A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical

Anja Lehmann[1] and Stefano Tessaro[2]

[1] Darmstadt University of Technology, Germany
alehmann@cdc.informatik.tu-darmstadt.de
[2] Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
tessaros@inf.ethz.ch

Abstract. The design of cryptographic hash functions is a very complex and failure-prone process. For this reason, this paper puts forward a completely *modular* and *fault-tolerant* approach to the construction of a full-fledged hash function from an underlying simpler hash function H and a further primitive F (such as a block cipher), with the property that collision resistance of the construction only relies on H, whereas indifferentiability from a random oracle follows from F being ideal. In particular, the failure of one of the two components must not affect the security property implied by the other component.
The *Mix-Compress-Mix* (MCM) approach by Ristenpart and Shrimpton (ASIACRYPT 2007) envelops the hash function H between two *injective* mixing steps, and can be interpreted as a first attempt at such a design. However, the proposed instantiation of the mixing steps, based on block ciphers, makes the resulting hash function impractical: First, it cannot be evaluated online, and second, it produces larger hash values than H, while only inheriting the collision-resistance guarantees for the shorter output. Additionally, it relies on a *trapdoor* one-way permutation, which seriously compromises the use of the resulting hash function for random oracle instantiation in certain scenarios.
This paper presents the first *efficient* modular hash function with online evaluation and short output length. The core of our approach are novel block-cipher based designs for the mixing steps of the MCM approach which rely on significantly weaker assumptions: The first mixing step is realized *without* any computational assumptions (besides the underlying cipher being ideal), whereas the second mixing step only requires a one-way permutation *without* a trapdoor, which we prove to be the minimal assumption for the construction of injective random oracles.

1 Introduction

Multi-Property Hash Functions. Cryptographic hash functions play a central role in efficient schemes for several cryptographic tasks, such as message authentication, public-key encryption, digital signatures, key derivation, and many

others. Yet the huge variety of contexts in which hash functions are deployed makes the security requirements on them very diverse: While some schemes only assume relatively simple properties such as *one-wayness* or different forms of *collision resistance*, other schemes, including practical ones such as OAEP [4, 15] and PSS [5], are only proven secure under the assumption that the underlying hash function is a *random oracle* [3], i.e., a truly random function which can be evaluated by the adversary. On the one hand, while a number of *provably-secure* collision-resistant hash functions, such as VSH [9] or SWIFFT [18], have been designed, they are not appropriate candidates for random oracle instantiation. On the other hand, well-known theoretical limitations [8, 19] only permit constructions of hash functions for random oracle instantiation from *idealized* primitives [10], such as a *fixed-input-length* random oracle or an *ideal cipher*,[3] but (as first pointed out in [2]) these constructions may lose any security guarantees as soon as the adversary gets to exploit non-ideal properties of the underlying primitive.[4]

While one could in principle always employ a suitable hash function tailored at the individual security property needed by one particular cryptographic scheme at hand, common practices such as code re-use and the development of standards call for the design of a *single* hash function satisfying as many properties as possible. This point of view has also been adopted by NIST's on-going SHA-3 competition [17], and motivated a series of works [2, 1] shifting the design problem of multi-property hash functions to the task of constructing good multi-property *compression* functions. A further line of research has been devoted to robust multi-property *combiners* [13], which merge two hash functions such that the resulting function satisfies each of the properties possessed by at least one of the two starting functions. While these works simplify the design task, building multi-property hash functions from *single*-property primitives remains far from being simple, and is the main topic of this paper.

Statement of the Main Problem. This paper presents a *modular* design for hash functions that are *collision resistant* in the *standard* model and can, simultaneously, be used for random oracle instantiation in the *ideal* model. We consider a setting where both a hash function H as well as some other (potentially ideal) primitive F (such as a block cipher) are given (a similar setup was previously considered by Ristenpart and Shrimpton [23]): We aim at devising a construction $\mathsf{C}^{H,F}$ which is collision resistant as long as H is collision resistant,[5] and which behaves as a random oracle (with respect to the notion of *indifferentiability* [19, 10]) whenever F is ideal. For this approach to be practically appealing, the construction must preserve the good properties of H: For

[3] An ideal cipher $\mathbf{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ associates an (invertible) random permutation $\mathbf{E}(k,\cdot)$ with each key k.

[4] Of course, a real block cipher *cannot* be ideal. (Likewise, a hash function cannot be a random oracle either.) Yet modeling it as ideal captures the adversary's inability of exploiting any structure, and a security proof in this model implies in particular the inexistence of any generic attacks treating the block cipher as a black box.

[5] In particular, we require the existence of a standard-model reduction.

instance, it must allow for online processing of data (which is crucial for large inputs or in streaming applications) whenever H can be evaluated online.[6] Also, the construction should not increase the size of the hashes of H.

In particular, we advocate a safe and modular design paradigm where each of both properties should ideally rely only on *one* of both component primitives, whereas the other primitive may be arbitrarily insecure, except for (possibly) satisfying some minimal structural requirement (that can be ensured by design), such as F being a permutation or H being sufficiently regular. This differs from the point of view taken in [23], where H is *guaranteed* to be collision resistant and is extended by means of an ideal primitive F into a random oracle, while preserving the collision-resistance guarantees of H: We believe that practical considerations, especially efficiency, may in fact motivate the use of hash functions with no provable security guarantees. Thus, it is desirable that even the ability of finding collisions for H does not impact the indifferentiability of the construction, as long as F is still ideal. Either way, both points of view are related: Any solution satisfying our stronger requirements (including the one we propose in this paper) also fits within the framework of [23], while the solution proposed in [23] also satisfies stronger requirements, as discussed below.

We also remark that using the multi-property combiner of [14] one can combine a random oracle (built from F) and H into a hash function that provably observes both properties. However, as combiners inherently do not exploit the knowledge of which one of both functions has a certain property, the resulting construction is rather inefficient, e.g., it doubles the output length.

The MCM Approach. Given a hash function H as above, the so-called *mix-compress-mix (MCM)* approach, introduced by Ristenpart and Shrimpton [23], considers the construction

$$\mathsf{MCM}^{M_1,M_1,H}(x) := M_2(H(M_1(x))),$$

where M_1 and M_2 are arbitrary-input-length injective maps (the so-called *mixing stages*) with *stretch* τ_1 and τ_2, respectively, i.e., such that M_i outputs a string of length $|x| + \tau_i$ on input $x \in \{0,1\}^*$. The injectivity of the mixing stages ensures that MCM preserves the collision resistance of H in the standard model. Additionally, it was shown in [23] that MCM is indifferentiable from a random oracle if M_1 and M_2 are random *injective* oracles (i.e., M_i returns a random $(|x| + \tau_i)$-bit string for each input $x \in \{0,1\}^*$ that differs from all previously returned values with the same length) *and* H is collision resistant and sufficiently regular. Dodis et al. [12] subsequently interpreted this result as the combination of two facts: (i) The mapping $x \mapsto H(M_1(x))$ is *preimage aware*[7] under the same

[6] Most hash functions rely on some iterated (and thus inherently online) design, such as Merkle-Damgård [11, 21], or sponges [6].

