows from Lemma 3.

Since $|P_i| \le 6q^2$ for any i, there are at most $6q^2 \cdot 15$ possible random assignments in $\{P\}$, each (indifferently forward or backward) is able to lead to BadInvP with probability at most $\frac{|\mathcal{CH}|}{2^n-6q^2}$ (also consider an example: for a forward one which adds $(x_1, \mathbf{P}.\mathrm{P}(1,+,x_1))$ to P_1, the probability is $Pr[\exists C \in \mathcal{CH}$ s.t. $\mathbf{P}.\mathrm{P}(1,+,x_1) = val_1^-(C)] \le \frac{|\mathcal{CH}|}{2^n-|P_1|}$). Hence $Pr[\mathsf{BadInvP}] \le 6q^2 \cdot 15 \cdot \frac{13q^3}{2^n-6q^2} \le \frac{1170q^5}{2^n-6q^2}$.

Bad Events around the Middle Rounds. The event BadMidP concentrates on the random assignments in the 3 middle rounds.[16] In $\overline{D}^{\Sigma_2}$, consider a set of chains $\mathcal{LMC}$,[17] which consists of all the 5-tuples $(y_6, (x_7, y_7), (x_8, y_8), (x_9, y_9), x_{10}) \in P_6^- \times P_7 \times P_8 \times P_9 \times P_{10}^+$ such that $y_6 \oplus x_7 = y_8 \oplus x_9$ and $y_7 \oplus x_8 = y_9 \oplus x_{10}$.

Definition 13. *In $\overline{D}^{\Sigma_2}$, the event* BadMidP *happens if $|\mathcal{LMC}|$ is enlarged by any random assignment in P_7, P_8, or P_9.*

Consider a random assignment in P_7: if it is a forward one, then the probability that it leads to BadMidP is at most $\frac{|P_8|\cdot|P_9|\cdot|P_{10}|}{2^n-|P_7|} \le \frac{8q^3}{2^n-2q}$; if it is a backward one, then the probability is at most $\frac{|P_6|\cdot|P_8|\cdot|P_9|}{2^n-|P_7|} \le \frac{8q^3}{2^n-2q}$. Similarly for a random assignment in P_{10}. For random assignments in P_8, a forward one triggers BadMidP with probability at most $\frac{|P_6|\cdot|P_7|\cdot|P_9|}{2^n-|P_8|} \le \frac{8q^3}{2^n-2q}$, while a backward one triggers with probability at most $\frac{|P_{10}|\cdot|P_9|\cdot|P_7|}{2^n-|P_8|} \le \frac{8q^3}{2^n-2q}$. By the above, $Pr[\mathsf{BadMidP}] \le 3 \cdot (2q) \cdot \frac{8q^3}{2^n-2q} \le \frac{48q^4}{2^n-2q}$.

Colliding Chains. An answer from $\mathbf{P}$ is bad if two active chains C and D unexpectedly $val_i^\delta(C) = val_i^\delta(D)$ after a random assignment. This is captured by BadlyCollide (this term is borrowed from [CHK+14]).

Definition 14. *We say that event* BadlyCollide *happens if during $\overline{D}^{\Sigma_2}$, there exist two chains $C, D \in \mathcal{CH}$ such that with respect to a random forward assignment in $\{P\}$, the following three are simultaneously fulfilled:*

- *Before the assignment, C and D are not equivalent;*
- *Before the assignment, $val_l^+(C) \ne val_l^+(D)$, and $val_{l+1}^+(C) = \perp \vee val_{l+1}^+(D) = \perp$;*[18]
- *After the assignment, $val_{l+1}^+(C) = val_{l+1}^+(D) \ne \perp$.*

BadlyCollide *also happens if the following three are simultaneously fulfilled with respect to a random backward assignment in $\{P\}$:*

[16] It seems a bit inelegant to define three (actually four, if BadLockMid is counted) bad events around random assignment in $\{P\}$. However we really did not figure out an approach which is both elegant and able to result in such a low security bound.

[17] *Long Middle Chain.*

[18] Different from BadlyCollide in [CHK+14], here the two values are restricted to be obtained from the same direction.

– *Before the assignment, C and D are not equivalent;*
– *Before the assignment, $val_l^-(C) \neq val_l^-(D)$, and $val_{l-1}^-(C) = \bot \vee val_{l-1}^-(D) = \bot$;*
– *After the assignment, $val_{l-1}^-(C) = val_{l-1}^-(D) \neq \bot$.*

With respect to a fixed pair of chains $C, D \in \mathcal{CH}$ and a single random forward assignment, consider the probability of BadlyCollide (due to $val_{l+1}^+(C)$ suddenly equals $val_{l+1}^+(D)$). *Wlog* assume that $val_{l+1}^+(C) = \bot$ before this assignment. Then, conditioned on $\neg$BadP, this random assignment has to define $P_l^+(val_l^+(C))$; and after the assignment, it holds $val_{l+1}^+(C) = \mathbf{P}.\mathrm{P}(l, +, val_l^+(C)) \oplus k'$ where k' is the corresponding key. By this, $Pr[val_{l+1}^+(C) = val_{l+1}^+(D)] \leq \frac{1}{2^n - |P_l|} \leq \frac{1}{2^n - 6q^2}$ (as $val_{l+1}^+(D) \neq \bot$ after the assignment and $val_l^+(C) \neq val_l^+(D)$, $val_{l+1}^+(D) \neq \bot$ must already hold before it). The discussion for BadlyCollide due to random backward assignments is similar by symmetry. Since $\mathcal{CH} \leq 13q^3$, there are at most $(13q^3)^2$ pairs of chains (C, D); for each (C, D), there are at most 15 random assignments that are possible to lead to BadlyCollide (indifferently forward or backward). Hence $Pr[\text{BadlyCollide} \mid \neg\text{BadP}] \leq \frac{2535q^6}{2^n - 6q^2}$.

A tuple of primitives $(\mathbf{E}, \mathbf{P})$ is *good* if none of the seven bad events (BadLockMid, BadHitAdapt, BadE, BadP, BadInvP, BadMidP, BadlyCollide) happens during the Σ_2 execution $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$; and such Σ_2 executions are also *good*. The overall probability of the seven cumulates to $\frac{2^8 \cdot q^6}{2^n} + \frac{72q^4}{2^n - 6q^2} + \frac{25q^4}{2^n - 5q^2} + \frac{1170q^5}{2^n - 6q^2} + \frac{1170q^5}{2^n - 6q^2} + \frac{48q^4}{2^n - 2q} + \frac{2535q^6}{2^n - 6q^2} \leq \frac{2^{13.4} \cdot q^6}{2^n}$ (assuming $6q^2 < 2^n/2$). In the next paragraph, we will show that $\widetilde{S}$ does not abort during such good Σ_2 executions.

No Abortion in Good Executions: the Formal Proof. The abortions are mainly due to the adaptations of chains, hence this paragraph mainly focus on proving this type of abortion impossible. Following an old convention [CHK+14], we first exhibit basic properties of good executions, then prove that $\widetilde{S}$ does not abort due to adaptations, and finally present the main non-abortion argument (at the end of this paragraph). Note that all the lemmas below implicitly assume that in the Σ_2 execution, up to the point they focus on, $\widetilde{S}$ has not aborted.

Basic Properties. First, the relation $\equiv$ is invariant during random assignments.

Lemma 6. *During a good execution $\overline{D}^{\Sigma_2}$, two chains C, $D \in \mathcal{CH}$ are equivalent before any random assignment if and only if they are equivalent after the assignment.*

Proof. Since no entry in the sets will be overwritten, $C \equiv D$ before a random assignment trivially implies $C \equiv D$ after it. As to the backward direction, the case $C[4] = D[4]$ is trivial. *Wlog* assume that $C[4] = 0$ and $D[4] = 7$, and $val_0^-(D) \neq C[1] \wedge val_7^-(C) \neq D[1]$ before a random assignment while $val_0^-(D) = C[1]$ after it. Note that the random assignment is necessarily in $\{P\}$, since assignments in ES cannot suddenly lead to $val_0^-(D) = C[1]$; also, since no overwriting, it must be that there exists $j \in \{1,2,3,4,5\}$ such that the random assignment is in P_j and $val_j^+(C) \notin P_j^+ \wedge val_j^-(D) \notin P_j^-$ before the assignment. Then, if the assignment is a forward one, $\mathsf{BadInvP}$ happens with respect to D; if it is a backward one, $\mathsf{BadInvP}$ happens with respect to C. $\square$

Second, a chain C in $ChainQueue$ has been in $\mathcal{CH}$ even before it is enqueued.

Lemma 7. *During an execution $\overline{D}^{\Sigma_2}$, if at some point a chain C is enqueued during a call to* InnerP*, then it has been in $\mathcal{CH}$ right before this call to* InnerP*.*

Proof. This can be seen by construction: a chain $C = (y_0, k_1, k_2, 0)$ can be enqueued only if it has been in ES, while a chain $C = (y_7, k_1, k_2, 7)$ can be enqueued only if $y_7 \in P_7^-$ and $x_8 = y_7 \oplus k_2 \in P_8^+$ and $x_9 = P_8^+(x_8) \oplus k_1 \in P_9^+$. $\square$

Third, all the chains completed during the same execution of RecursiveCompletion are to be adapted at the same P_l.

Lemma 8. *During a good execution $\overline{D}^{\Sigma_2}$, we have:*

(i) Assume that a chain C is to be adapted at P_l. Then all the chains enqueued during the completion of C are also to be adapted at P_l;
(ii) All the chains completed during the same execution of RecursiveCompletion *are to be adapted at the same P_l.*

Proof. First, consider proposition (i). Consider any tuple $(C, 4)$ in $ChainQueue$ where $C = (y_0, k_1, k_2, 0)$. By construction, $val_{14}^-(C) \in P_{14}^-$ even before C was enqueued, so that during the execution of Complete$(C, 4)$, the subsequent call InnerP$(14, -, val_{14}^-(C))$ (in EvalBWD) will not enqueue any chain; only the subsequent call InnerP$(6, -, val_6^-(C))$ may enqueue chains. By construction, these chains are to be adapted at P_4, which is exactly the same as C.

Consider any tuple $(C, 4)$ in $ChainQueue$ where $C = (y_7, k_1, k_2, 7)$. By construction, $val_{10}^+(C) \in P_{10}^+$ even before C was enqueued, so that when C is completed, the subsequent call InnerP$(10, +, val_{10}^+(C))$ (in EvalFWD) will not enqueue any chain; only the subsequent call InnerP$(2, +, val_2^+(C))$ enqueues chains. These chains are to be adapted at P_4, which is exactly the same as C.

The other two cases $(y_0, k_1, k_2, 0, 12)$ and $(y_7, k_1, k_2, 7, 12)$ are similar by symmetry. These discussions establish proposition (i).

Then, *wlog* consider the case RecursiveCompletion$(2, +, x_2)$, in which $l = 4$. The initial call which led to RecursiveCompletion$(2, +, x_2)$ is of the form InnerP$(2, +, x_2)$, and all the tuples enqueued by this InnerP call are of the form $(y_0^j, k_1^j, k_2^j, 0, 4)$, so that by proposition (i), all the chains enqueued and completed during this execution are to be adapted at P_4. The other three cases where RecursiveCompletion is called with parameters $(6, -, y_6)$, $(10, +, x_{10})$, and $(14, -, y_{14})$ are similar. These establish proposition (ii). $\square$

The forth one is an implication of Lemma 8: a chain C will not suddenly be equivalent to a chain in $ChainQueue$.

Lemma 9. *During a good execution $\overline{D}^{\Sigma_2}$, let C be a partial chain which is enqueued at some time, and assume that no chain equivalent to C was enqueued before C is enqueued. Then from the point C is enqueued, at any point until C is dequeued, it is not possible that $D \equiv C$ for any chain D in ChainQueue.*

Proof. The case $C[4] = D[4]$ is trivial. For other cases, note that by construction, C and D must be enqueued during the same execution of RecursiveCompletion, so that by Lemma 8 (ii), they are to be adapted at the same P_l. *Wlog* assume that $l = 4$, $D[4] = 0$, and $C[4] = 7$. Also assume that $D[2] = C[2]$ and $D[3] = C[3]$, since otherwise they are never equivalent. Then if $val_7^-(D) \neq \perp \wedge val_7^-(D) \neq C[1]$ before C is enqueued, $val_7^-(D) = C[1]$ is never possible, so that $D \equiv C$ is not possible. Otherwise, if $val_7^-(D) = \perp$ before C is enqueued, then since there is no call to ForceVal$(\cdot, \cdot, \perp, 12)$ in this execution of RecursiveCompletion (Lemma 8 (ii)), the last assignment before $val_7^-(D) \neq \perp$ has to be a random one, after which $D \equiv C$ is not possible by Lemma 6 (note that $C, D \in \mathcal{CH}$ by assumption and Lemma 7). The arguments for all the other possible cases follow the same line. □

Last, a computation path will not be completed twice.

Lemma 10. *During a good execution $\overline{D}^{\Sigma_2}$, assume that at some point a chain $C = (y_i, k_1, k_2, i)$ is enqueued. If a chain D equivalent to C had been enqueued before this point, then $C \in CompSet$ when C is dequeued.*

Proof. By construction, no entry in any of the sets can be overwritten; hence the equivalence of C and D is kept till D is adapted, so that right after D is adapted, it still holds $D \equiv C$. Regardless of the value of i, C will be obtained by the two subsequent calls to EvalFWD, so that $C \in CompSet$ holds right after the execution Complete$(D, \cdot, \cdot)$, and also keeps holding till C is dequeued. □

$\widetilde{S}$ Does Not Abort due to Adaptations. The goal is to show that $val_l^+(C) \notin P_l^+$ and $val_l^-(C) \notin P_l^-$ right before any call to ForceVal$(val_l^+(C), val_l^-(C), \perp, l)$. Assume that *in a good execution $\overline{D}^{\Sigma_2}$, C is a partial chain which is enqueued at some time and to be adapted at P_l, and no chain equivalent to C was enqueued before C is enqueued* (this constitutes the *common assumption* for Lemma 11, 12, and 13). Consider the whole completion process of C, we have: first, before C is enqueued, the "endpoints" of C are sufficiently far from its adaptation zone P_l, so that the two values $val_l^+(C)$ and $val_l^-(C)$ are $\perp$ and are not in P_l.