[7] Informally, a construction $\mathsf{C}^{\mathbf{F}}$ based on an ideal primitive $\mathbf{F}$ is *preimage aware* if there exists an algorithm – called the *preimage extractor* – which given the input-output history of $\mathbf{F}$ and an output y, either aborts or returns x such that $\mathsf{C}^{\mathbf{F}}(x) = y$, and after such query no adversary can find an input x' such that $\mathsf{C}^{E}(x') = y$ (and $x' \neq x$ in case the extraction query did not abort).

assumptions, and (ii) Post-processing the output of a preimage-aware function with a (possibly injective) random oracle yields a full-fledged random oracle. A concrete instantiation of injective random oracles – called the TE-construction – relying on an ideal cipher and a *trapdoor* one-way permutation has also been proposed in [23]: To date, this was the only known such construction.

Interestingly, we observe that the MCM approach provides a modular design approach for hash functions as advocated above, since the indifferentiability result can be made *independent* of the collision resistance of H. (This was unnoticed in [23], and is briefly discussed in the full version of this paper.) However, its deployment is subject to a number of practical and theoretical drawbacks, whose solution was stated as an open problem in [23]: First, *every* construction of injective random oracles (and in particular the TE-construction) cannot be *online*, as, roughly speaking, each output bit needs to be influenced by *all* of the input in order to exhibit random behavior. Additionally, the fact that the TE-construction is length-increasing has a serious impact on the resulting hash size: In particular, the stretch τ_i typically equals the bit length of a sufficiently secure RSA modulus, i.e., $\tau_i \geq 2048$ bits for reasonable security. Finally, the use of a *trapdoor* one-way permutation within the TE-construction is rather undesirable: In contrast to (non trapdoor) one-way permutations, the assumption is very strong, e.g., it implies public-key encryption in the random oracle model [4]. Also, as pointed out in [23], the compositional guarantees of protocols using the MCM approach (with the TE-construction) to instantiate a random oracle are affected, as properties such as deniability may be lost (cf. e.g. the works by Pass [22] and by Canetti et al. [7]).

These observations give rise to a number of challenging open questions. Can we instantiate the *first* mixing stage of MCM with a weaker primitive which allows for online processing? Can we instantiate the *second* mixing stage (where online processing is not an issue) as an injective RO with limited stretch (possibly even with no stretch at all)? And finally, can we weaken the underlying assumption, eliminating the need of the trapdoor, or possibly even entirely removing the underlying assumption?

Contributions and Roadmap of this Paper. In this paper, we present the first efficient modular construction of a hash function in the sense described above. Our solution relies on the MCM approach, and in particular we address and solve all of the aforementioned open questions, and hence make a substantial step towards making the MCM approach practical.

First Mixing Stage. In Section 3, we present a novel mode of operation for a block cipher $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ implementing an arbitrary-input-length injective map – called *iterated mix* (IM) – that permits *online* processing of its inputs, making only one call to E per n-bit message block, and has only stretch $n/2$. Our first main theorem shows that the construction $\mathsf{IMC}^{E,H}(M) := H(\mathsf{IM}^E(M))$ applying H to the output of IM is preimage aware if E is an ideal cipher and, additionally, the hash function H satisfies a rather weak regularity requirement (which is somewhat incomparable to the one used in [23], albeit equally natural): Namely, given a random n-bit string m and some arbitrary

string S, the value $H(S\|m)$ has (min-)entropy not much lower than n (if n is smaller than H's hash size), or not much lower than the hash size otherwise. In fact, even completely insecure hash functions can have this property, and it is also natural to assume that it is satisfied by any reasonably built hash function. We also present a variant of the IM-construction which requires a block cipher with single-block key length n at the price of making two block-cipher calls per message block.

We stress that (contrary to the TE-construction) our result does not rely on *any* computational assumptions: In particular, the IM-construction relies on invertible primitives, and is itself *efficiently* invertible. Thus, IM does not implement a random injective oracle.

Second Mixing Stage. With the goal of making the MCM approach preserve the hash size of the underlying hash function in mind, the second part of this paper (Section 4) addresses the question of building *length-preserving* injective random oracles. (We call this a (non-invertible) *random permutation oracle* (RPO).) We show that for any three permutations E, E', π from n bits to n bits, the permutation

$$\mathsf{NIRP}^{E,E',\pi}(x) := E'(\pi(E(x)))$$

is indifferentiable from a RPO if both E and E' are (fixed-key) ideal ciphers, and π is a one-way permutation, without a trapdoor.

In practice, E, E' are instantiated by a block cipher with two distinct fixed keys. This limits us to n being a valid block size (e.g. $n = 128$ bits), which can be smaller than the usual hash size (e.g. $h = 256$). This motivates the question of extending the input/output size of random permutation oracles: In Section 4.2, we present constructions (which are reminiscent of the Shrimpton-Stam compression function [25]) for extending every n to n bits RPO into a $\gamma \cdot n$ bits to $\gamma \cdot n$ bits RPO for any fixed $\gamma > 1$.

In the full version we further show that in order to construct injective ROs the assumption of a one-way permutation cannot be weakened to a one-way function (at least under black-box security reductions).

Putting Pieces Together. Finally, instantiating MCM with IM and NIRP (or its extension through our extender) as its first and second mixing stage, respectively, leads to the first construction of a hash function with the following properties:

(i) Its collision resistance can be reduced in the standard model to the collision resistance of the underlying hash function.
(ii) It is indifferentiable from a random oracle in the ideal cipher model (with a one-way permutation), as long as the underlying hash function is sufficiently regular.
(iii) It can be evaluated online as long as the underlying hash function can be evaluated in an online fashion.
(iv) It has hash size equal to the one of the underlying hash function.
(v) It can be used to instantiate a random oracle in *all* computationally secure schemes in the random oracle model, with no composability limitations.

2 Preliminaries

Notational Preliminaries. Throughout this paper, $\{0,1\}^n$ denotes the set of strings s of length $|s| = n$, whereas $(\{0,1\}^n)^*$ and $(\{0,1\}^n)^+$ are the sets of strings consisting of n-bit blocks with and without the empty string, respectively. The notation $s\|s'$ stands for the concatenation of the strings s and s'. Also, we use $\mathsf{INJ}(m,n)$ to denote the set of *injective* functions $f : \{0,1\}^m \to \{0,1\}^n$ (in particular, $\mathsf{INJ}(n,n)$ is the set of permutations from n bits to n bits). Further, it is convenient to define $\mathsf{BC}(\kappa,n)$ as the set of *block ciphers*, i.e., of *keyed* functions $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ such that each key $k \in \{0,1\}^\kappa$ defines a permutation $E_k(\cdot) := E(k,\cdot) \in \mathsf{INJ}(n,n)$ (and denote as $E^{-1}(k,\cdot)$ the corresponding *inverse*).

Algorithms are in general randomized, and throughout this paper we fix a RAM model of computation for these algorithms. We use the notation $\mathcal{A}^{\mathcal{O}}(r)$ to denote the (oracle) algorithm $\mathcal{A}^{(\cdot)}$ which runs on input r with access to the oracle $\mathcal{O}$. In particular, an algorithm $\mathcal{A}^{(\cdot)}$ is said to have *running time* t (also denoted as $\mathrm{time}(\mathcal{A}) = t$) if the sum of its description length and the worst-case number of steps it takes (counting oracle queries as single steps), taken over all randomness values, all inputs and all compatible oracles, is at most t. If the algorithm takes inputs of arbitrary length, then $\mathrm{time}(\mathcal{A},\ell)$ refines the above notion to only take the maximum over inputs of length at most ℓ.

Finally, the shorthand $x \xleftarrow{\$} S$ stands for the action of drawing a fresh random element x uniformly from the set S, whereas $x \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(r)$ denotes the process of sampling x by letting $\mathcal{A}$ interact with $\mathcal{O}$ on input r (and probabilities are taken over the random coins of $\mathcal{A}$ and $\mathcal{O}$).

One-Way Functions and Permutations. We define the *one-way advantage* of an adversary $\mathcal{A}$ against a function $f : \{0,1\}^m \to \{0,1\}^n$ as

$$\mathbf{Adv}^{\mathrm{owf}}_f(\mathcal{A}) = \mathsf{P}[x \xleftarrow{\$} \{0,1\}^m,\, x' \xleftarrow{\$} \mathcal{A}(f(x)) : f(x) = f(x')].$$

For the special case of a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$, it is convenient to use the shorthand $\mathbf{Adv}^{\mathrm{owp}}_\pi(\mathcal{A}) = \mathsf{P}[x \xleftarrow{\$} \{0,1\}^n,\, x' \xleftarrow{\$} \mathcal{A}(\pi(x)) : x = x']$ for the *one-way permutation* advantage.

Idealized primitives. We consider a number of (more or less) standard *idealized primitives* throughout this paper, which are always denoted by bold-face letters. For a set X, a *random oracle (RO)* $\mathbf{R} : X \to \{0,1\}^n$ is a system associating a random n-bit string $\mathbf{R}(x)$ with each input x. If $X = \{0,1\}^m$, then $\mathbf{R}$ is called a *fixed-input-length RO* (FIL-RO), whereas it is a *variable-input-length* RO (VIL-RO) if $X = \{0,1\}^*$. An *ideal cipher (IC)* $\mathbf{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher E chosen uniformly from the set $\mathsf{BC}(\kappa,n)$, and allows both *forward queries* $\mathbf{E}(k,x)$ as well as *backward queries* $\mathbf{E}^{-1}(k,y)$. If $\kappa = 0$, then we omit the first input and we call this a *fixed-key ideal cipher*. Note that for an IC $\mathbf{E}$ and distinct fixed key values $k_0, k_1, \ldots$, $\mathbf{E}(k_0,\cdot), \mathbf{E}(k_1,\cdot), \ldots$ are independent fixed-key ICs. In contrast, a *(fixed-input-length) random injective oracle (FIL-RIO)* $\mathbf{I} : \{0,1\}^m \to \{0,1\}^n$ implements a uniformly chosen function from

$\mathsf{INJ}(m,n)$. In the special case $m = n$ we call this a *random permutation oracle (RPO)* $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$.

We stress that the substantial difference between a fixed-key IC and a RPO is that the former allows for inversion queries, whereas the latter does not (and is in particular hard to invert).

INDIFFERENTIABILITY. The notion of indifferentiability was introduced by Maurer et al. [19] to generalize indistinguishability to constructions $\mathsf{C}^{\mathbf{F}} : X \to \{0,1\}^n$ using a *public* (idealized) primitive $\mathbf{F}$ (e.g., an IC, a FIL-RO, or a combination of these), i.e., that can be accessed by the adversary. Roughly speaking, $\mathsf{C}^{\mathbf{F}}$ is *indifferentiable* from an ideal primitive $\mathbf{F}'$ if there exists a simulator $\mathcal{S}^{\mathbf{F}'}$ accessing $\mathbf{F}'$ such that $(\mathsf{C}^{\mathbf{F}}, \mathbf{F})$ and $(\mathbf{F}', \mathcal{S}^{\mathbf{F}'})$ are indistinguishable. In particular, we will be concerned with the cases where $\mathbf{F}'$ is either a RO or a RIO/RPO, and we define the *RO-indifferentiability advantage* of the distinguisher $\mathcal{D}$ against the construction $\mathsf{C}^{\mathbf{F}}$ and simulator $\mathcal{S}$ as the quantity

$$\mathbf{Adv}^{\text{ind-ro}}_{\mathsf{C}^{\mathbf{F}},\mathcal{S}}(\mathcal{D}) \;=\; \left|\mathsf{P}\left[\mathcal{D}^{\mathsf{C}^{\mathbf{F}},\mathbf{F}} = 1\right] - \mathsf{P}\left[\mathcal{D}^{\mathbf{R},\mathcal{S}^{\mathbf{R}}} = 1\right]\right|,$$

where $\mathbf{R} : X \to \{0,1\}^n$ is a RO with the same input and output sets as C. The *IRO-indifferentiability advantage* $\mathbf{Adv}^{\text{ind-iro}}_{\mathsf{C}^{\mathbf{F}},\mathcal{S}}$ is defined analogously by using a RIO $\mathbf{I}$ instead of $\mathbf{R}$. We stress that both quantities are related by a simple birthday-like argument, i.e., $\mathbf{Adv}^{\text{ind-iro}}_{\mathsf{C}^{\mathbf{F}},\mathcal{S}}(\mathcal{D}) \le \mathbf{Adv}^{\text{ind-ro}}_{\mathsf{C}^{\mathbf{F}},\mathcal{S}}(\mathcal{D}) + \frac{1}{2}\cdot(q+q_{\mathcal{S}})^2 \cdot 2^{-n}$, where q is the number of query $\mathcal{D}$ makes to its first oracle, whereas $q_{\mathcal{S}}$ is the overall number of queries $\mathcal{S}$ makes when answering $\mathcal{D}$'s queries. Note that indifferentiability ensures *composability*, i.e., if a cryptographic scheme is secure using an ideal primitive $\mathbf{F}'$ accessible by the adversary, then it remains secure when replacing $\mathbf{F}'$ with a construction $\mathsf{C}^{\mathbf{F}}$ which is indifferentiable from $\mathbf{F}'$ and letting the adversary access $\mathbf{F}$. See [19, 10] for a formal treatment in the information-theoretic and computational models.

COLLISION-RESISTANCE. Let $H : K \times \{0,1\}^* \to \{0,1\}^h$ be a (keyed) hash function with key generator $\mathcal{K}$. The *collision-finding advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\text{cr}}_{H}(\mathcal{A}) := \mathsf{P}[k \xleftarrow{\$} \mathcal{K}, (M, M') \xleftarrow{\$} \mathcal{A}(k) : M \neq M' \wedge H_k(M) = H_k(M')]$$

The notion naturally extends to keyless hash functions (which can be considered in the same spirit proposed in [24]) and to constructions from some ideal primitive $\mathbf{F}$ (where $\mathcal{A}$ is additionally given access to $\mathbf{F}$).