Lemma 11. *Before the call to* InnerP *which led to C being enqueued, it holds $val_{l-1}^+(C) = \perp$ and $val_{l+1}^-(C) = \perp$.*[19]

Proof. Consider the case $C = (y_0, k_1, k_2, 0, 4)$ is enqueued by InnerP$(2, +, x_2)$: then before this call, $val_3^+(C) = \perp$ clearly holds, since $val_2^+(C) = x_2 \notin P_2^+$. On the other hand, assume that $val_5^-(C) \neq \perp$, then we have $val_i^-(C) \in P_i^-$ for $i = 6, 7, 8, 9, 10$. Denote by (x_i, y_i) these 5 entries. Then, consider the last random assignment before all these 5 entries are in $\{P\}$:

[19] Note that $val_l^+(C) = val_l^-(C) = \perp \notin P_l^\delta$ is an implication of this claim.

(i) it cannot have been in any of P_7, P_8, and P_9, otherwise $\mathsf{BadMidP}$ happens;
(ii) it cannot have been a forward assignment in P_6 nor a backward one in P_{10}, otherwise $\mathsf{BadInvP}$ happens (with respect to the chain $(y_7, y_8 \oplus x_9, y_7 \oplus x_8, 7)$).

By these, the last random assignment must have been a backward one in P_6 or a forward one in P_{10}, after which a chain equivalent to C would have been enqueued, a contradiction. Hence the lemma holds in this case.

Next, consider the case $C = (y_7, k_1, k_2, 7, 4)$ is enqueued by $\textsc{Inner}\text{P}(6, -, y_6)$: then before this call, $val_5^-(C) = \perp$ clearly holds. On the other hand, assume $val_3^+(C) \neq \perp$, then we have $((k_1, k_2), val_{16}^+(C)) \in ES^-$ and $val_i^+(C) \in P_i^+$ for $i = 1, 2, 14, 15$. Denote by (x_i, y_i) the four corresponding entries in $\{P\}$, and let $x_{16} = val_{16}^+(C)$ and $y_0 = ES^-((k_1, k_2), x_{16})$. Then, consider the last random assignment before this situation:

(i) it cannot have been in ES, otherwise either $\mathbf{E}.\text{E}(+, (k_1, k_2), y_0) \oplus k_2 \in P_{15}^-$ or $\mathbf{E}.\text{E}(-, (k_1, k_2), x_{16}) \oplus k_1 \in P_1^+$ and BadE happens;
(ii) it cannot have been a backward one in P_1 or P_2, nor a forward one in P_{14} or P_{15}, otherwise $\mathsf{BadInvP}$ happens (with respect to the chain $(y_0, k_1, k_2, 0)$);
(iii) it cannot have been a forward one in P_1 nor a backward one in P_{15}, otherwise BadP happens (with respect $(y_0, k_1, k_2, 0)$).

By these, the last random assignment must have been a forward one in P_2 or a backward one in P_{14}, after which a chain equivalent to C would have been enqueued, a contradiction. Hence the lemma also holds in this case.

The two cases when $l = 12$ are similar by symmetry. □

Then, during the period between the point C is enqueued and the point C is dequeued, if $val_l^+(C)$ or $val_l^-(C)$ is added to P_l, then there is a chain D which shares the same val_l^+ or val_l^- value with C.[20]

Lemma 12. *If $val_l^\delta(C) \in P_l^\delta$ $(\delta \in \{+, -\})$ when C is dequeued, then there is a chain D such that $val_l^\delta(C) = val_l^\delta(D) \neq \perp$ and $val_l^\delta(C)$ is added to P_l^δ during completion of D.*

Proof. To simplify notations, we assume $l = 4$, and focus on $val_4^+(C)$; but the argument is completely generic. From Lemma 11 we get that before C is enqueued, $val_3^+(C) = \perp$, which implies $val_4^+(C) = \perp$.

Consider the last assignment in the sets before $val_4^+(C) \neq \perp$ holds. We show that immediately after this assignment, it holds $val_4^+(C) \notin P_4^+$. By Lemma 11, this assignment happens earliest right before C is enqueued, at which point $C \in \mathcal{CH}$ (by Lemma 7), so that:

(i) it cannot have been a random backward one in $\{P\}$ as otherwise $\mathsf{BadInvP}$ occurs;
(ii) it cannot have been in ES as otherwise BadE occurs;

[20] The proofs of Lemma 12 and Lemma 13 are respectively similar to the first and second half of the proof of Lemma 16 in the full version of [LS13]. The flow is a bit too long for IDEM, hence we divide it, and hope this improves readability.

(iii) it cannot have been due to ForceVal, since by Lemma 8 (ii), the previous calls to ForceVal during the current execution of RecursiveCompletion only add entries to P_4, so that it is not possible that $val_4^+(C) = \bot$ due to a missed entry in P_{12} which is added by a call to ForceVal.

Therefore the last assignment has to be a random forward one in $\{P\}$, after which $val_4^+(C) \notin P_4^+$ as otherwise BadP happens.

Now, since $val_4^+(C) \notin P_4^+$ immediately after $val_4^+(C) \neq \bot$ holds, while $val_4^+(C) \in P_4^+$ when C is dequeued, the only possibility is that $val_4^+(C)$ is added to P_4^+ during completion of another chain D, for which at least $val_4^+(C) = val_4^+(D) \neq \bot$ holds. □

But it is not possible that C shares the same val_l^+/val_l^- value with some chain D. Consequently, during the period between the point C is enqueued and the point C is dequeued, the two values $val_l^+(C)$ and $val_l^-(C)$ cannot be added to P_l even if they are changed to non-empty.

Lemma 13. *When C is dequeued, it holds $val_l^+(C) \notin P_l^+$ and $val_l^-(C) \notin P_l^-$.*

Proof. To simplify notations, we assume $l = 4$, and show $x_4 \notin P_4^+$ when C is dequeued; but the argument is completely generic. Assume otherwise, i.e. $val_4^+(C) \in P_4^+$ when C is dequeued, then from Lemma 12, we get that there is a chain D such that: (i) $val_4^+(D) = val_4^+(C) \neq \bot$ when C is dequeued; (ii) D is enqueued before C is enqueued and is dequeued after C is enqueued; (iii) at any point, C and D are not equivalent. For the third point, note that by assumption (the *common assumption* of Lemma 11, 12, and 13), C and D are not equivalent when C is enqueued, and then by Lemma 9, from the point C is enqueued, at any point until C is dequeued, it is not possible that $C \equiv D$.

We then argue that $val_4^+(D) = val_4^+(C) \neq \bot$ is not possible for any such chain D. Consider three cases as follows:

Case I: at some point (in $\overline{D}^{\Sigma_2}$), it holds $val_2^+(D) \neq val_2^+(C)$. Consider the point right before the call to InnerP(i, δ, z_C) which led to C being enqueued. At this point, we have:

(i) C has been in $\mathcal{CH}$ (Lemma 7);
(ii) D also has been in $\mathcal{CH}$ since D is enqueued even earlier (note that this also holds in the case where D and C are enqueued by the same call to InnerP(i, δ, z_C));
(iii) $val_3^+(C) = \bot$ (by Lemma 11).

So that the last assignment (which happens earliest right before C is enqueued) before $val_3^+(C) \neq \bot \wedge val_3^+(D) \neq \bot$ has to be a random forward one in $\{P\}$, because:

(i) it cannot have been a backward one in $\{P\}$ as otherwise BadInvP occurs;
(ii) it cannot have been in ES as otherwise BadE occurs;
(iii) by Lemma 8 (ii), it cannot have been due to ForceVal (similar to the discussion in the proof of Lemma 12).

Therefore, right after $val_3^+(C) \neq \perp \wedge val_3^+(D) \neq \perp$ holds, it holds $val_3^+(C) \neq val_3^+(D)$, otherwise the last random forward assignment implies BadlyCollide (we've argued that C, D are never equivalent). Similarly, $val_4^+(C) \neq val_4^+(D)$ must hold in the future, otherwise one of the bad events happens.

The discussions in *Case I* exclude the possibility that $val_2^+(D) \neq val_2^+(C)$ while $val_3^+(D) = val_3^+(C) \neq \perp$. In this case, $val_4^+(D) = val_4^+(C) \neq \perp$ is possible; but this case itself is not possible.

Case II: at some point, it holds $val_2^+(D) = val_2^+(C) \neq \perp$ and $val_3^+(D) \neq val_3^+(C)$. Similarly to *Case I*, the last assignment (which happens after C and D are enqueued) before $val_4^+(C) \neq \perp \wedge val_4^+(D) \neq \perp$ holds has to be a random forward one in $\{P\}$, after which $val_4^+(C) = val_4^+(D) \neq \perp$ is not possible due to the absence of BadlyCollide.

Case III: at some point, it holds $val_2^+(D) = val_2^+(C) \neq \perp$ and $val_3^+(D) = val_3^+(C) \neq \perp$. In this case $val_4^+(D) = val_4^+(D) \neq \perp$ is clearly not possible as otherwise $D \equiv C$.

These terminate the proof. $\square$

The above establish that $\widetilde{S}$ does not abort due to adaptations.

Lemma 14. *In a good execution $\overline{D}^{\Sigma_2}$, before any call to* FORCEVAL$(x_l, y_l, \perp, l)$ *($l \in \{4, 12\}$), $x_l \notin P_l^+ \wedge y_l \notin P_l^-$ must hold.*

Proof. Assume that the call to FORCEVAL$(x_l, y_l, \perp, l)$ is made during the execution of COMPLETE(C, l). Then it necessarily be that $C \notin CompSet$ when C is dequeued, as otherwise the FORCEVAL call would not happen. Then from Lemma 13 we get that when C is dequeued, it holds $val_l^+(C) \notin P_l^+$ and $val_l^-(C) \notin P_l^-$. Consider $val_l^+(C)$: if $val_l^+(C) \neq \perp$ when C is dequeued, then $val_l^+(C) \notin P_l^+$ is clearly kept; if $val_l^+(C) = \perp$ when C is dequeued, then by construction, $val_l^+(C)$ can only be changed non-empty by a random forward assignment in P_{l-1} which happens during the execution of COMPLETE(C, l) (otherwise BadP happens), and $val_l^+(C) \notin P_l^+$ holds after this point otherwise BadP happens. Similarly, $val_l^-(C) \notin P_l^-$ right before the call to FORCEVAL. $\square$

$\widetilde{S}$ Does Not Abort. Since the abortion due to adaptations has been proved impossible, we finally obtain the non-abortion lemma itself.

Lemma 15. *$\widetilde{S}$ does not abort in a good execution $\overline{D}^{\Sigma_2}$.*

Proof. By construction, $\widetilde{S}$ only aborts during calls to FORCEVAL. But in a good execution, we have: first, $\widetilde{S}$ does not abort during the calls FORCEVAL$(z, z', +, i)$ and FORCEVAL$(z, z', -, i)$, otherwise BadHitAdapt happens; second, by Lemma 14, the calls FORCEVAL$(x, y, \perp, l)$ also do not lead to $\widetilde{S}$ aborts. $\square$

3.7 Final Transition from Σ_2 to Σ_3

This part is achieved by a tweaked version of the randomness mapping argument [CHK+14], and is actually very standard. As the beginning, with respect to $\overline{D}$, we introduce some notions first (borrowed from [ABD+13] and [CS15]).

First, recall the notion *good Σ_2-tuple*: $\alpha = (\mathbf{E}, \mathbf{P})$ is a good Σ_2-tuple if none of the bad events defined around Σ_2 executions happens during $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$ (page 24). Second, denote by $\mathcal{R}$ the set of all possible tuples of sets $\{P\}$ (of $\widetilde{S}$) standing at the end of Σ_2 executions when running with good Σ_2-tuples. For a good Σ_2-tuple $\alpha = (\mathbf{E}, \mathbf{P})$ and a tuple of sets $\{P\} \in \mathcal{R}$, if the sets of $\widetilde{S}$ standing at the end of $\overline{D}^{\Sigma_2(\alpha)}$ share exactly the same contents with $\{P\}$, then denote by $\overline{D}^{\Sigma_2(\alpha)} \to \{P\}$. Third, consider a set-tuple $\{P\} = \{P_1, \ldots, P_{15}\} \in \mathcal{R}$. For a tuple of random permutations $\mathbf{P}$, if for any $z \in P_i^{\delta}$, it holds $\mathbf{P}.\mathrm{P}(i, \delta, z) = P_i^{\delta}(z)$, then we said that $\mathbf{P}$ *coincides with* $\{P\}$, and denote $\mathbf{P} \cong \{P\}$.

Then, we have the following lemma, which states that the encryption and decryption results of IDEM$_{15}$ computed from $\{P\}$ are the same as those of $\widetilde{E}$.

Lemma 16. *Suppose that $\widetilde{E}.\mathrm{E}(\delta, (k_1, k_2), z)$ is queried during a good execution $\overline{D}^{\Sigma_2}$. Then, at the end of $\overline{D}^{\Sigma_2}$, when $\delta = +$, we have $\widetilde{E}.\mathrm{E}(+, (k_1, k_2), x) = val_{16}^{+}(x, k_1, k_2, 0)$; and $\widetilde{E}.\mathrm{E}(-, (k_1, k_2), y) = val_0^{-}(y \oplus k_2, k_1, k_2, y, 15)$ when $\delta = -$.*

Proof. *Wlog* consider $\delta = +$. If $\widetilde{E}.\mathrm{E}(+, (k_1, k_2), z)$ is made by $\widetilde{S}$, then $\widetilde{S}$ is completing a chain, so that the equality holds right after this completion – and will keep holding till the end since no entry will be overwritten. If $\widetilde{E}.\mathrm{E}(+, (k_1, k_2), z)$ is issued by $\overline{D}$, then $\overline{D}$ will emulate a call to EvalFwd$(x, k_1, k_2, 0, 15)$. Let $C = (x, k_1, k_2, 0)$, and let $x_i = val_i^{+}(C)$ and $y_i = val_i^{-}(C)$ for $i = 6, 7, 8, 9, 10$; by the above, at the end of $\overline{D}^{\Sigma_2}$, it holds $(x_i, y_i) \in P_i$. Consider the last call to InnerP before this situation: some impossibilities are excluded

- it cannot have been InnerP$(7, \cdot, \cdot)$, InnerP$(8, \cdot, \cdot)$, nor InnerP$(9, \cdot, \cdot)$, otherwise BadMidP happens;
- it cannot have been InnerP$(6, +, x_6)$ nor InnerP$(10, -, y_{10})$, otherwise BadInvP happens with respect to $(y_7, k_1, k_2, 7)$.