THE MCM-CONSTRUCTION. For a hash function $H : \{0,1\}^* \to \{0,1\}^h$, and injective maps $\mathsf{M}_1 : \{0,1\}^* \to \{0,1\}^*$, $\mathsf{M}_2 \in \mathsf{INJ}(h', n)$, where $n \ge h' \ge h$, the MCM-*construction* implements a map $\{0,1\}^* \to \{0,1\}^n$ as

$$\mathsf{MCM}^{\mathsf{M}_1,H,\mathsf{M}_2}(M) := \mathsf{M}_2(H(\mathsf{M}_1(M)) \,\|\, 0^{h'-h}).$$

We also define $\mathsf{MCM}^{\mathsf{M}_1,H,\mathsf{M}_2}_k := \mathsf{MCM}^{\mathsf{M}_1,H_k,\mathsf{M}_2}$ for all $k \in K$ if the hash function H is keyed (with key space K) . Also, the definition does not allow $\mathsf{M}_1, \mathsf{M}_2$ to be

keyed (in contrast to [23]). This is because we will present *keyless* instantiations of $\mathsf{M}_1, \mathsf{M}_2$. Note that we assume M_2 to be fixed input length without loss of generality. The following simple result was shown in [23], and holds both for keyed as well as for keyless hash functions.

Lemma 1. *For all collision-finding adversaries $\mathcal{A}$ outputting a pair of messages each of length at most ℓ, there exists a collision-finding adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{MCM}^{\mathsf{M}_1,H,\mathsf{M}_2}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{cr}}_{H}(\mathcal{B})$, where $\mathrm{time}(\mathcal{B}) = \mathrm{time}(\mathcal{A}) + \mathcal{O}(2(\ell + \mathrm{time}(\mathsf{M}_1, \ell)))$.*

Preimage Awareness. We briefly review the notion of preimage awareness [12] for a hash function $\mathsf{H}^{\mathbf{F}} : \{0,1\}^* \to \{0,1\}^h$ built from an idealized primitive $\mathbf{F}$. A *preimage extractor* $\mathcal{E}$ is a (deterministic) algorithm taking a history α of input-output pairs of $\mathbf{F}$ and a value $y \in \{0,1\}^h$ such that $\mathcal{E}(\alpha, y)$ returns a value $x \in \{0,1\}^* \cup \{\perp\}$. We consider a random experiment (called the pra-*game*) involving an adversary $\mathcal{A}$ which can query *both* $\mathbf{F}$ and $\mathcal{E}(\alpha, \cdot)$ (where α is the current history containing the interaction with $\mathbf{F}$ so far, i.e., the adversary cannot change the first argument), and where a set Q contains all $\mathcal{E}$-queries y of $\mathcal{A}$ and an associative array V stores as $\mathsf{V}[y] \in \{0,1\}^* \cup \{\perp\}$ (for all $y \in Q$) the answer of the query y to $\mathcal{E}$. The pra-*advantage of the adversary $\mathcal{A}$ with preimage extractor $\mathcal{E}$, and primitive* $\mathbf{F}$ is the quantity

$$\mathbf{Adv}^{\mathrm{pra}}_{\mathsf{H},\mathbf{F},\mathcal{E}}(\mathcal{A}) := \mathsf{P}[(M, y) \xleftarrow{\$} \mathcal{A}^{\mathcal{E}(\alpha,\cdot),\mathbf{F}} : y \in Q \wedge \mathsf{H}^{\mathbf{F}}(M) = y \wedge \mathsf{V}[y] \neq M].$$

It turns out that preimage aware functions are good domain extenders for FIL-ROs: More concretely, with H as above, consider the construction $\mathsf{C}^{\mathbf{F},\mathbf{R}'} : M \mapsto \mathbf{R}'(\mathsf{H}^{\mathbf{F}}(M))$ for a FIL-RO $\mathbf{R}' : \{0,1\}^h \to \{0,1\}^n$. Then, the following result was proved in [12].

Lemma 2 (PRA + FIL-RO = VIL-RO [12]). *There exists a simulator $\mathcal{S}$ such that for all distinguishers $\mathcal{D}$ making q queries to $\mathsf{C}^{\mathbf{F},\mathbf{R}'}$ of length at most ℓ, q_1 queries to $\mathbf{F}$ and q_2 queries to $\mathbf{R}'$, there exists an adversary $\mathcal{A}$ with*

$$\mathbf{Adv}^{\text{ind-ro}}_{\mathsf{C}^{\mathbf{F},\mathbf{R}'},\mathcal{S}}(\mathcal{D}) \leq \mathbf{Adv}^{\mathrm{pra}}_{\mathsf{H},\mathbf{F},\mathcal{E}}(\mathcal{A}).$$

The simulator $\mathcal{S}$ runs in time $\mathcal{O}(q_1 + q_2 \cdot \mathrm{time}(\mathcal{E}))$ and makes q_2 queries, whereas $\mathcal{A}$ runs in time $\mathrm{time}(\mathcal{D}) + \mathcal{O}(q \cdot \mathrm{time}(\mathsf{H}, \ell) + q_0 + q_1)$ and makes $q \cdot q_{\mathsf{H},\ell} + q_1$ $\mathbf{F}$-queries and q_2 extraction queries, where $q_{\mathsf{H},\ell}$ is the maximal number of oracle queries made by H to process an input of length at most ℓ.

3 An On-Line Mixing Stage: The IMC-Construction

3.1 Description

The IM-Construction. The *iterated mix construction* (or IM-construction for short), depicted in Figure 1, relies on a block cipher $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ and an injective mapping $\mathsf{PAD} : \{0,1\}^* \to \{0,1\}^{n/2} \times (\{0,1\}^n)^*$ which

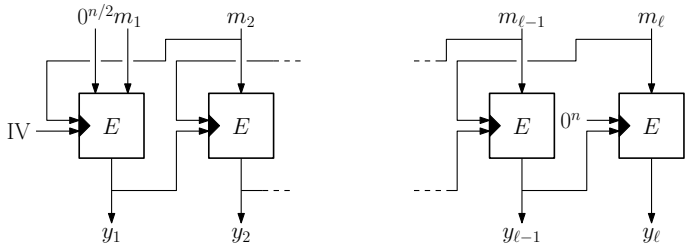

Fig. 1. The IM-construction with block cipher $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$.

pads every string so that it consists of one $n/2$-bit block, followed by as many n-bit blocks as necessary.[8] On input $M \in \{0,1\}^*$, it first obtains $\mathsf{PAD}(M) = m_1 \| \ldots \| m_\ell$, and computes the output $y_1 \| \ldots \| y_\ell$ iteratively such that $y_1 := E(\mathrm{IV} \| m_2, 0^{n/2} \| m_1)$ (where IV is an n-bit fixed *initialization value*) and $y_i := E(y_{i-1} \| m_{i+1}, m_i)$ for all $i = 1, \ldots, \ell$, where $m_{\ell+1} := 0^n$.