So that the only two possibilities are InnerP$(6, -, y_6)$ and InnerP$(10, +, x_{10})$, after which $(y_7, k_1, k_2, 7) \equiv C$ would have been enqueued and completed and the equality holds. □

The number of adaptations is the same as the number of queries to $\mathbf{E}$.

Lemma 17. *During a good Σ_2 execution $\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})}$, the number of adapted entries in $\{P\}$ (i.e. calls to* ForceVal$(x, y, \perp, l)$*) equals $|\widetilde{E}^{\mathbf{E}}.ES|$.*

Proof. By construction, each call ForceVal$(x, y, \perp, l)$ *corresponds to an execution* Complete *which further corresponds to a query to $\widetilde{E}$. On the other hand,* *each query to $\widetilde{E}$ made by $\widetilde{S}$ clearly corresponds to a call to* $\textsc{ForceVal}(x, y, \perp, l)$*; while each query to $\widetilde{E}$ made by $\overline{D}$ will lead to an execution of* $\textsc{EvalFwd}$ *(or* $\textsc{EvalBwd}$*) and an execution of* $\textsc{Complete}$ *– a call to* $\textsc{ForceVal}(x, y, \perp, l)$*.*

Above is the previous proof in the submission. It indeed oversimplified. We complement the points: (i) it's clear that two different $\textsc{ForceVal}$-calls/two different $\textsc{Complete}$-calls cannot correspond to the same entry in $\widetilde{E}^{\mathbf{E}}.ES$; (ii) two different entries in $\widetilde{E}^{\mathbf{E}}.ES$ cannot correspond to the same call to $\textsc{ForceVal}$ either. Because two different entries in $\widetilde{E}^{\mathbf{E}}.ES$ $((y_0, k_1, k_2, 0)$, $(y_0', k_1', k_2', 0))$ must correspond to two different $\textsc{Complete}$-calls. As $\textsc{Complete}$-calls are processed one-by-one, if two such calls lead to the same $\textsc{ForceVal}$-call, the execution would abort during the second one. □

The Σ_2 and Σ_3 executions that are "linked" by the sets of $\widetilde{S}$ behave the same in the view of $\overline{D}$.

Lemma 18. *Let $\alpha = (\mathbf{E}, \mathbf{P})$ be a good Σ_2-tuple, and denote by $\{P\}$ the sets of $\widetilde{S}$ standing at the end of $\overline{D}^{\Sigma_2(\alpha)}$. Then for any tuple $\mathbf{P'}$ such that $\mathbf{P'} \cong \{P\}$, the transcripts of queries and answers of $\overline{D}$ in $\overline{D}^{\Sigma_2(\alpha)}$ and $\overline{D}^{\Sigma_3(\mathbf{P'})}$ are the same; and $\overline{D}^{\Sigma_2(\alpha)} = \overline{D}^{\Sigma_3(\mathbf{P'})}$.*

Proof. By an induction, assume that the transcripts obtained by $\overline{D}$ are the same up to some point in the two executions, and consider the next query of $\overline{D}$. Since $\overline{D}$ is deterministic, the next query in the two executions are the same. We argue that the answers obtained in the two executions are the same. Depending on the type of this query, we distinguish two cases:

(i) the query is to P: then the answers are the same, since the answer obtained in $\overline{D}^{\Sigma_2(\alpha)}$ equals the value in $\{P\}$, and $\mathbf{P'}$ *coincides with* $\{P\}$;
(ii) the query is to E: then due to Lemma 16 and the fact that $\mathbf{P'} \cong \{P\}$, the answers obtained in $\overline{D}^{\Sigma_2(\alpha)}$ and $\overline{D}^{\Sigma_3(\mathbf{P'})}$ are the same.

Therefore, the two transcripts of $\overline{D}$ are the same. Since $\overline{D}$ is deterministic, the two outputs of $\overline{D}$ are also the same. □

For any $\{P\} \in \mathcal{R}$, the probabilities of the following two events are close:

(i) a Σ_2 execution with a random tuple $(\mathbf{E}, \mathbf{P})$ generates $\{P\}$;
(ii) a random tuple $\mathbf{P}$ coincides with $\{P\}$.

Lemma 19. *For any $\{P\} \in \mathcal{R}$, it holds*

$$\frac{Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]}{Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}]} \geq 1 - \frac{25q^4}{2^n}.$$

Proof. Let $\{P\} = \{P_1, \ldots, P_{15}\}$. Then $Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}] = \prod_{i=1}^{15} \prod_{j=0}^{|P_i|-1} \frac{1}{2^n - j}$. As to $Pr[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}]$, consider a good Σ_2-tuple $\alpha' = (\mathbf{E'}, \mathbf{P'})$ which satisfies

$\overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')} \to \{P\}$. It can be easily checked that $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}$ if and only if the transcripts of the combination $(\overline{D}, \widetilde{S})$ in $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})}$ and $\overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')}$ are the same[21] – also, the random values accessed during $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})}$ are exactly the same as those accessed during $\overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')}$. Assume that during $\overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')}$, there are u (v, resp.) entries in P_4 (P_{12}, resp.) that are defined by random assignments, and let $w = |\widetilde{E}^{\mathbf{E}'}.ES|$. Then it holds

$$Pr[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}] \leq (\prod_{i=1}^{3}\prod_{j=0}^{|P_i|-1}\frac{1}{2^n-j}) \cdot (\prod_{i=5}^{11}\prod_{j=0}^{|P_i|-1}\frac{1}{2^n-j}) \cdot (\prod_{i=13}^{15}\prod_{j=0}^{|P_i|-1}\frac{1}{2^n-j})$$
$$\cdot (\prod_{j=0}^{u-1}\frac{1}{2^n-j}) \cdot (\prod_{j=0}^{v-1}\frac{1}{2^n-j}) \cdot (\frac{1}{2^n-w})^w.$$

By Lemma 17, it holds $u+v+w = |P_4| + |P_{12}|$. Moreover it holds $w \leq 5q^2$ (Lemma 5), $|P_4| \geq u$, and $|P_{12}| \geq v$, hence

$$\frac{Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]}{Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}]} \geq \frac{(\prod_{j=0}^{|P_4|-1}\frac{1}{2^n-j}) \cdot (\prod_{j=0}^{|P_{12}|-1}\frac{1}{2^n-j})}{(\prod_{j=0}^{u-1}\frac{1}{2^n-j}) \cdot (\prod_{j=0}^{v-1}\frac{1}{2^n-j}) \cdot (\frac{1}{2^n-w})^w}$$
$$\geq \frac{(\frac{1}{2^n})^w}{(\frac{1}{2^n-w})^w} \geq 1 - \frac{w^2}{2^n} \geq 1 - \frac{25q^4}{2^n}.$$

as claimed. □

An implication of Lemma 18 is that the good Σ_2 executions can be partitioned with respect to the sets generated by them: for any $\{P\} \in \mathcal{R}$ and any two tuples $(\mathbf{E},\mathbf{P})$ and $(\mathbf{E}',\mathbf{P}')$, once $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}$ and $\overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')} \to \{P\}$, then $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} = \overline{D}^{\Sigma_2(\mathbf{E}',\mathbf{P}')}$. With this in mind, let Θ_1 be a subset of $\mathcal{R}$ such that for any tuple $(\mathbf{E},\mathbf{P})$ such that $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\} \in \Theta_1$ it holds $\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} = 1$. Then the following inequality holds. Its interpretation is that the Σ_3 executions in which $\overline{D}$ outputs 1 can be partitioned with respect to the member of Θ_1 without any "repeat count".

Lemma 20. $Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1] \geq \sum_{\{P\}\in\Theta_1} Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]$.

Proof. We show that for any tuple $\mathbf{P}^*$, there exists at most one $\{P\} \in \mathcal{R}$ s.t. $\mathbf{P}^* \cong \{P\}$. Assume otherwise, i.e. $\exists \{P\}' \in \mathcal{R}$ s.t. $\{P\} \neq \{P\}' \wedge \mathbf{P}^* \cong \{P\} \wedge \mathbf{P}^* \cong \{P\}'$. Assume that for two good tuples $\alpha = (\mathbf{E},\mathbf{P})$ and $\alpha' = (\mathbf{E}',\mathbf{P}')$, it holds $\overline{D}^{\Sigma_2(\alpha)} \to \{P\}$ and $\overline{D}^{\Sigma_2(\alpha')} \to \{P\}'$. Then, consider any query of the combination $(\overline{D}, \widetilde{S})$ in the two executions $\overline{D}^{\Sigma_2(\alpha)}$ and $\overline{D}^{\Sigma_2(\alpha')}$: (i) any query to $\mathbf{P}/\mathbf{P}'$ obtains the same answer, since $\mathbf{P}.\mathrm{P}(i,\delta,z) = P_i^{\delta}(z) = \mathbf{P}^*.\mathrm{P}(i,\delta,z) = P_i'^{\delta}(z) = \mathbf{P}'.\mathrm{P}(i,\delta,z)$; (ii) by Lemma 16, any query to $\mathbf{E}/\mathbf{E}'$ also obtains the same

[21] This can be shown by an induction similar to that of Lemma 18.

answer. By an induction similar to Lemma 18, we have that the transcripts of the combination $(\overline{D}, \widetilde{S})$ in the two executions $\overline{D}^{\Sigma_2(\alpha)}$ and $\overline{D}^{\Sigma_2(\alpha')}$ are the same, so that the two set-tuples $\{P\}$ and $\{P\}'$ should be the same, a contradiction. After this, we have

$$\begin{aligned} Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1] &\geq Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1 \wedge \exists \{P\} \in \mathcal{R} \text{ s.t. } \mathbf{P} \cong \{P\}] \\ &= \sum_{\{P\} \in \Theta_1} Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}] \text{ (by Lemma 18)} \end{aligned}$$

as claimed. □

Then the following lemma completes the transition from Σ_2 to Σ_3, and terminates the indifferentiability proof for IDEM$_{15}$.

Lemma 21. $|Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathrm{IDEM}_{15}^{\mathbf{P}}, \mathbf{P})} = 1] - Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\widetilde{E}^{\mathbf{E}}, \widetilde{S}^{\widetilde{E}^{\mathbf{E}}, \mathbf{P}})} = 1]| \leq \frac{2^{14} \cdot q^6}{2^n}$.

Proof. Wlog assume that $Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} = 1] \geq Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1]$, then

$$\begin{aligned} &|Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1] - Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} = 1]| \\ \leq &Pr_{\mathbf{E},\mathbf{P}}[(\mathbf{E}, \mathbf{P}) \text{ is not good}] \\ &+ Pr_{\mathbf{E},\mathbf{P}}[(\mathbf{E}, \mathbf{P}) \text{ is good} \wedge \overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} = 1] - Pr_{\mathbf{P}}[\overline{D}^{\Sigma_3(\mathbf{P})} = 1] \\ \leq &\frac{2^{13.4} \cdot q^6}{2^n} + \sum_{\{P\} \in \Theta_1} (Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}] - Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]) \text{ (by Lemma 20, 15)} \\ \leq &\frac{2^{13.4} \cdot q^6}{2^n} + \sum_{\{P\} \in \Theta_1} \frac{25q^4}{2^n} \cdot Pr_{\mathbf{E},\mathbf{P}}[\overline{D}^{\Sigma_2(\mathbf{E},\mathbf{P})} \to \{P\}] \text{ (by Lemma 19)} \leq \frac{2^{14} \cdot q^6}{2^n} \end{aligned}$$

as claimed. □

4 Sequential Distinguisher for IDEM$_6$

Consider IDEM$_6^{\mathbf{P}}$ from six independent permutations $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_6)$. The distinguisher $D^{\mathrm{IDEM}_6^{\mathbf{P}}, \mathbf{P}}$ fixes k_2 to an arbitrarily value to reduce IDEM$_6$ to SEM$_3$, and then runs LS's distinguisher on SEM$_3$ [LS13], as follows:

(1) choose an arbitrary value $k_2 \in \{0,1\}^n$;
(2) choose arbitrary values $x_5, k_1, k_1' \in \{0,1\}^n$, where $k_1 \neq k_1'$;
(3) compute $y_4 := x_5 \oplus k_1$ and $y_4' := x_5 \oplus k_1'$;
(4) compute $x_3 := \mathbf{P}_3^{-1}(k_2 \oplus \mathbf{P}_4^{-1}(y_4))$ and $x_3' := \mathbf{P}_3^{-1}(k_2 \oplus \mathbf{P}_4^{-1}(y_4'))$, by issuing queries to $\mathbf{P}$;
(5) compute $y_2 := x_3 \oplus k_1$ and $y_2' := x_3' \oplus k_1'$;
(6) compute $k_1'' := y_2 \oplus x_3'$ and $k_1''' := y_2' \oplus x_3$, and output 0 if k_1, k_1', k_1'', and k_1''' are not pairwise distinct;

(7) compute $x_1 := \mathbf{P}_1^{-1}(k_2 \oplus \mathbf{P}_2^{-1}(y_2))$ and $x'_1 := \mathbf{P}_1^{-1}(k_2 \oplus \mathbf{P}_2^{-1}(y'_2))$, by issuing queries to $\mathbf{P}$;
(8) compute $x := x_1 \oplus k_1$, $x' := x'_1 \oplus k'_1$, $x'' := x_1 \oplus k''_1$, and $x''' := x'_1 \oplus k'''_1$;
(9) query $y := E((k_1, k_2), x)$, $y' := E((k'_1, k_2), x')$, $y'' := E((k''_1, k_2), x'')$, and $y''' := E((k'''_1, k_2), x''')$;
(10) output 1 if $y \oplus y' \oplus y'' \oplus y''' = 0$, while output 0 otherwise.

By this, we have the following theorem.

Theorem 3. $\mathrm{IDEM}_6^{\mathbf{P}}$ *is not seq-indifferentiable from* $\mathbf{E}$.

Proof. With respect to the key k_2 chosen by $D^{\mathrm{IDEM}_6^{\mathbf{P}},\mathbf{P}}$, IDEM_6 is equivalent to the following 3-round single-key Even-Mansour cipher SEM_3^*:

$$\mathrm{SEM}_3^*(k_1, y_0) = k_1 \oplus \mathbf{P}_3^*(k_1 \oplus \mathbf{P}_2^*(k_1 \oplus \mathbf{P}_1^*(k_1 \oplus y_0))),$$

where $\mathbf{P}_1^*(z) = \mathbf{P}_2(k_2 \oplus \mathbf{P}_1(z))$, $\mathbf{P}_2^*(z) = \mathbf{P}_4(k_2 \oplus \mathbf{P}_3(z))$, and $\mathbf{P}_3^*(z) = \mathbf{P}_6(k_2 \oplus \mathbf{P}_5(z))$. Moreover, step (2) to step (10) of $D^{\mathrm{IDEM}_6^{\mathbf{P}},\mathbf{P}}$ are equivalent to the sequential distinguisher against SEM_3 introduced in [LS13]. Then by Theorem 1 in [LS13], $D^{\mathrm{IDEM}_6^{\mathbf{P}},\mathbf{P}}$ has advantage negligibly close to 1. □

As mentioned, the hope of attacking even one more round has been ruled out by the sequential indifferentiability proof for IDEM_7.

5 Seq-Indifferentiability for 7-round IDEM

It is natural to ask whether the additional n-bit key offers more freedom to the adversary and enable to attack more than this trivial 2×3 rounds. The second main result – also the main theorem of this section – provides a negative answer, and is as follows:

Theorem 4. *The 7-round Even-Mansour cipher* IDEM_7 *from seven independent random permutations* $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_7)$ *and two n-bit keys* (k_1, k_2) *alternatively xored is strongly and statistically* $(q, \sigma, t, \varepsilon)$*-seq-indifferentiable from* $\mathbf{E}$*, where* $\sigma = q^3$, $t = O(q^3)$, *and* $\varepsilon \leq \frac{27q^6}{2^n} = O(\frac{q^6}{2^n})$.

The proof is much simpler than that of Theorem 2, since there is no recursive chain completion. In the following, we first present the simulator, then give the proof.

5.1 Simulator for IDEM_7

To make a distinction from the notations used in Sect. 3, we denote by $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ the simulator for IDEM_7 with access to $\mathbf{E}$ and $\mathbf{P}$. Similarly to $\mathbf{S}^{\mathbf{E},\mathbf{P}}$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ also offers an interface $\mathrm{P}(i, \delta, z)$ where $(i, \delta, z) \in \{1, \ldots, 7\} \times \{+, -\} \times \{0,1\}^n$ and maintains a set P_i for each i to keep the already defined pairs of IO. The other notations P_i^+, P_i^-, and $|P_i|$ are all similar to those introduced in the context of

IDEM$_{15}$. $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ uses an additional set ES to maintain the history of its queries to $\mathbf{E}$, which is similar to the set of $\widetilde{E}^{\mathbf{E}}$ introduced in Sect. 3. We also use the notations ES^{+}, ES^{-}, and $|ES|$ similar to Sect. 3.

Upon a query to $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(i,\delta,z)$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ calls an inner procedure $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}$, and $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}$ answers with $P_i^{\delta}(z)$ if $x \in P_i^{\delta}$, or queries $\mathbf{P}.\mathrm{P}(i,\delta,z)$ to obtain the answer z' and adds z and z' to P_i if $z' \notin P_i^{\overline{\delta}}$ while aborts otherwise.

The chain completing mechanism of $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ is much simpler than that of $\mathbf{S}^{\mathbf{E},\mathbf{P}}$, and is somehow close to that appeared in [CS15]: $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ completes the potential partial chains upon receiving a new query $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(i,\delta,x)$ with $i \in \{3,4,5\}$. More clearly, when the query is of the form $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(3,+,x)$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(4,-,y)$, or $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(5,+,x)$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ considers all newly created tuples $(x_3,x_4,x_5) \in P_3^{+} \times P_4^{+} \times P_5^{+}$, and computes $k_1 := P_4^{+}(x_4) \oplus x_5$, $k_2 := P_3^{+}(x_3) \oplus x_4$. $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ then evaluates in IDEM$_7$ both backward and forward until obtaining the corresponding y_7 and x_7, that is, computing the following values by calling $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}$ and querying $\mathbf{E}$, in the order: (1) $y_2 := x_3 \oplus k_1$; (2) $y_1 := \mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}(2,-,y_2) \oplus k_2$; (3) $y_0 := \mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}(1,-,y_1) \oplus k_1$; (4) $y_7 := \mathbf{E}.\mathrm{E}(+,(k_1,k_2),y_0) \oplus k_2$; (5) $x_6 := P_5^{+}(x_5) \oplus k_2$; (6) $x_7 := \mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}^{in}(6,+,x_6) \oplus k_1$. $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ finally aborts if $x_7 \in P_7^{+}$ or $y_7 \in P_7^{-}$, otherwise adds (x_7,y_7) to P_7 as a newly defined pair of IO.

When the query is $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(3,-,y)$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(4,+,x)$, or $\mathcal{S}^{\mathbf{E},\mathbf{P}}.\mathrm{P}(5,-,y)$, $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ considers all newly created tuples $(x_3,x_4,x_5) \in P_3^{+} \times P_4^{+} \times P_5^{+}$, computes k_1 and k_2, evaluates in IDEM$_7$ both forward and backward until obtaining the corresponding x_1 and y_1, and finally adds (x_1,y_1) to P_1 or aborts if $x_1 \in P_1^{+}$ or $y_1 \in P_1^{-}$. The strategy is illustrated in Fig. 1 (right).

To simplify the reasoning, we introduce a modified simulator $\mathcal{T}^{\mathbf{E},\mathbf{P}}$, which is obtained by embedding two early abort conditions into $\mathcal{S}^{\mathbf{E},\mathbf{P}}$:

(i) when a chain C is to be adapted at P_1 (P_7, resp.), right after the assignment (line 13 or 16 in the code below) inside the call to P^{in} which led to C being detected, if the value y_2 (x_6, resp.) corresponding to C has been in P_2^{-} (P_6^{+}, resp.), then $\mathcal{T}$ aborts. This is captured by the procedure CheckFreeBuffer;

(ii) right after an assignment in P_3, P_4, or P_5 (line 13/16), $\mathcal{T}$ aborts if the assignment creates a "lock" in the middle three rounds: for $(i,j) \in \{(3,4),(4,5)\}$, if $\exists (x_i,y_i),(x_i',y_i') \in P_i$ and $(x_j,y_j),(x_j',y_j') \in P_j$ such that $x_i \oplus y_j = x_i' \oplus y_j'$ and $y_i \oplus x_j = y_i' \oplus x_j'$. This is captured by the procedure CheckLock. This situation is harmful for the procedure CompChain in some cases.

With all the above in mind, we have the pseudocode of $\mathcal{S}$ and $\mathcal{T}$ as follows. Note that the *underlined lines* only exist in $\mathcal{T}$ (say, $\mathcal{S}$ does not early abort).

```
1: Simulator $\mathcal{S}^{\mathbf{E},\mathbf{P}}$:                    $\boxed{\textbf{Simulator } \mathcal{T}^{\mathbf{E},\mathbf{P}}:}$
2: Variables: Sets $\{P\} = \{P_1,\ldots,P_7\}$ and $ES$, initially empty
3: public procedure $\mathrm{P}(i,\delta,z)$
4:   return $\mathrm{P}^{in}(i,\delta,z)$
```

```
5:  private procedure $\mathrm{P}^{in}(i, \delta, z)$
6:    if $z \notin P_i^{\delta}$ then
7:      $z' := \mathbf{P}.\mathrm{P}(i, \delta, z)$
8:      if $z' \in P_i^{\overline{\delta}}$ then // when $i = 1, 7$
9:        abort
10:     $\underline{\textsc{CheckFreeBuffer}(i, \delta, z')}$
11:     if $\delta = +$ then
12:       $\underline{\textsc{CheckLock}(i, z, z')}$
13:       $\underline{P_i := P_i \cup \{(z, z')\}}$
14:     else // $\delta = -$
15:       $\underline{\textsc{CheckLock}(i, z', z)}$
16:       $\underline{P_i := P_i \cup \{(z', z)\}}$
17:     if $i = 3 \wedge \delta = +$ then
18:       forall $(x_4, x_5) \in P_4^+ \times P_5^+$ do
19:         $\textsc{CompChain}(z, x_4, x_5, 3, 7)$
20:     else if $i = 4 \wedge \delta = +$ then
21:       forall $(x_3, x_5) \in P_3^+ \times P_5^+$ do
22:         $\textsc{CompChain}(x_3, z, x_5, 4, 1)$
23:     else if $i = 5 \wedge \delta = +$ then
24:       forall $(x_3, x_4) \in P_3^+ \times P_4^+$ do
25:         $\textsc{CompChain}(x_3, x_4, z, 5, 7)$
26:     else if $i = 3 \wedge \delta = -$ then
27:       forall $(x_4, x_5) \in P_4^+ \times P_5^+$ do
28:         $\textsc{CompChain}(z', x_4, x_5, 3, 1)$
29:     else if $i = 4 \wedge \delta = -$ then
30:       forall $(x_3, x_5) \in P_3^+ \times P_5^+$ do
31:         $\textsc{CompChain}(x_3, z', x_5, 4, 7)$
32:     else if $i = 5 \wedge \delta = -$ then
33:       forall $(x_3, x_4) \in P_3^+ \times P_4^+$ do
34:         $\textsc{CompChain}(x_3, x_4, z', 5, 1)$
35:   return $P_i^{\delta}(z)$
36: private procedure $\textsc{CompChain}(x_3, x_4, x_5, i, l)$
37:   $k_1 := P_4^+(x_4) \oplus x_5$
38:   $k_2 := P_3^+(x_3) \oplus x_4$
39:   if $l = 1$ then
40:     $x_6 := P_5^+(x_5) \oplus k_2$
41:     $x_7 := \mathrm{P}^{in}(6, +, x_6) \oplus k_1$
42:     $x_8 := \mathrm{P}^{in}(7, +, x_7) \oplus k_2$
43:     $y_0 := \mathbf{E}.\mathrm{E}(-, (k_1, k_2), x_8)$
44:     $ES := ES \cup \{(y_0, x_8, (k_1, k_2))\}$
45:     $x_1 := y_0 \oplus k_1$
46:     $y_2 := x_3 \oplus k_1$
47:     $y_1 := \mathrm{P}^{in}(2, -, y_2) \oplus k_2$
48:     if $x_1 \in P_1^+ \vee y_1 \in P_1^-$ then
49:       abort
50:     $P_1 := P_1 \cup \{(x_1, y_1)\}$
51:   else // $l = 7$
52:     $y_2 := x_3 \oplus k_1$
53:     $y_1 := \mathrm{P}^{in}(2, -, y_2) \oplus k_2$
54:     $y_0 := \mathrm{P}^{in}(1, -, y_1) \oplus k_1$
55:     $x_8 := \mathbf{E}.\mathrm{E}(+, (k_1, k_2), y_0)$
56:     $ES := ES \cup \{(y_0, x_8, (k_1, k_2))\}$
57:     $y_7 := x_8 \oplus k_2$
58:     $x_6 := P_5^+(x_5) \oplus k_2$
59:     $x_7 := \mathrm{P}^{in}(6, +, x_6) \oplus k_1$
60:     if $x_7 \in P_7^+ \vee y_7 \in P_7^-$ then
61:       abort
62:     $P_7 := P_7 \cup \{(x_7, y_7)\}$
63: private procedure $\textsc{CheckFreeBuffer}(i, \delta, z')$
64:   if $(i, \delta) = (3, +) \wedge \exists (x_4, y_5) \in P_4^+ \times P_5^-$ s.t. $z' \oplus x_4 \oplus y_5 \in P_6^+$ then
65:     abort
66:   else if $(i, \delta) = (4, +) \wedge \exists (x_3, x_5) \in P_3^+ \times P_5^+$ s.t. $x_3 \oplus z' \oplus x_5 \in P_2^-$ then
67:     abort
68:   else if $(i, \delta) = (5, +) \wedge \exists (y_3, x_4) \in P_3^- \times P_4^+$ s.t. $y_3 \oplus x_4 \oplus z' \in P_6^+$ then
69:     abort
70:   else if $(i, \delta) = (3, -) \wedge \exists (y_4, x_5) \in P_4^- \times P_5^+$ s.t. $z' \oplus y_4 \oplus x_5 \in P_2^-$ then
71:     abort
72:   else if $(i, \delta) = (4, -) \wedge \exists (y_3, y_5) \in P_3^- \times P_5^-$ s.t. $y_3 \oplus z' \oplus y_5 \in P_6^+$ then
73:     abort
74:   else if $(i, \delta) = (5, -) \wedge \exists (x_3, y_4) \in P_3^+ \times P_4^-$ s.t. $x_3 \oplus y_4 \oplus z' \in P_2^-$ then
75:     abort
76: private procedure $\textsc{CheckLock}(i, x, y)$
77:   if $i = 3 \wedge \exists ((x_3, y_3), (x_4, y_4), (x_4', y_4')) \in P_3 \times P_4 \times P_4$
78:       s.t. $x \oplus y_4' = x_3 \oplus y_4 \wedge y \oplus x_4' = y_3 \oplus x_4$ then abort
79:   if $i = 5 \wedge \exists ((x_5, y_5), (x_4, y_4), (x_4', y_4')) \in P_5 \times P_4 \times P_4$
80:       s.t. $x_4 \oplus y_5 = x_4' \oplus y \wedge y_4 \oplus x_5 = y_4' \oplus x$ then abort
81:   if $i = 4 \wedge \exists ((x_3, y_3), (x_3', y_3'), (x_4, y_4)) \in P_3 \times P_3 \times P_4$
82:       s.t. $x_3 \oplus y_4 = x_3' \oplus y \wedge y_3 \oplus x_4 = y_3' \oplus x$ then abort
```

```
83: if $i = 4 \wedge \exists((x_5, y_5), (x'_5, y'_5), (x_4, y_4)) \in P_5 \times P_5 \times P_4$
84:     s.t. $x_4 \oplus y_5 = x \oplus y'_5 \wedge y_4 \oplus x_5 = y \oplus x'_5$ then abort
```

5.2 Intermediate System Σ'_2.

Denote by $\Sigma'_1(\mathbf{E}, \mathcal{S}^{\mathbf{E},\mathbf{P}})$ the simulated system, and by $\Sigma'_3(\mathrm{IDEM}_7^{\mathbf{P}}, \mathbf{P})$ the real system. As a quite standard first step, we introduce an intermediate system $\Sigma'_2(\mathrm{IDEM}_7^{\mathcal{T}^{\mathbf{E},\mathbf{P}}}, \mathcal{T}^{\mathbf{E},\mathbf{P}})$, in which the cipher IDEM_7 calls the interfaces of $\mathcal{T}$ to compute (as done in [MPS12,CS15]). The three systems involved in this proof are depicted in Fig. 3.