In contrast to the TE-construction of [23], the IM-construction is *iterated* and allows for (essentially) online processing, with the minimal restriction that only the first $i-1$ output blocks $y_1, \ldots, y_{i-1}$ can be computed from the first i message blocks $m_1, \ldots, m_i$. This one-block-lookahead evaluation strategy only marginally impacts the efficiency of the construction, and is crucial in order to ensure the desired security requirements.

Injectivity of the IM-Construction. It is not difficult to see that the construction is injective: Given an output $y_1 \| \ldots \| y_\ell$ (for some ℓ) we can iteratively *efficiently* reconstruct the padding $m_1 \| \ldots \| m_\ell$ of the input M by computing $m_i := E^{-1}(y_{i-1} \| m_{i+1}, m_i)$ for all $i = \ell, \ell-1, \ldots, 2$, with $m_{\ell+1} = 0^n$, and finally $0^{n/2} \| m_1 := E^{-1}(\mathrm{IV} \| m_2, y_1)$. Thus, IM *cannot* be a VIL-RIO, and not even one way, even though it is surprisingly still strong enough to instantiate the first mixing step of the MCM approach, as we show below.

The IMC-Construction. It is convenient to define the combination of the IM-construction and a hash function H as the *iterated mix-compress construction* (or IMC-construction, for short), which, on input a string $M \in \{0,1\}^*$, outputs $\mathsf{IMC}^{E,H}(M) := H(\mathsf{IM}^E(M))$. If H is keyed, then we similarly define the keyed function $\mathsf{IMC}_k^{E,H}(M) := \mathsf{IMC}^{E,H_k}(M)$. Note that if H can be evaluated online, then this is the case for the IMC-construction as well.

Shorter Key Size. The use of a block cipher with key length equal twice the block length is acceptable in practice.[9] Still, in oder to ensure compatibility with a larger number of block ciphers, we propose an alternative construction (called the DM-IM-construction) which relies on a block cipher $E : \{0,1\}^n \times \{0,1\}^n \to$

[8] This can be done in the canonical way by appending the bit 1 followed by as many 0 bits as necessary in order to fulfill the length requirement.

[9] For instance, AES supports key size 256 bits with block length $n = 128$ bits.

$\{0,1\}^n$, at the cost of making *two* calls per processed message block. The underlying idea consists of producing an n-bit key value at each round by using the Davies-Meyer construction on y_{i-1} and m_{i+1}: More precisely, we compute $y_1 := E(E(m_2, \mathrm{IV}) \oplus \mathrm{IV}, 0^{n/2} \| m_1)$ and $y_i := E(E(m_{i+1}, y_{i-1}) \oplus y_{i-1}, m_i)$ for all $i = 2, \ldots, \ell$. As above, for a hash function H, we define $\mathsf{DM\text{-}IMC}^{E,H}(M) := H(\mathsf{DM\text{-}IM}^{E}(M))$. (And analogously for the keyed case.)

3.2 Preimage Awareness

The purpose of this section is to prove that, for an ideal cipher $\mathbf{E} : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$, the construction $\mathsf{IMC}^{\mathbf{E},H}$ is preimage aware, provided H satisfies very weak randomness-preserving properties that we discuss first.

Hash Function Balance. The IMC-construction does not exhibit any useful properties if H can be arbitrary (consider e.g. the case where H is constant). It is nevertheless reasonable to assume H to satisfy minimal structural properties which could be (and generally are) ensured by design. In particular, we require H to preserve some of the randomness of a uniformly chosen input m of a given length n (where n is e.g. the block length of the cipher used in the IM-construction), and this should hold even if m is appended to some other fixed input string M.

Definition 1. *An (unkeyed) hash function $H : \{0,1\}^* \to \{0,1\}^h$ is (ϵ, n)-prefix-balanced if for all messages $M \in (\{0,1\}^n)^*$ and hash function outputs $y \in \{0,1\}^h$ we have $\mathsf{P}[m \xleftarrow{\$} \{0,1\}^n : H(M\|m) = y] \leq \epsilon$.*

The notion extends naturally to a *keyed* hash function $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^h$: We say that it is (ϵ, n)-prefix balanced if for all keys k the function H_k is $(\epsilon(k), n)$-prefix balanced, and $\sum_k \mathsf{P}(k) \cdot \epsilon(k) \leq \epsilon$, where $\mathsf{P}(k)$ is the probability that the key generator samples the key k. We remark that the best ϵ one can hope for is $\epsilon = 2^{-n}$ as long as $n \leq h$ holds, whereas $\epsilon \geq 2^{-h}$ for $n \geq h$. Note that our notion is somewhat incomparable to the one of [23], where on the one hand balancedness under variable input lengths is considered (rather than for some fixed length n, as in our case), but, on the other hand, the property is not required under prepending of fixed prefixes: Still we find this extension to be natural in a hashing scenario. It is important to realize that prefix balancedness does not imply any useful security properties for H: The function $H : (\{0,1\}^n)^+ \to \{0,1\}^n$ such that $H(M\|m) := m$ for all n-bit strings m and all M with length multiple of n is $(n, 2^{-n})$-prefix-balanced, despite finding collisions or preimages in this function being trivial.

Main Theorem. The following theorem is the main result of the first part of this paper: It provides a *concrete* characterization of the security of the IMC-construction in the ideal-cipher model. We stress that the result only relies on E being an ideal cipher, and H being sufficiently balanced, but no computational assumption is made, i.e., the result holds with respect to computationally unbounded adversaries.

Theorem 1 (Preimage Awareness of IMC). *Let* $\mathbf{E} : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ *be an ideal cipher and let* $H : \{0,1\}^* \to \{0,1\}^h$ *be an* (ϵ, n)*-prefix-balanced hash function. There exists a preimage extractor* $\mathcal{E}$ *(given in the proof) such that, for all adversaries* $\mathcal{A}$ *issuing at most* q *queries to* $\mathbf{E}$ *and* $q_{\mathcal{E}}$ *queries to* $\mathcal{E}$*, we have*

$$\mathbf{Adv}^{\mathrm{pra}}_{\mathsf{IMC}^{\mathbf{E},H},\mathbf{E},\mathcal{E}}(\mathcal{A}) \leq 3 \cdot q(q+1) \cdot 2^{-(n+1)} + q \cdot 2^{-n/2} + q(q+2q_{\mathcal{E}}) \cdot \tfrac{\epsilon}{2}.$$

Furthermore, $\mathcal{E}$ *answers an extraction query in time* $\mathcal{O}(|\alpha| \cdot \log|\alpha|)$.

The result extends naturally to a keyed hash function by just averaging the bound over all choices of the key. The security of IMC is bounded by (roughly) $\min\{2^{n/2}, \sqrt{\epsilon}\}$, and is not worse than the one in the TE-construction (which additionally relies on the security of the underlying trapdoor one-way permutation). Note that Theorem 1 is concerned with the entire IMC-construction: An interesting (and seemingly challenging) open question consists of distilling the (minimal) properties needed by IM to yield preimage awareness for IMC.

The remainder of this section is devoted to the proof outline of Theorem 1. Technical details are postponed to the full version, as well as a discussion on how to obtain similar bounds for DM-IMC.