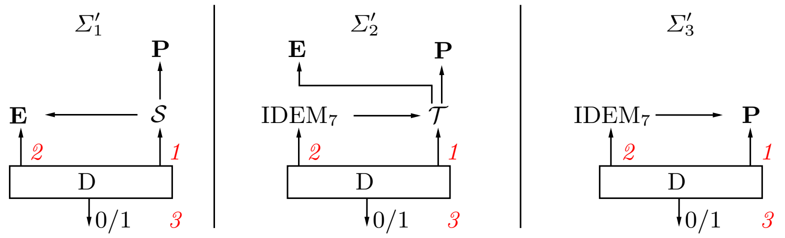


Fig. 3. Systems used in the seq-indifferentiability proof for IDEM_7. The number in red and *italic* illustrates the order of the queries/actions (of the sequential distinguisher).

5.3 Complexity of $\mathcal{S}$ and $\mathcal{T}$

The termination argument of $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ and $\mathcal{T}^{\mathbf{E},\mathbf{P}}$ is captured by the following lemma.

Lemma 22. *For any tuple of primitives* $(\mathbf{E}, \mathbf{P})$ *and any seq-distinguisher* D *of total oracle query cost at most* q*, the following hold:*

(i) At the end of the Σ'_2 *execution* $D^{\Sigma'_2(\mathbf{E},\mathbf{P})}$*, with respect to the sets of* $\mathcal{T}^{\mathbf{E},\mathbf{P}}$*, for* $i \in \{3, 4, 5\}$*, it holds* $|P_i| \le q$*; for* $i \in \{1, 2, 6, 7\}$*,* $|P_i| \le 2q^3$*; and* $|ES| \le q^3$*;*
(ii) during the Σ'_1 *execution* $D^{\Sigma'_1(\mathbf{E},\mathbf{P})}$*,* $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ *issues at most* q^3 *queries to* $\mathbf{E}$*, and runs in time at most* $O(q^3)$*.*

Proof. Since the total oracle query cost of D is at most q, $\mathcal{T}$ receives at most q queries of the form $\mathrm{P}(3, \delta, z)$, so that $|P_3| \le q$. The same for $|P_4|$ and $|P_5|$. By this, the procedure CompChain is called at most $|P_3| \cdot |P_4| \cdot |P_5| \le q^3$ times.

Next, consider $|P_1|$: $|P_1|$ can by enlarged by at most 1 when $\mathcal{T}$ receives a query of the form $\mathrm{P}(1, \delta, z)$ (at most q), or when $\mathcal{T}$ calls CompChain, so that it is at most $q + q^3 \le 2q^3$. The same for $|P_2|$, $|P_6|$, and $|P_7|$. Then $|ES| \le q^3$ follows from the fact that $\mathcal{T}$ queries $\mathbf{E}$ only during an execution of CompChain.

It can be easily seen that the argument for proposition (i) as well as the bounds also hold for $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ (in $D^{\Sigma'_1}$), so that $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ makes at most q^3 queries to $\mathbf{E}$. Moreover, the time complexity of $\mathcal{S}^{\mathbf{E},\mathbf{P}}$ is clearly dominated by the executions of CompChain, so that is at most $O(q^3)$. □

5.4 Indistinguishability of Outputs

As done in Sect. 3, we first upper bound the abort probability of $\mathcal{T}$, then give the formal transition between the systems.

The Abort Probability of $\mathcal{T}$ is Negligible. For ease of description, we adapt the two helper functions val_l^+ and val_l^- used in the IDEM$_{15}$ context, and denoted $sval_l^+$ and $sval_l^-$. But they only accept the input values of *the middle three rounds* as inputs (so that they can be much simpler than val_l^+ and val_l^-).

```
function $sval_l^+(x_3, x_4, x_5)$
  if $x_3 \notin P_3^+ \vee x_4 \notin P_4^+ \vee x_5 \notin P_5^+$
    then return false
  if $l \in \{3, 4, 5\}$ then return $x_l$
  $k_2 := P_3^+(x_3) \oplus x_4$
  $k_1 := P_4^+(x_4) \oplus x_5$
  $j = 6$
  $z = P_5^+(x_5) \oplus k_2$
  while $j \neq l$ do
    if $j \leq 7$ then
      if $z \notin P_j^+$ then return $\perp$
      if $j$ is odd then
        $z = P_j^+(z) \oplus k_2$
      else // $j$ is even
        $z = P_j^+(z) \oplus k_1$
      $j := j + 1$
    else // $j = 8$
      if $((k_1, k_2), z) \notin ES^-$ then return $\perp$
        $z = ES^-((k_1, k_2), z) \oplus k_1$
        $j := 1$
  return $z$
```

```
function $sval_l^-(x_3, x_4, x_5)$
  if $x_3 \notin P_3^+ \vee x_4 \notin P_4^+ \vee x_5 \notin P_5^+$
    then return false
  if $l \in \{3, 4, 5\}$ then return $P_l^+(x_l)$
  $k_2 := P_3^+(x_3) \oplus x_4$
  $k_1 := P_4^+(x_4) \oplus x_5$
  $j = 2$
  $z = x_3 \oplus k_1$
  while $j \neq l$ do
    if $j \geq 1$ then
      if $z \notin P_j^-$ then return $\perp$
      if $j$ is odd then
        $z = P_j^-(z) \oplus k_1$
      else // $j$ is even
        $z = P_j^-(z) \oplus k_2$
      $j := j - 1$
    else // $j = 0$
      if $((k_1, k_2), z) \notin ES^+$ then return $\perp$
        $z = ES^+((k_1, k_2), z) \oplus k_2$
        $j := 7$
  return $z$
```

Note that in the arguments throughout this section, each time we call the two functions, they are called on parameters with the correct form, and won't return the tag **false**.

Then we can give the arguments. First, each value obtained from $\mathbf{E}$ during the chain completion is a fresh random value – for this, note that in $D^{\Sigma_2'}$, $\mathbf{E}$ can only be queried by $\mathcal{T}$, so that the set $\mathcal{T}.ES$ is actually the query history of $\mathbf{E}$.

Lemma 23. *During any execution $D^{\Sigma_2'(IDEM_7, \mathcal{T}^{\mathbf{E},\mathbf{P}})}$, each time $\mathcal{T}^{\mathbf{E},\mathbf{P}}$ issues a query $\mathrm{E}(\delta, (k_1, k_2), z)$ to $\mathbf{E}$, it holds $((k_1, k_2), z) \notin ES^\delta$ right before the query.*

Proof. $\mathcal{T}$ queries $\mathbf{E}$ only when it completes a partial chain (x_3, x_4, x_5). Since the contents of the sets are never overwritten, if two such queries are the same, then it can be easily deduced that the two chains corresponding to the two queries are the same, which is not possible by construction. □

Then we are able to bound the overall abort probability.

Lemma 24. *For any seq-distinguisher D of total oracle query cost at most q ($q^3 < \frac{2^n}{4}$), during the execution $D^{\Sigma_2'(IDEM_7, \mathcal{T}^{\mathbf{E},\mathbf{P}})}$, $Pr[\mathcal{T}^{\mathbf{E},\mathbf{P}}$ aborts$] \leq \frac{26q^6}{2^n}$.*

Proof. Since CompChain is called at most q^3 times, by the lemmas (25, 26, 27, and 28) below, we have

$$Pr[\mathcal{T}^{\mathbf{E},\mathbf{P}} \text{ aborts}] \leq \frac{4q^6}{2^n - 2q^3} + \frac{2q^6}{2^n - q} + \frac{2q^4}{2^n - q} + q^3 \cdot (\frac{5q^3}{2^n - 2q^3}) \leq \frac{26q^6}{2^n}$$

as claimed. □

For any seq-distinguisher D of total oracle query cost at most q, the following lemmas analyze the abort probability due to each possibility.

Lemma 25. *The overall probability that $\mathcal{T}$ aborts at line 9 is at most $\frac{4q^6}{2^n-2q^3}$.*

Proof. Since $\mathbf{P}$ are 7 permutations, if the abort condition at line 8 holds, then it is necessarily due to the random value $\mathbf{P}.\mathrm{P}(i,\delta,z)$ colliding with a value in $P_i^{\overline{\delta}}$ added by previous adaptations (line 50, 62). As a consequence, only calls to $\mathrm{P}^{in}(1,\delta,z)$ and $\mathrm{P}^{in}(7,\delta,z)$ are able to lead to abortion. For $\mathrm{P}^{in}(1,\delta,z)$, since CompChain is executed at most q^3 times (Lemma 22), the number of values added by line 50 is at most q^3, so that the probability is at most $|P_1| \cdot \frac{q^3}{2^n-|P_1|} \leq \frac{2q^6}{2^n-2q^3}$. Similarly for $\mathrm{P}^{in}(7,\delta,z)$, so that the bound in total is $\frac{4q^6}{2^n-2q^3}$. □

Lemma 26. *The overall probability that $\mathcal{T}$ aborts during* CheckFreeBuffer *is at most $\frac{2q^6}{2^n-q}$.*

Proof. Consider a chain $((x_3,y_3),(x_4,y_4),(x_5,y_5))$ detected by $\mathcal{T}$. Regardless of the concrete type of the last assignment before this chain is detected, the probability that $\mathcal{T}$ aborts during CheckFreeBuffer due to this chain is at most $\frac{\mathrm{Max}\{|P_2|,|P_6|\}}{2^n-\mathrm{Max}\{|P_3|,|P_4|,|P_5|\}} \leq \frac{2q^3}{2^n-q}$ – for concreteness, consider an example: if the last assignment defines $P_3^+(x_3)$ to $\mathbf{P}.\mathrm{P}(3,+,x_3)$, then the condition at line 64 holds with probability $Pr[\mathbf{P}.\mathrm{P}(3,+,x_3) \oplus x_4 \oplus y_5 \in P_6^+] \leq \frac{|P_6|}{2^n-|P_3|}$. By Lemma 22, at most q^3 chains are detected, so that the overall probability is at most $\frac{2q^6}{2^n-q}$. □

Lemma 27. *The overall probability that $\mathcal{T}$ aborts during* CheckLock *is at most $\frac{2q^4}{2^n-q}$.*

Proof. The analysis is a bit similar to Lemma 26: at the end of $D^{\Sigma_2'}$, consider a 4-tuple $((x_3,y_3),(x_3',y_3'),(x_4,y_4),(x_4',y_4')) \in P_3 \times P_3 \times P_4 \times P_4$. Regardless of the concrete type of the last assignment before this tuple is in $\{P\}$, the probability that $\mathcal{T}$ aborts during CheckLock due to this tuple is at most $\frac{1}{2^n-q}$ – for example, if the last assignment defines $P_3^+(x_3)$ to $\mathbf{P}.\mathrm{P}(3,+,x_3)$, then the condition at line 77 holds with probability $Pr[\mathbf{P}.\mathrm{P}(3,+,x_3) = y_3 \oplus x_4 \oplus x_4'] \leq \frac{1}{2^n-q}$. By Lemma 22, there are at most q^4 such tuples. Ditto for 4-tuples $((x_4,y_4),(x_4',y_4'),(x_5,y_5),(x_5',y_5')) \in P_4 \times P_4 \times P_5 \times P_5$, so that the overall probability is at most $\frac{2q^4}{2^n-q}$. □

Lemma 28. *During an execution* $\textsc{CompChain}(x_3, x_4, x_5, i, l)$*, the probability that* $\mathcal{T}$ *aborts at line 49 (in case* $l = 1$*) or line 61 (in case* $l = 7$*) is at most* $\frac{5q^3}{2^n-2q^3}$.

Proof. Let $C = (x_3, x_4, x_5)$, denote by k_1 and k_2 the two associated keys, and *wlog* consider $\textsc{CompChain}(C, i, 1)$. Consider the probability that $sval_1^+(C)$ is in P_1^+ right before the adaptation of C: by Lemma 23, this probability equals $Pr[\mathbf{E}.\mathrm{E}(-, (k_1, k_2), sval_8^+(C)) \oplus k_1 \in P_1^+]$, which is at most $\frac{|P_1|}{2^n-|ES|} \leq \frac{2q^3}{2^n-q^3}$.

Then, consider the probability that $sval_1^-(C)$ is in P_1^- right before the adaptation of C. Briefly speaking, right after $sval_1^-(C) \neq \perp$ holds, it holds $Pr[sval_1^-(C) \in P_1^-] \leq \frac{|P_1|}{2^n-|P_2|} \leq \frac{2q^3}{2^n-2q^3}$; if $sval_1^-(C) \notin P_1^-$ at this point, then during the period between this point and the adaptation of C, the probability that $sval_1^-(C)$ is added to P_1^- is at most $\frac{q^3}{2^n-2q^3}$. Hence the overall probability that $sval_1^-(C)$ is in P_1^- right before the adaptation of C is at most $\frac{2q^3}{2^n-2q^3} + \frac{q^3}{2^n-2q^3} \leq \frac{3q^3}{2^n-2q^3}$. The detailed discussions for this probability are as follows, and are divided into two cases depending on the value of i.

Case 1: $i \in \{3, 4\}$. Since $\mathcal{T}$ did not abort during $\textsc{CheckFreeBuffer}$ right before this execution $\textsc{CompChain}(C, i, 1)$, we have: right after the random assignment (line 13 or 16) inside the call to P^{in} which led to C being detected, it holds $sval_2^-(C) \notin P_2^-$. By construction, from this point till the adaptation of C, all the assignments in P_2 are random backward ones (line 16); hence right after $P_2^-(sval_2^-(C))$ is defined (it may be defined before the execution $\textsc{CompChain}(C, i, 1)$), it holds $Pr[sval_1^-(C) \in P_1^-] = Pr[\mathbf{P}.\mathrm{P}(2, -, sval_2^-(C)) \oplus k_2 \in P_1^-] \leq \frac{|P_1|}{2^n-|P_2|} \leq \frac{2q^3}{2^n-2q^3}$.

If $sval_1^-(C) \notin P_1^-$ holds right after $P_2^-(sval_2^-(C))$ is defined, then consider any execution $\textsc{CompChain}(x'_3, x'_4, x'_5, i, 1)$ during the period between this point and the adaptation of C, and let $D = (x'_3, x'_4, x'_5)$: if $sval_1^-(C) = sval_1^-(D) \neq \perp$ before the adaptation of D, then $sval_1^-(C)$ will be added to P_1^-. But the probability is negligible:

- if $sval_2^-(C) = sval_2^-(D)$, then $sval_1^-(C) \neq sval_1^-(D)$ otherwise $C = D$ (when $i = 3$, then by construction, D has to be of the form (x_3, x'_4, x'_5), and $sval_j^-(C) = sval_j^-(D)$ for $j = 1, 2, 3$ and clearly $C = D$; when $i = 4$, then it can be deduced that $sval_j^-(C) = sval_j^-(D)$ for $j = 1, 2, 3, 4$);
- if $sval_2^-(C) \neq sval_2^-(D)$, then among $P_2^-(sval_2^-(C))$ and $P_2^-(sval_2^-(D))$, the probability that the one defined later leads to $sval_1^-(C) = sval_1^-(D)$ is at most $\frac{1}{2^n-|P_2|}$ (since $\mathcal{T}$ did not abort during $\textsc{CheckFreeBuffer}$, right after the random assignment inside the call to P^{in} which led to C and D being detected, both $sval_2^-(C) \notin P_2^-$ and $sval_2^-(D) \notin P_2^-$, and both $P_2^-(sval_2^-(C))$ and $P_2^-(sval_2^-(D))$ must be defined after this point. *Wlog* assume $P_2^-(sval_2^-(C))$ is defined later, then right after $sval_2^-(C) \in P_2^-$ holds, $Pr[sval_1^-(C) = sval_1^-(D)] = Pr[\mathbf{P}.\mathrm{P}(2, -, sval_2^-(C)) \oplus k_2 = sval_1^-(D)] \leq$

$\frac{1}{2^n-|P_2|}$). Since the number of such chain D is at most q^3, the overall probability is $\frac{q^3}{2^n-2q^3}$.

Case 2: $i = 5$. Similarly to *Case 1*, right after the random assignment in P_5 (line 16), it holds $sval_2^-(C) \notin P_2^-$ since $\mathcal{T}$ did not abort; and right after $P_2^-(sval_2^-(C))$ is defined, it holds $Pr[sval_1^-(C) \in P_1^-] \le \frac{2q^3}{2^n-2q^3}$.

If $sval_1^-(C) \notin P_1^-$ holds right after $P_2^-(sval_2^-(C))$ is defined, then during the period between this point and the adaptation of C, the overall probability that $sval_1^-(C)$ is added to P_1^- is at most $\frac{q^3}{2^n-2q^3}$. For this, consider any execution $\textsc{CompChain}(x_3', x_4', x_5, 5, 1)$ during this period and let $D = (x_3', x_4', x_5)$. First, if $sval_2^-(C) = sval_2^-(D)$, then $sval_1^-(C) \ne sval_1^-(D)$ otherwise $y_3 \oplus x_4 = y_3' \oplus x_4' \wedge x_3 \oplus y_4 = x_3' \oplus y_4'$, which would have led $\mathcal{T}$ to abortion during a previous execution of $\textsc{CheckLock}$. Second, if $sval_2^-(C) \ne sval_2^-(D)$, then the case is similar to *Case 1*: among $P_2^-(sval_2^-(C))$ and $P_2^-(sval_2^-(D))$, the probability that the one defined later leads to $sval_1^-(C) = sval_1^-(D)$ is at most $\frac{1}{2^n-|P_2|}$, and the overall probability is $\frac{q^3}{2^n-2q^3}$.

By the above, the probability that $\mathcal{T}$ aborts during $\textsc{CompChain}(x_3, x_4, x_5, i, 1)$ is at most $\frac{2q^3}{2^n-2q^3} + \frac{3q^3}{2^n-2q^3} \le \frac{5q^3}{2^n-2q^3}$. The discussions for $\textsc{CompChain}(x_3, x_4, x_5, i, 7)$ are similar by symmetry. These terminate the proof. $\square$

Non-Abortion Implies Indistinguishability of Answers. This subsection proves the indistinguishability of Σ_1' and Σ_3' based on the non-abortion argument for $\mathcal{T}$, and is similar to Section 3.7. A tuple of primitives $\alpha = (\mathbf{E}, \mathbf{P})$ is a *good Σ_2'-tuple* if $\mathcal{T}^\alpha$ does not abort during the Σ_2' execution $D^{\Sigma_2'(\alpha)}$. Denote by $\mathcal{R}'$ the set of all possible set-tuples $\{P\} = \{P_1, \ldots, P_7\}$ that can be generated by $\mathcal{T}$ when running with good Σ_2'-tuples. Also, if the sets of $\mathcal{T}^\alpha$ standing at the end of $D^{\Sigma_2'(\alpha)}$ are exactly the same with $\{P\} \in \mathcal{R}'$, then denote by $D^{\Sigma_2'(\alpha)} \to \{P\}$; if the tuple $\mathbf{P}$ agrees with all the values defined by $\{P\}$, say $\mathbf{P}$ *coincides with* $\{P\}$ and denote $\mathbf{P} \cong \{P\}$. With these notations, we have the following arguments. First, in a good Σ_2' execution $D^{\Sigma_2'(\mathbf{E},\mathbf{P})}$, the answers given by IDEM_7 are the same as those given by $\mathbf{E}$.

Lemma 29. *For any good Σ_2'-tuple $\alpha = (\mathbf{E}, \mathbf{P})$, D obtains the same answer for any query to $\mathbf{E}$.E/$\text{IDEM}_7^{\mathcal{T}^\alpha}$.E in the two executions $D^{\Sigma_1'(\alpha)}$ and $D^{\Sigma_2'(\alpha)}$.*

Proof. In $D^{\Sigma_2'(\alpha)}$, each time D issues a query $(+, (k_1, k_2), y_0)$ to IDEM_7, IDEM_7 will queries $\mathcal{T}^\alpha.\text{P}(3, +, x_3)$, $\mathcal{T}^\alpha.\text{P}(4, +, x_4)$, and $\mathcal{T}^\alpha.\text{P}(5, +, x_5)$ for the corresponding values, so that after IDEM_7 answers this query, it holds $(x_3, y_3) \in P_3$, $(x_4, y_4) \in P_4$, and $(x_5, y_5) \in P_5$. By this, during $D^{\Sigma_2'(\alpha)}$, there was necessarily a call to $\mathcal{T}^\alpha.\textsc{CompChain}(x_3, x_4, x_5, \cdot, \cdot)$, after which the answer of IDEM_7 (computed from $\mathcal{T}$'s sets $\{P\}$) will be the same as $\mathbf{E}$. The queries $(-, (k_1, k_2), x_8)$ are similar by symmetry. These establish the claim, since the answers in $D^{\Sigma_1'(\alpha)}$ are directly given by $\mathbf{E}$. $\square$

The Σ_1', Σ_2', and Σ_3' executions that are "linked" by the sets of $\mathcal{T}$ behave the same in the view of D.

Lemma 30. *Let $\alpha = (\mathbf{E}^\alpha, \mathbf{P}^\alpha)$ be a good Σ_2'-tuple, and let $D^{\Sigma_2'(\alpha)} \to \{P\}$. Then for any tuple $\mathbf{P}'$ such that $\mathbf{P}' \cong \{P\}$, the transcripts of queries and answers of D in $D^{\Sigma_1'(\alpha)}$, $D^{\Sigma_2'(\alpha)}$, and $D^{\Sigma_3'(\mathbf{P}')}$ are the same; and $D^{\Sigma_1'(\alpha)} = D^{\Sigma_2'(\alpha)} = D^{\Sigma_3'(\mathbf{P}')}$.*

Proof. By an induction similar to Lemma 18, consider each query of D:

(i) the query is to P: then in $D^{\Sigma_1'(\alpha)}$ and $D^{\Sigma_2'(\alpha)}$, the query must be made during the first phase (in which D only queries P). It can be easily seen that in this phase, if $\mathcal{T}^\alpha$ does not abort in $D^{\Sigma_2'(\alpha)}$, then $\mathcal{S}^\alpha$ does not abort in $D^{\Sigma_1'(\alpha)}$; and the operation sequence of $\mathcal{T}^\alpha$ (excluding the calls to CheckFreeBuffer and CheckLock) is the same as that of $\mathcal{S}^\alpha$. Hence D obtains the same answer for the P-query in $D^{\Sigma_1'(\alpha)}$ and $D^{\Sigma_2'(\alpha)}$. On the other hand, the answers obtained in $D^{\Sigma_2'(\alpha)}$ and $D^{\Sigma_3'(\mathbf{P}')}$ are also the same since $\mathbf{P}' \cong \{P\}$;

(ii) the query is to E: then by Lemma 29, the answers obtained in $D^{\Sigma_1'(\alpha)}$ and $D^{\Sigma_2'(\alpha)}$ are the same. Also, the answers obtained in $D^{\Sigma_2'(\alpha)}$ and $D^{\Sigma_3'(\mathbf{P}')}$ are the same, since the permutation values used by IDEM$_7$ to compute the answers are the same.

Therefore, the three transcripts of D are the same. Since D is deterministic, the three outputs of D are also the same. □

For any $\{P\} \in \mathcal{R}'$, the probabilities of the following two events are close:

(i) a Σ_2' execution with a random tuple $(\mathbf{E}, \mathbf{P})$ generates $\{P\}$;

(ii) a random permutation tuple $\mathbf{P}$ *coincides with* $\{P\}$.

Lemma 31. *With respect to a fixed distinguisher D of total oracle query cost at most q, for any $\{P\} \in \mathcal{R}'$, it holds*

$$\frac{Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]}{Pr_{\mathbf{E},\mathbf{P}}[D^{\Sigma_2'(\mathbf{E},\mathbf{P})} \to \{P\}]} \geq 1 - \frac{q^6}{2^n}.$$

Proof. Following the same line as Lemma 19, the deviation of the ratio (from 1) is due to the distance between the distribution of $\mathbf{E}$'s answers and the distribution of $\mathbf{P}$'s answers. The number of queries to $\mathbf{E}$ is at most $|ES| \leq q^3$, so that

$$\frac{Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]}{Pr_{\mathbf{E},\mathbf{P}}[D^{\Sigma_2'(\mathbf{E},\mathbf{P})} \to \{P\}]} \geq 1 - \frac{|ES|^2}{2^n} \geq 1 - \frac{q^6}{2^n}.$$

as claimed. □

Let Θ_1' be a subset of $\mathcal{R}'$ such that for any tuple $(\mathbf{E}, \mathbf{P})$ such that $D^{\Sigma_2'(\mathbf{E},\mathbf{P})} \to \{P\} \in \Theta_1'$ it holds $D^{\Sigma_2'(\mathbf{E},\mathbf{P})} = 1$. Then the analogue of Lemma 17 and Lemma 20 also hold in this context. These yield the final transition lemma.

Lemma 32. *For any seq-distinguisher D of total oracle query cost at most q, it holds*

$$\left| Pr_{\mathbf{P}}[D^{\Sigma_3'(\mathrm{IDEM}_7^{\mathbf{P}},\mathbf{P})} = 1] - Pr_{\mathbf{E},\mathbf{P}}[D^{\Sigma_1'(\mathbf{E},\mathcal{S}^{\mathbf{E},\mathbf{P}})} = 1] \right| \leq \frac{27q^6}{2^n}.$$

Proof. Let $\alpha = (\mathbf{E}, \mathbf{P})$, and *wlog* assume $Pr_\alpha[D^{\Sigma_1'(\alpha)} = 1] \geq Pr_{\mathbf{P}}[D^{\Sigma_3'(\mathbf{P})} = 1]$, then

$$\begin{aligned}
&\left| Pr_{\mathbf{P}}[D^{\Sigma_3'(\mathbf{P})} = 1] - Pr_\alpha[D^{\Sigma_1'(\alpha)} = 1] \right| \\
\leq & \frac{26q^6}{2^n} + Pr_\alpha[\alpha \text{ is a good } \Sigma_2'\text{-tuple} \wedge D^{\Sigma_1'(\alpha)} = 1] - Pr_{\mathbf{P}}[D^{\Sigma_3'(\mathbf{P})} = 1] \\
& (\text{since by Lemma 24, } Pr[\mathcal{T}^\alpha \text{ aborts during } D^{\Sigma_2'(\alpha)}] \leq \frac{26q^6}{2^n}) \\
\leq & \frac{26q^6}{2^n} + Pr_\alpha[\alpha \text{ is a good } \Sigma_2'\text{-tuple} \wedge D^{\Sigma_2'(\alpha)} = 1] - Pr_{\mathbf{P}}[D^{\Sigma_3'(\mathbf{P})} = 1] \\
& (\text{since by Lemma 30, if } \alpha \text{ is good then } D^{\Sigma_1'(\alpha)} = D^{\Sigma_2'(\alpha)}) \\
\leq & \frac{26q^6}{2^n} + \sum_{\{P\} \in \Theta_1'} (Pr_\alpha[D^{\Sigma_2'(\alpha)} \to \{P\}] - Pr_{\mathbf{P}}[\mathbf{P} \cong \{P\}]) \\
\leq & \frac{26q^6}{2^n} + \sum_{\{P\} \in \Theta_1'} \frac{q^6}{2^n} \cdot Pr_\alpha[D^{\Sigma_2'(\alpha)} \to \{P\}] \text{ (by Lemma 31)} \leq \frac{27q^6}{2^n}
\end{aligned}$$

as claimed. □

6 Conclusion

As a first step towards understanding the security of iterated Even-Mansour with key-length larger than the block-size, this work analyzes (seq-)indifferentiability of Even-Mansour with two independent round-keys alternatively xored, and proves that 7 rounds is necessary and sufficient to achieve sequential indifferentiability while 15 rounds is sufficient to achieve full indifferentiability.