Interaction Graphs. An interaction with the ideal cipher $\mathbf{E}$ can be described in terms of the *history* α, consisting of triples (k, x, y), where $k \in \{0,1\}^{2n}$, and $x, y \in \{0,1\}^n$. Both a forward query $\mathbf{E}(k, x)$ with output y and a backward query $\mathbf{E}^{-1}(k, y)$ with output x result in a triple (k, x, y) being added to α.[10] However, it is far more convenient to describe α in terms of a directed (edge labeled) graph $G = G(\alpha) = (V, E)$ with vertex set $V := \{0,1\}^n$ and edge set $E \subseteq V \times V$ such that $(y, y') \in E$ with labels $\mathsf{label}(y, y') = m$ and $\mathsf{next}(y, y') = m'$ if (i) $(y\|m', m, y') \in \alpha$ with $y \neq \mathrm{IV}$ or (ii) $(y\|m', 0^{n/2}\|m, y') \in \alpha$ if $y = \mathrm{IV}$. A (directed) path $\mathrm{IV} = y_0 \to y_1 \to \cdots \to y_\ell$ in G is called *valid* if for all $i = 1, \ldots, \ell - 1$ we have $\mathsf{label}(y_i, y_{i+1}) = \mathsf{next}(y_{i-1}, y_i)$. It is additionally called *complete* if $\mathsf{next}(y_{\ell-1}, y_\ell) = 0^n$. The *value* of a complete valid path is defined as $H(y_1\| \ldots \|y_\ell)$, and its *preimage* is the string M which is padded to $\mathsf{label}(y_0, y_1)\| \cdots \|\mathsf{label}(y_{\ell-1}, y_\ell)$.

The Preimage Extractor $\mathcal{E}$. On input a history α and a (potential) output $z \in \{0,1\}^h$ of IMC, the preimage extractor $\mathcal{E}$ first computes the subgraph G' of $G(\alpha)$ induced by the vertices which are reachable through a valid path. If G' is not a directed tree, then $\mathcal{E}$ aborts and outputs $\perp$. Otherwise, if G' contains one single valid complete path with value z and preimage M, it outputs M. In any other case, it outputs $\perp$.

It is not hard to see that $\mathcal{E}$ can be implemented with running time $\mathcal{O}(|\alpha| \cdot \log|\alpha|)$ (i.e., where $|\alpha|$ approximately equals the number of edges in the graph

[10] The actual history used in the definition of preimage awareness indeed contains more information, such as whether the triple is added by a forward or by a backward query, but this is irrelevant in the following.

$G(\alpha)$) due to the fact that $\mathcal{E}$ aborts if G' is not a tree: Otherwise, the number of possible valid paths may be very high, even exponential.[11]

Proof Intuition. Assume without loss of generality that the adversary $\mathcal{A}$ never repeats a query twice[12] and that whenever it terminates in the pra-game outputting a pair (M, z), it has made all queries necessary to evaluate the IMC-construction on input M (with output z). In other words, the interaction graph $G(\alpha)$ of the final history α contains a valid complete path with preimage M and value z. But because the query z was previously issued to $\mathcal{E}$, if $\mathcal{A}$ wins the game, one of the following has to occur: (i) The subgraph of the valid paths is *not* a directed tree, (ii) No valid path with value z existed when the $\mathcal{E}$-query z was issued, but such a path was created afterwards, or (iii) There exist at least two valid paths with value z. We show that these events are unlikely.

A key step is proving that, with very high probability, valid paths are constructed only by means of *forward* queries: A construction of a valid path by backward queries may be successful either because we can "connect" the path with an already existing one (built by forward queries), or because we construct the entire path backwards. However, both cases turn out to be unlikely: In the former case, a fresh backward query outputs a random m (under the permutation property), and this can only be the next-label for an already existing edge with low probability. (This motivates the one-block-lookahead strategy in IM.) In the latter case, it is very unlikely to have all of the first $n/2$ bits returned by the last evaluation query being equal to 0. (This motivates the padding in the first block.) However, if a path is generated only by forward queries, we can ensure that the value of a valid path is always sufficiently random due to the prefix-balancedness of H. We refer the reader to the full version for a formalization of this argument.

This highlights a very intriguing property of the IM-construction: Although it can be efficiently inverted on any *valid* output, it is very unlikely that we can come up with such a valid output without first evaluating the construction. (In particular, this prevents that even a known collision for H will lead to a valid collision for the IMC-construction.)

4 A Length-Preserving Mixing Stage: Random Permutation Oracles

Post-processing the output of the IMC-construction with a random injective oracle yields a full-fledged random oracle (by Theorem 1 and Lemma 2), whose collision resistance can be reduced to the one of the underlying function H in the

[11] One may argue that we are taking a rather conservative approach: Even if the graph were not a tree, it would most likely have a limited number of valid paths. Still, this considerably simplifies the security analysis with no noticeable loss in the obtained bounds.

[12] In particular, if $\mathcal{A}$ asks a forward query $\mathbf{E}(k, x)$ which is answered by y, the matching backward query $\mathbf{E}^{-1}(k, y)$ is never issued. (And vice versa.)

standard model by Lemma 1. The use of the TE-construction [23] for this task is subject to two main drawbacks: It requires a *trapdoor* one-way permutation and also enlarges the output of the compressing stage. (The lack of online evaluation capabilities is not a restriction, as we have to process only inputs of *fixed* length equal the output length of the underlying hash function.) In this section, we solve both issues. We present a block-cipher based construction of a *fixed* input-length *length-preserving* RIO, i.e., a (non-invertible) random permutation oracle (RPO), that only relies on a one-way permutation *without* a trapdoor. In the full version, we show that this assumption is somewhat minimal, as RIOs/RPOs cannot be built from an ideal primitive and a one-way *function*.

Additionally, in order to reduce the dependence between the underlying block- and hash sizes, we present domain/range extenders for RPOs.

4.1 Making Block-Ciphers Non-Invertible: The NIRP-Construction

DESCRIPTION. The NIRP-construction combines a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ and two (fixed-key) ciphers $E_1, E_2 : \{0,1\}^n \to \{0,1\}^n$ in a "sandwich-like" manner. More precisely, for any input $m \in \{0,1\}^n$ the NIRP-constructions is defined such that $\mathsf{NIRP}^{E_1,E_2,\pi}(m) := E_2(\pi(E_1(m)))$. (Also cf. Figure 2.) Obviously, $\mathsf{NIRP}^{E_1,E_2,\pi}$ is a permutation.

SECURITY OF NIRP. We show that the NIRP-construction is indifferentiable from a (non-invertible) random permutation oracle if instantiated with two *ideal* single-key[13] block ciphers $\mathbf{E}_1, \mathbf{E}_2$ and a one-way permutation π (*without* a trapdoor). The result is summarized by the following theorem.