Acknowledgements

We deeply thank the anonymous referees of FSE 2015 and ASIACRYPT 2015, for their useful comments and corrections. We thank Meicheng Liu and Jianghua Zhong for their suggestions. We also thank Yu Chen, Jian Guo, and Wentao Zhang, for their encouragements.

This work is partially supported by National Key Basic Research Project of China (2011CB302400), National Science Foundation of China (61379139) and the "Strategic Priority Research Program" of the Chinese Academy of Sciences, Grant No. XDA06010701.

A Independent Keys and Its Consequence: Generalizing to Cases With More General Key Schedules.

As mentioned in Introduction, in IDEM, the two n-bit keys are *independent*. In this section, we briefly discuss the consequences.

Generalizing-and-Keeping-Independence. First, it is not hard to see that the analysis on IDEM can be generalized to IEM with key schedules with the following two properties:

(i) the $2n$ key bits can be divided into two non-intersect n-bit halves k_1 and k_2;
(ii) when i is odd, the round-key involved in the i-th round-key xor is derived by an efficiently invertible n-bit permutation γ_i from k_1; when i is even, the round-key involved in the i-th round-key xor is derived by an efficiently invertible n-bit permutation γ_i from k_2.

But such key schedules are quite contrived. Besides directly interleaving k_1 and k_2 with out any additional transformation (the key schedule of IDEM, and an instance is LED-128), we did not find any real instance with such generalized key schedules.

Further Generalization. We now consider the key schedules used in real $\mathsf{BC}[2n, n]$ designs. A variety of them – including AES-256, Serpent, – use FSR-based key schedules. A possible approach is to model such key schedules as an array of efficiently invertible $2n$-bit permutations $(\gamma_0, \gamma_1, \ldots, \gamma_{r-1})$ such that:

(i) each γ_i gives two consecutive round-keys, i.e. it holds $(k_i, k_{i+1}) = \gamma_i(K)$ and k_i and k_{i+1} are the two round-keys used before and after the permutation P_i;
(ii) any two consecutive permutations γ_i and γ_{i+1} agree on the $(i+1)^{\text{th}}$ round-key, i.e. for $(k_i, k_{i+1}) = \gamma_i(K)$ and $(k'_{i+1}, k_{i+2}) = \gamma_{i+1}(K)$ it holds $k_{i+1} = k'_{i+1}$.

At current time, we are not sure about whether the analysis on IDEM can be easily generalized to IEM with such general key schedules. The reason is that many arguments in this work rely on the independence between the two n-bit keys (for instance, the proof of Lemma 2 in the supporting doc; the details in the detection zone of the simulator for IDEM_7; the proof of Lemma 26). Moreover, there is another subtle problem when we are trying to generalize the full indifferentiability proof to such IEM with 15 rounds: when detecting "external" chains, it seems like that the modified simulator has no choice but to exhaustively search for the correct $2n$-bit master keys (after it recovers the two round-keys k_1 and k_{14}), i.e. tests all possible $2n$-bit master keys K to find the one for which $\gamma_0(K)$ gives k_1 and $\gamma_{14}(K)$ gives k_{14}. (note that the simulator $\mathbf{S}$ in subsection 3.1 takes advantage of the fact that in IDEM, k_1 and k_{14} are derived from two independent halves of the master key). This brings in a soar in the time complexity of the simulator, i.e. from $O(q^8)$ (in this work) to $O(2^{2n})$. These discussions show that how to concretely model the key schedule(s) *might* affect the security of Even-Mansour with key-length twice the block-size.

Although there exist such difficulties, the following conclusion is not questionable, that is, *the indifferentiability proof(s) for IDEM is already capable of showing that it is possible to build $\mathsf{BC}[2n, n]$ from key-alternating ciphers without using very complex key schedules – or even with "no" key schedule*. The interest on designing blockciphers (key-alternating ciphers) with no key schedule was (probably) woken up by LED [GPPR11].

B Possibility of Further Reducing Rounds (For the Indifferentiability Proof)

B.1 Reduce the Number of Rounds in Chain Detection Zone.

In the current framework, arranging less rounds in the chain detection zones is not possible. First, arranging as few as three rounds in the middle detection zone is not possible:

– if we do not arrange detection condition on P_7 (as depicted in the blue arrows in Fig. 4), then the "pureness" property (analogue of Lemma 8) still exists, but D can easily fill the middle zone without waking S and break S as a consequence. A possible query sequence is depicted in red lines and words in Fig. 4.

– if we arrange detection conditions on P_7, then it further includes two possibilities. Before we see the details, we remark that in both of two cases, increasing the number of rounds in the detection zones at two sides does not resolve problems. Hence we have the conclusion: the previous conjectured proof strategy of LS (Fig. 7) is "not easy to be carried out" albeit not totally ruled out.

  • We use the detection conditions similar to the proof for IDEM$_7$ (please see Fig. 5), such that for each detected middle chain, the endpoint in the buffer round has not been defined. Then the values $val_l^\delta(C)$ does not necessarily remain constant during the adaptations of other chains (note that in this case, the "pureness" property does not exist). This can be shown by adapting the operation sequence in the appendix of the full version of [LS13]. Consider a distinguisher $\mathbf{D}$ with the following operation sequence:
  (1) $\mathbf{D}$ chooses four arbitrary values $x_4, k_1, k_2, k_2' \in \{0,1\}^n$ ($k_2 \neq k_2'$);
  (2) $\mathbf{D}$ takes x_4, k_1, k_2 as a partial chain and evaluates forward: $\mathbf{D}$ computes $x_5 := \mathbf{P}_4(x_4) \oplus k_1$, $x_6 := \mathbf{P}_5(x_5) \oplus k_2$, $x_7 := \mathbf{P}_6(x_6) \oplus k_1$, $x_8 := \mathbf{P}_7(x_7) \oplus k_2$, and $x_9 := \mathbf{P}_8(x_8) \oplus k_1$; (at this point the simulator starts completing $(y_7, k_1, k_2, 7, 10)$)
  (3) $\mathbf{D}$ takes x_4, k_1, k_2' as a partial chain and evaluates backward: $\mathbf{D}$ computes $x_1 := \mathbf{P}_4^{-1}(x_4 \oplus k_2')$, …, $x_{14} := \mathbf{E}.\mathrm{E}(-, (k_1, k_2'), x_1 \oplus k_1)$, …, and $x_{12} := \mathbf{P}_{12}^{-1}(x_{13} \oplus k_1)$; (at this point the simulator starts completing $(y_0', k_1, k_2', 0, 10)$)

It is not hard to check that during the completion of $(y'_0, k_1, k'_2, 0)$, the simulator evaluates forward, computes $x'_6 := \mathbf{P}_5(x_5) \oplus k'_2$, $y'_6 := \mathbf{P}_6(x'_6)$, $x'_7 := y'_6 \oplus k_1$, $y'_7 := \mathbf{P}_7(x'_7)$, $x'_8 := y'_7 \oplus k'_2$, $y'_8 := \mathbf{P}_8(x'_8)$, and enqueues two tuples $C = (y'_7, k_1, k_2, 7, 4)$ and $D = (y_7, k_1, k'_2, 7, 10)$ which will satisfy $val_5^-(C) = y_5 \oplus k_2 \oplus k'_2 = val_5^-(D)$ and $val_4^-(C) = val_5^-(C) \oplus k_1 = val_4^-(D)$. Then $val_3^-(D)$ will clearly change from $\perp$ to non-empty during the adaptation of C. This is disastrous, since the endpoints of D may not be random – at least, at current time, a proof in this case is out of range.

- We use a modified strategy such that the "pureness" property is kept (Fig. 6). This seemingly delicate simulator is trivially insecure. Consider a distinguisher $\mathbf{D}$:
  (1) $\mathbf{D}$ chooses four arbitrary values $x_7, x_8, x'_8, y_9 \in \{0,1\}^n$, and computes $y'_9 := x_8 \oplus x_8 \oplus y_9$;
  (2) $\mathbf{D}$ queries $y_8 := \mathbf{P}_8(x_8)$, $y'_8 := \mathbf{P}_8(x'_8)$, $x_9 := \mathbf{P}_9^{-1}(y_9)$, and $x'_9 := \mathbf{P}_9^{-1}(y'_9)$;
  (3) $\mathbf{D}$ computes $y_6 := x_7 \oplus y_8 \oplus x_9$ and $y'_6 := x_7 \oplus y'_8 \oplus x'_9$ and queries $x_6 := \mathbf{P}_6^{-1}(y_6)$ and $x'_6 := \mathbf{P}_6^{-1}(y'_6)$;
  (4) $\mathbf{D}$ queries $y_7 := \mathbf{P}_7(x_7)$.

  It is not hard to check that the simulator finally enqueues two chains corresponding to (x_6, x_7, x_8, x_9) and (x'_6, x_7, x'_8, x'_9) and will adapt them at P_{10}, whereas they collide on P_{10} since $y_9 \oplus x_8 \oplus y_7 = y'_9 \oplus x'_8 \oplus y_7$.

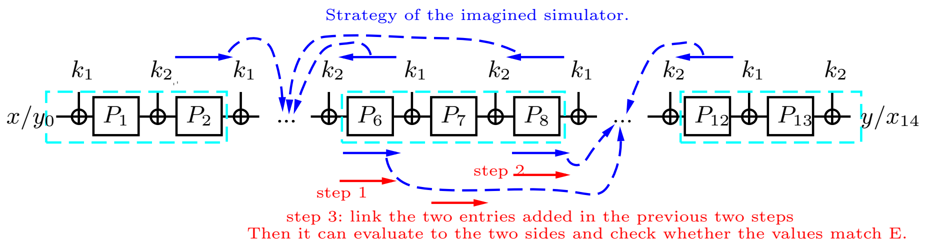

Strategy of the imagined simulator.
k_1
k_2
x/y_0
P_1
P_2
P_6
P_7
P_8
P_{12}
P_{13}
y/x_{14}
step 1
step 2
step 3: link the two entries added in the previous two steps
Then it can evaluate to the two sides and check whether the values match E.

Fig. 4. 13 rounds with no detection condition on the middle round seems to be a bad solution. The entire 13 rounds is too long to be placed, hence we only present the core modified part; similarly for the figures below.

Second, arranging four rounds in the middle detection zone is not possible. The most problematic point is: in this case, the number of rounds in total is 14, an even number; as a consequence, the detection zone at the two sides does not work (Fig. 8). If we increase the number of rounds in the side-detection-zone, then it turns to IDEM$_{15}$ again. Discussions above ruin out the hopes on compressing detection zones.

B.2 Remove the Buffer Rounds.

As mentioned in Introduction, the additional n-bit key allows the adversary to be more free to choose values. As a consequence, once removing even a single pair

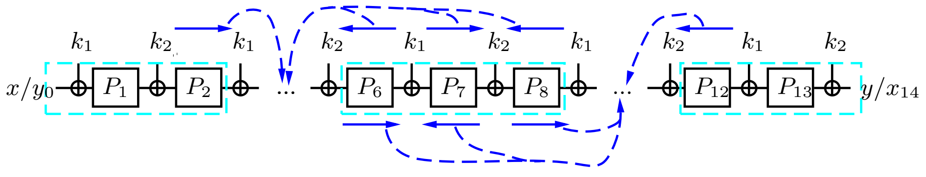


Fig. 5. 13 rounds with detection conditions on the middle round similar to the proof for IDEM$_7$ seems to be hard to handle.

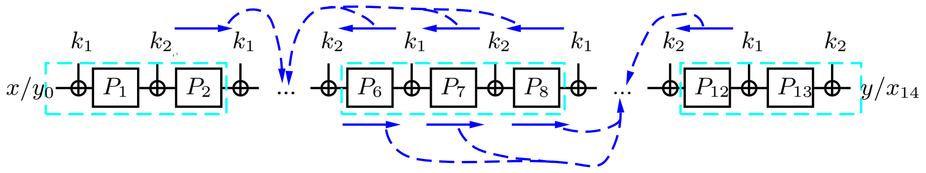


Fig. 6. 13 rounds: keeps the pureness property but still fails.

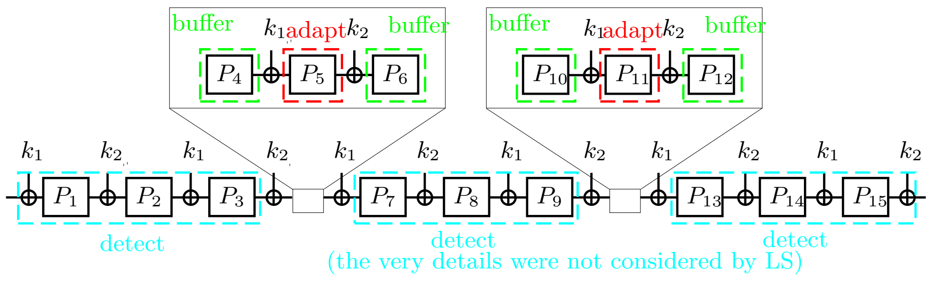


Fig. 7. The conjectured strategy of LS (for 15-round case).