Theorem 2. *Let $\mathbf{E}_1, \mathbf{E}_2 : \{0,1\}^n \to \{0,1\}^n$ be two independent fixed-key ideal ciphers and let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a permutation. There exists a simulator $\mathcal{S}$ (given in the proof) such that for all distinguisher $\mathcal{D}$ issuing at most q queries to the NIRP-construction, and at most q_a, q_b, q_c, q_d queries to $\mathbf{E}_1, \mathbf{E}_1^{-1}, \mathbf{E}_2, \mathbf{E}_2^{-1}$, respectively, there exists an owp-adversary $\mathcal{A}$ with*

$$\mathbf{Adv}^{\text{ind-rio}}_{\mathsf{NIRP}^{\mathbf{E}_1,\mathbf{E}_2,\pi},\mathcal{S}}(\mathcal{D}) \quad \leq \quad 2 \cdot q_c(2q + q_a) \cdot 2^{-n} \quad + \quad q_d \cdot \mathbf{Adv}^{\text{owp}}_{\pi}(\mathcal{A}).$$

The simulator $\mathcal{S}$ runs in time $\mathcal{O}(q_a + q_b + q_c + q_d + (2q_a + q_b + 2q_d) \cdot \text{time}(\pi))$ and makes $q_a + 2q_b + 2q_c$ queries to its oracle, whereas the adversary $\mathcal{A}$ runs in time $\text{time}(\mathcal{A}) \leq \text{time}(\mathcal{D}) + \text{time}(\mathcal{S})$.

OUTLINE OF THE PROOF. The first part of the indifferentiability proof describes the simulator $\mathcal{S}^{\mathbf{P}}$ that mimics the ideal ciphers $\mathbf{E}_1, \mathbf{E}_2$ (with their inverses $\mathbf{E}_1^{-1}, \mathbf{E}_2^{-1}$) given access to a RPO $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$. Moreover we use the notation $\mathcal{S}^{\mathbf{P}} = (\mathcal{S}_{\mathbf{E}_1}, \mathcal{S}_{\mathbf{E}_2}, \mathcal{S}_{\mathbf{E}_1^{-1}}, \mathcal{S}_{\mathbf{E}_2^{-1}})$ to make the four sub-oracles of the simulator (answering the different query types) explicit. The second part (which is postponed to the full version) upper bounds $\mathcal{D}$'s advantage $\mathbf{Adv}^{\text{ind-rio}}_{\mathsf{NIRP}^{\mathbf{E}_1,\mathbf{E}_2,\pi},\mathcal{S}}(\mathcal{D})$ in distinguishing the ideal setting (with a simulator) and the real setting.

[13] Recall that in the ideal cipher model, it is easy to derive two such ciphers from a single ideal cipher $\mathbf{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ as $\mathbf{E}_1 := \mathbf{E}(k_1, \cdot)$ and $\mathbf{E}_2 := \mathbf{E}(k_2, \cdot)$ for two arbitrary distinct keys $k_1 \neq k_2$.

THE SIMULATOR. The global state of the simulator $\mathcal{S}^{\mathbf{P}}$ consists of a table $\mathcal{T}$ (which is initially empty) of tuples of the form (a, b, c, d) consistent with evaluations of the NIRP-construction as in Figure 2, that is, where a, b are simulated input-output values of the first cipher $\mathbf{E}_1$, i.e., $\mathbf{E}_1(a) = b$ (which can be generated both by forward queries to $\mathbf{E}_1$ and by backward queries to $\mathbf{E}_1^{-1}$) and analogously c, d play the same role for the second block cipher $\mathbf{E}_2$. Furthermore, the invariant $c = \pi(b)$ and $\mathbf{P}(a) = d$ holds. It is also convenient to define $A \subseteq \{0,1\}^n$ as the set of values $a \in \{0,1\}^n$ such that $(a, b, c, d) \in \mathcal{T}$ for some b, c, d. Analogously, we define the sets B, C, and D.

To achieve *perfect* simulation given oracle access to $\mathbf{P}$, upon a new query to one of its four sub-oracles the simulator defines a new tuple (a, b, c, d) in $\mathcal{T}$, with the input of the query placed at the appropriate position (as long as no such tuple already exists, in which case the corresponding output value is returned), and such that all remaining components are set to independent random values *conditioned* on these individual values appearing in no other tuple, on $d = \mathbf{P}(a)$, and on $c = \pi(b)$. This is easily achievable with access to π^{-1} and $\mathbf{P}^{-1}$: For example, on input a (to $\mathcal{S}_{\mathbf{E}_1}$), we choose a random $b \xleftarrow{\$} \{0,1\}^n \setminus B$ (i.e., different from all b' appearing in some other tuple), and set $c := \pi(b)$ and $d := \mathbf{P}(a)$. (This is done analogously on input b.) On the other hand, on input c, we compute $b := \pi^{-1}(c)$, a random $a \xleftarrow{\$} \{0,1\}^n \setminus A$ (i.e., different from all previous a'), and then set $d := \mathbf{P}(a)$. Finally, on input d, we set $a := \mathbf{P}^{-1}(d)$ and subsequently generate a random $b \leftarrow \{0,1\}^n \setminus B$ and set $c := \pi(b)$.

However, in our setting we have to dispense with π^{-1} and $\mathbf{P}^{-1}$. In particular, this means that in the latter two cases the simulator cannot set the values b and a, respectively, but rather sets these components to a *dummy value* $\perp$, and completes these tuples with the actual values if they eventually appear as inputs of $\mathbf{E}_1$ or $\mathbf{E}_1^{-1}$ queries. Also note that the simulator must not generate random values a and b that collide with a dummy value in order to ensure the permutation property. This can be efficiently avoided by simply testing that $\mathbf{P}(a) \neq d$ (and $\pi(b) \neq c$) for all d's in tuples of the form $(\perp, b, c, d)$ (all c's in tuples of the form $(a, \perp, c, d)$), and whenever the test fails, we replace the dummy value by the actual value, and draw a new a (or b). There are only two remaining cases where the simulator fails to answer queries (and aborts):

(i) A query a is made and a tuple $(a, \perp, c, d)$ exists: In this case the simulator *must* return $\pi^{-1}(c)$, but this requires inverting π, which is generally not feasible. (Call this event Abort_1.)
(ii) A query b is made and a tuple $(\perp, b, c, d)$ exists: In this case, the simulator *must* return $\mathbf{P}^{-1}(d)$, but cannot invert $\mathbf{P}$. (Call this event Abort_2.)

By the above discussion, perfect simulation is achieved until one of these events occurs: A game-based argument yields $\mathbf{Adv}^{\text{ind-rio}}_{\mathsf{NIRP}^{\mathbf{E}_1,\mathbf{E}_2,\pi},\mathcal{S}}(\mathcal{D}) \leq \mathsf{P}[\mathsf{Abort}_1] + \mathsf{P}[\mathsf{Abort}_2]$. In the full version we give a complete pseudo-code description of the simulator and show that the probabilities of both events are very small.