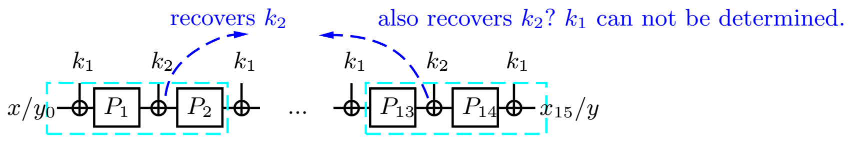


Fig. 8. 14 rounds: the side-detection-zone does not work.

of "buffer" rounds (in the current framework), there exist quite trivial attacks.[22] We first see what if we remove the two buffer rounds surrounding the "first" adaptation round. Then there is a distinguisher $\mathbf{D}$:

(1) $\mathbf{D}$ chooses four arbitrary values $x_2, k_1, k_2, k_2' \in \{0,1\}^n$;
(2) $\mathbf{D}$ takes x_2, k_1, k_2 as a partial chain and evaluates backward: $\mathbf{D}$ computes $y_1 := x_2 \oplus k_2$, $y_0 := \mathbf{P}_1^{-1}(y_1) \oplus k_1$, $x_{14} := \mathbf{E}.\mathrm{E}(+, (k_1, k_2), y_0)$, $x_{13} := \mathbf{P}_{13}^{-1}(x_{14} \oplus k_2)$, and $x_{12} := \mathbf{P}_{12}^{-1}(x_{13} \oplus k_1)$;
(3) $\mathbf{D}$ takes x_2, k_1, k_2' as a partial chain and evaluates backward: $\mathbf{D}$ computes $y_1' := x_2 \oplus k_2'$, $y_0' := \mathbf{P}_1^{-1}(y_1') \oplus k_1$, $x_{14}' := \mathbf{E}.\mathrm{E}(+, (k_1, k_2'), y_0')$, $x_{13}' := \mathbf{P}_{13}^{-1}(x_{14}' \oplus k_2')$, and $x_{12}' := \mathbf{P}_{12}^{-1}(x_{13}' \oplus k_1)$;
(4) $\mathbf{D}$ queries $\mathbf{P}_2(x_2)$.

It is easy to see that $\mathbf{S}$ will adapt two chains $C = (y_0, k_1, k_2, 0)$ and $C' = (y_0, k_1, k_2', 0)$ at P_3 and $val_3^+(C) = val_3^+(C')$ and $\mathbf{S}$ will abort.

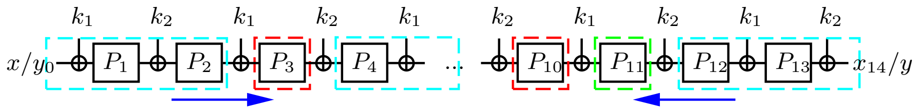


Fig. 9. Removing the buffer for the first adaptation zone is not possible.

Similarly by symmetry, we cannot remove the two buffer rounds surrounding the "second" adaptation round. A somehow surprising implication of these discussions is the failure of another plausible strategy for building simulator for the seq-indifferentiability proof of IDEM$_7$ (please see Fig. 10). This strategy is closer to the successful strategy for IDEM$_{15}$. But there is a (seq-)distinguisher D:

(1) D chooses four arbitrary values $y_2, k_2, k_1, k_1' \in \{0,1\}^n$;
(2) for $(y_2, k_1, k_2, 2)$, D evaluates forward: D computes $y_3 := \mathbf{P}_3(y_2 \oplus k_1)$, $y_4 := \mathbf{P}_3(y_3 \oplus k_2)$, $y_5 := \mathbf{P}_4(y_4 \oplus k_1)$, and $y_6 := \mathbf{P}_5(y_5 \oplus k_2)$;
(3) for $(y_2, k_1', k_2, 2)$, D evaluates forward: D computes $y_3' := \mathbf{P}_3(y_2 \oplus k_1')$, $y_4' := \mathbf{P}_3(y_3' \oplus k_2)$, $y_5' := \mathbf{P}_4(y_4' \oplus k_1')$, and $y_6' := \mathbf{P}_5(y_5' \oplus k_2)$;
(4) D queries $\mathbf{P}_2^{-1}(y_2)$.

$\mathcal{S}$ will complete two chains characterized by $C = (y_2, k_1, k_2, 2)$ and $C' = (y_2, k_1', k_2, 2)$ and $val_1^-(C) = val_1^-(C') = y_2 \oplus k_2$ and $\mathcal{S}$ will abort.

Always an Odd Number of Rounds? By the discussions above, it seems like that we always focus on IDEM with an odd number of rounds, which deviates from the fact that the number of rounds in AES-256 is 14. For this, we stress that the model IDEM studied in this paper is mainly a theoretical model, and the number of rounds studied in the proofs should not be directly related to those in practical

[22] Note that to avoid the chain detection problem in 14-round case (already discussed), we cannot remove a single buffer round.

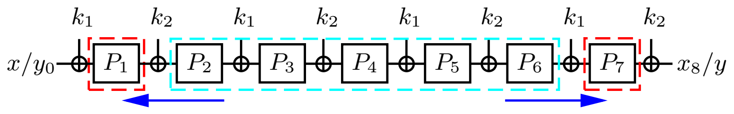


Fig. 10. A failed simulation strategy for IDEM$_7$.

designs – they are mainly theoretical results. Somewhat similar deviations occur in the other contexts, for instance, for SEM, 1 round already suffices to resist known-key attacks [ABM14], while known-key attacks could penetrate at least 8 rounds on AES-128 [Gil14].

C Possibility of Further Reducing the Complexity of S

It is natural to wonder that in the indifferentiability proof for IDEM$_{15}$, whether the complexity of **S** can be further reduced. Actually, we highly suspect that this further optimization is possible, and already made two attempts, but both of them result in (partial) failures.

C.1 The First Failed Attempt: Leading to Another Tradeoff

The first attempt seems to be able to bound the number of middle chains to $O(q)$; but it works with a high expense of probability. Consider an undirected bipartite graph BM based on the sets P_7, P_8, and P_9, which conveniently encodes the information of chains and their endpoints in these sets. The two shores of the graph BM is $\{0,1\}^n$. Edges of BM are constructed as follows: for every 3-tuple $((x_7, y_7), (x_8, y_8), (x_9, y_9)) \in P_7 \times P_8 \times P_9$, we construct an edge $(x_7 \oplus y_8 \oplus x_9, y_7 \oplus x_8 \oplus y_9)$. This constitutes all edges of BM.

Now, if conditioned on the absence of some bad events, all the connected components of BM are acyclic (i.e. they are trees), then the sub-graph composed by the $4q$ endpoints in P_6 and P_{10} has at most $2q - 1 < 2q$ edges, i.e. the number of middle chains is at most $2q$.

At the first glance, it is not hard to avoid cycles; however, this is not the case. Consider the simplest example, which is a 4-cycle (say, a complete bipartite graph with 4 vertices). Assume that the four edges in this graph are $(x_7^i, y_7^i, x_8^i, y_8^i, x_9^i, y_9^i)$ $(i = 1, 2, 3, 4)$. Then the following holds after all the entries are in P_7, P_8, and P_9:

(1) $y_9^1 \oplus x_8^1 \oplus y_7^1 = y_9^2 \oplus x_8^2 \oplus y_7^2$;
(2) $x_7^2 \oplus y_8^2 \oplus x_9^2 = x_7^3 \oplus y_8^3 \oplus x_9^3$;
(3) $y_9^3 \oplus x_8^3 \oplus y_7^3 = y_9^4 \oplus x_8^4 \oplus y_7^4$;
(4) $x_7^4 \oplus y_8^4 \oplus x_9^4 = x_7^1 \oplus y_8^1 \oplus x_9^1$.

But the four constraints can be fulfilled with probability $O(q^9/2^n)$. The reason is that when $x_7^4 = x_7^1$, the constraints turn to the following three:

(1) $y_9^1 \oplus x_8^1 \oplus y_9^2 \oplus x_8^2 \oplus y_7^2 = y_9^3 \oplus x_8^3 \oplus y_7^3 \oplus y_9^4 \oplus x_8^4 (= y_7^1)$;

(2) $x_7^2 \oplus y_8^2 \oplus x_9^2 = x_7^3 \oplus y_8^3 \oplus x_9^3$;
(3) $y_8^4 \oplus x_9^4 = y_8^1 \oplus x_9^1$;

and it is not hard to check that the three can be fulfilled with probability $O(q^9/2^n)$. Hence the overall security bound increases to at least $O(q^9/2^n)$ if we use *this idea*, and the probability of cycles constitutes the new bottleneck.

We then briefly argue that $O(q^9/2^n)$ is *sufficient*. Borrowing a methodology from [LS13], consider a bad event BadMidRnd around the five middle sets:

– consider the history of n-bit values in middle sets $\mathcal{MH}$, which contains all the n-bit values in the five middle sets P_6, P_7, P_8, P_9, and P_{10}: for each $(x_i, y_i) \in P_i$, $\mathcal{MH}$ includes both x_i and y_i;
– the bad event BadMidRnd occurs, if during a random assignment in these 5 sets, the random value due to $\mathbf{P}$ equals the xor of 9 or less values in $\mathcal{MH}$.

After a preliminary analysis, we think that conditioned on $\neg$BadMidRnd, all the connected components of BM are acyclic. By a quick check, one can see that in this setting, $|P_i| \leq O(q)$, so that the complexity of the simulator is $O(q^4)$. Since the complexity of the simulator may actually affect the exact security of the construction [KPS13,CS15], this result $(q, O(q^4), O(q^4), O(q^9))$ can be seen as another meaningful security tradeoff besides the main result $(q, O(q^8), O(q^8), O(q^8))$ given in the main body (however, we stress that this is only a *preliminary analysis*).

C.2 The Second Failed Attempt

The second idea is to carefully consider each query $\mathbf{P}.\mathrm{P}(i, \cdot, \cdot)$ ($i \in \{6, 7, 8, 9, 10\}$) and check its influence on the structure of the entries in P_6, P_7, P_8, P_9, and P_{10}, and obtain an upper bound from these statistics. However, such influences not only depend on the number of entries in the five sets, but also depend on the direction of the queries which led to these entries being added; they even depend on the order of previous queries! The direction information is involved; but this is not unexpected, and can be tackled. The most problematic point is that the same previous queries in different orders may lead to different structures and different results. This is totally out of control, and does not lead to a success (at current time).

References

ABD$^+$13. Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and JohnP. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 531–550. Springer Berlin Heidelberg, 2013. full version: `http://eprint.iacr.org/2013/061.pdf`.

ABK98. Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, 174, 1998.

ABM14. Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In Shiho Moriai, editor, *Fast Software Encryption*, Lecture Notes in Computer Science, pages 348–366. Springer Berlin Heidelberg, 2014.

BDK$^+$10. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on aes-256 variants with up to 10 rounds. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 299–319. Springer Berlin Heidelberg, 2010.

BDPVA08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer Berlin Heidelberg, 2008.

BK09. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2009.

BKL$^+$12. Andrey Bogdanov, LarsR. Knudsen, Gregor Leander, François-Xavier Standaert, John Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 45–62. Springer Berlin Heidelberg, 2012.

CDMS10. Jean-Sbastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In Daniele Micciancio, editor, *Theory of Cryptography*, volume 5978 of *Lecture Notes in Computer Science*, pages 273–289. Springer Berlin Heidelberg, 2010.

CGH04. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

CHK$^+$14. Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the feistel construction. *Journal of Cryptology*, pages 1–54, 2014.

CLL$^+$14. Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John Steinberger. Minimizing the two-round even-mansour cipher. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 39–56. Springer Berlin Heidelberg, 2014.

CS14. Shan Chen and John Steinberger. Tight security bounds for key-alternating ciphers. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer Berlin Heidelberg, 2014.

CS15. Benoît Cogliati and Yannick Seurin. On the provable security of the iterated even-mansour cipher against related-key and chosen-key attacks. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 584–613. Springer Berlin Heidelberg, 2015. full version: http://eprint.iacr.org/2015/069.pdf.

DDKS14. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Cryptanalysis of iterated even-mansour schemes with two keys. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 439–457. Springer Berlin Heidelberg, 2014.

DDKS15. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Key recovery attacks on iterated even-mansour encryption schemes. *Journal of Cryptology*, pages 1–32, 2015.

DGHM13. Grégory Demay, Peter Gaži, Martin Hirt, and Ueli Maurer. Resource-restricted indifferentiability. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 664–683. Springer Berlin Heidelberg, 2013.

DKS13. Orr Dunkelman, Nathan Keller, and Adi Shamir. Slidex attacks on the even-mansour encryption scheme. *Journal of Cryptology*, pages 1–28, 2013.

DR02. Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

DRS09. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging merkle-damgård for practical applications. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer Berlin Heidelberg, 2009.

EM93. Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In Hideki Imai, RonaldL. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer Berlin Heidelberg, 1993.

FP15. Pooya Farshim and Gordon Procter. The related-key security of iterated even-mansour ciphers. In Gregor Leander, editor, *Fast Software Encryption*, volume 9054 of *Lecture Notes in Computer Science*, pages 342–363. Springer Berlin Heidelberg, 2015. full version: http://eprint.iacr.org/2014/953.pdf.

Gil14. Henri Gilbert. A simplified representation of aes. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 200–222. Springer Berlin Heidelberg, 2014.

GPPR11. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt 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 Berlin Heidelberg, 2011.

KHP07. Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-key rectangle attacks on reduced aes-192 and aes-256. In Alex Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 225–241. Springer Berlin Heidelberg, 2007.

KPS13. Eike Kiltz, Krzysztof Pietrzak, and Mario Szegedy. Digital signatures with minimal overhead from indifferentiable random invertible functions. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 571–588. Springer Berlin Heidelberg, 2013.

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 Berlin Heidelberg, 2012.

LS13. Rodolphe Lampe and Yannick Seurin. How to construct an ideal cipher from a small set of public permutations. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 444–463. Springer Berlin Heidelberg, 2013. full version: http://eprint.iacr.org/2013/255.pdf.

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*, volume 7194 of *Lecture Notes in Computer Science*, pages 285–302. Springer Berlin Heidelberg, 2012.

MRH04. Ueli 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*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer Berlin Heidelberg, 2004.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In KennethG. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer Berlin Heidelberg, 2011.

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

Ste12. John Steinberger. Improved security bounds for key-alternating ciphers via hellinger distance. Cryptology ePrint Archive, Report 2012/481, 2012. http://eprint.iacr.org/.

Ste15. John Steinberger. Block ciphers: From practice back to theory. TCC 2015 Invited Talk, 2015.

YMO08. Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky random oracle (extended abstract). In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 226–240. Springer Berlin Heidelberg, 2008.