NIRP = MCM WITH INVERTIBLE MIXING STEPS? Our NIRP-construction somehow reflects the MCM design with a permutation, instead of a hash func-

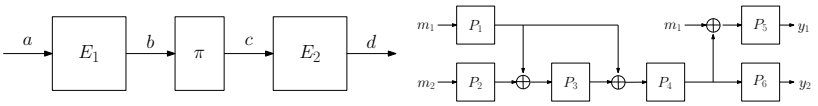

Fig. 2. Left: The NIRP-construction for underlying fixed-key block ciphers E_1, E_2, with (a, b, c, d) corresponding to the notation used in the simulator of Theorem 2. Right: The ESS-construction for underlying permutations $P_1, \ldots, P_6 : \{0,1\}^n \to \{0,1\}^n$.

tion, and this may suggest that the MCM approach works for invertible mixing steps as well. Yet, we remark that the proof cannot be adapted to the case where the first mixing stage processes inputs of variable input-length: The problem is that in the simulation of queries to $\mathbf{E}_2$ and $\mathbf{E}_2^{-1}$ we need to choose a pair $a, \mathbf{P}(a)$ and $b, \pi(b)$ respectively, and at a later time possibly learn the missing dummy values b and c when they are queried. But in order for this to succeed, we need the length of a and b to be compatible with the one of such later query, which is of course impossible in the variable-input-length case.

4.2 Extension of Random Permutation Oracles

The use of the NIRP-construction to post-process the output of a hash function H requires a block cipher with block size at least as large as its hash size, i.e., typically at least 160 bits. While block ciphers with large block size exist,[14] ciphers such as AES support only rather small block lengths, such as 128 bits. This motivates the following natural question: Given a RPO $\mathbf{P} : \{0,1\}^n \to \{0,1\}^n$, can we devise a construction $\mathsf{C}^{\mathbf{P}} : \{0,1\}^m \to \{0,1\}^m$ for $m > n$ which implements a permutation and is indifferentiable from a RPO? Note that this calls for simultaneous domain *and* range extension of $\mathbf{P}$, while we additionally want to ensure injectivity of the resulting construction. The problem is similar in spirit to the one considered in the private-key setting by Halevi and Rogaway [16], even though the peculiarities of the public setting make constructions far more challenging.[15]

The ESS-Construction. We present a construction – called ESS – for the case $m = 2n$ that relies on six *permutations* $P_1, \ldots, P_6 : \{0,1\}^n \to \{0,1\}^n$ and is reminiscent of the compression function $\mathsf{SS}^{P_1,P_2,P_3} : \{0,1\}^{2n} \to \{0,1\}^n$ by Shrimpton and Stam [25] such that $\mathsf{SS}^{P_1,P_2,P_3}(m_1 \| m_2) := P_3(P_1(m_1) \oplus P_2(m_2)) \oplus P_1(m_1)$: It adds three extra calls (as depicted in Figure 2) to ensure both indifferentiability of the $2n$-bit output, as well as invertibility. It is indeed not hard to verify

[14] Interestingly, such block ciphers are exactly the ones used within hash functions, e.g., to instantiate the Davies-Mayer construction.

[15] In particular, each such extender implies the construction of a compression function $\{0,1\}^m \to \{0,1\}^\ell$ for all $\ell < m$ from length-preserving random oracles which is indifferentiable from a random oracle from m bits to ℓ bits, a problem which has recently received much interest (cf. e.g. [20, 25]). On top of this, injectivity is an extra design challenge.

that ESS implements a permutation: Given output $y_1 \| y_2$, the first input-half m_1 is retrieved by computing $z := P_6^{-1}(y_2)$, $m_1 := z \oplus P_5^{-1}(y_1)$, and finally we compute $m_2 := P_2^{-1}(P_1(m_1) \oplus P_3^{-1}(P_1(m_1) \oplus P_4^{-1}(z)))$. (Of course, the inverses P_i^{-1} are not efficiently computable in general, but they are well-defined.)

Indifferentiability of ESS. The following theorem shows that whenever the underlying permutations are independent RPOs, the ESS-construction is indifferentiable from a RPO up to the birthday barrier.

Theorem 3. *Let $\mathbf{P}_1, \ldots, \mathbf{P}_6 : \{0,1\}^n \to \{0,1\}^n$ be independent RPOs. There exists a simulator $\mathcal{S}$ such that for all distinguishers $\mathcal{D}$ making at most q queries to the* ESS*-construction and to each of the underlying RPOs, we have*

$$\mathbf{Adv}^{\text{ind-rio}}_{\mathsf{ESS}^{\mathbf{P}_1,\ldots,\mathbf{P}_6},\mathcal{S}}(\mathcal{D}) \leq \left[q^2 \cdot (4n^2 + n + 28) + q \cdot (3n + 13)\right] \cdot 2^{-n}.$$

The simulator $\mathcal{S}$ runs in time $\mathcal{O}(q^2)$ and makes q queries.

Arbitrary Extension. A generalization of ESS– called MD-ESS – to construct a RPO $\{0,1\}^{i \cdot n} \to \{0,1\}^{i \cdot n}$ for $i > 2$ using $4 + i$ independent RPOs from n bits to n bits and making $4i + 1$ RPO evaluations in total can be obtained as follows: Let $\mathsf{MD\text{-}SS}^{P_1,P_2,P_3} : \{0,1\}^{n \cdot i} \to \{0,1\}^n$ be the (plain) Merkle-Damgård iteration (with no strengthening) that on input $M = m_1 \| \ldots \| m_i$ computes $v_j := \mathsf{SS}^{P_1,P_2,P_3}(v_{j-1} \| m_j)$ for $j = 1, \ldots, i$ (with v_0 being the IV), and outputs v_i. Then, on input $M = m_1 \| \ldots \| m_i \in \{0,1\}^{n \cdot i}$, MD-ESS first computes $y := P_4(\mathsf{MD\text{-}SS}^{P_1,P_2,P_3}(M))$, and finally outputs

$$(P_{4+1}(y) \oplus m_1) \| \cdots \| (P_{4+i-1}(y) \oplus m_{i-1}) \| P_{4+i}(y).$$

To verify that MD-ESS implements a permutation, we remark that its output uniquely determines y and $m_1, \ldots, m_{i-1}$, whereas m_i is determined by the chaining value v_{i-1} and $P_4^{-1}(y)$ as in the ESS-construction. Its security is shown in the full version. There, we also show that $P_{4+1}, \ldots, P_{4+i-1}$ (but not P_{4+i}) can be replaced by (invertible) single-key (ideal) ciphers. Also, it can easily be modified to support inputs with lengths $n' \geq n$ which are not multiples of n.

5 Conclusions

In this paper, we have shown the first modular and fault-tolerant hash function construction which achieves both collision resistance in the standard model and indifferentiability in the ideal model. In particular, this was achieved by building appropriate mixing steps IM and NIRP that are compatible with the MCM-construction and preserve the practical features of the inner compressing part, i.e., the hash function H. By Lemma 1, the construction $\mathsf{MCM}^{\mathsf{IM},H,\mathsf{NIRP}}$ (where possibly NIRP is replaced by its extension through one of the constructions presented in Section 4.2) inherits the collision resistance of H, as IM and NIRP are injective functions. In the ideal setting, we have shown that the combination of IM and H is preimage aware as long as H is sufficiently balanced (Theorem 1),

and that NIRP is indifferentiable from a random permutation oracle (Theorem 2). Thus, by applying Lemma 2, we conclude that $\mathsf{MCM}^{\mathsf{IM},H,\mathsf{NIRP}}$ is indifferentiable from a variable-input-length random oracle.

While the IM-construction is very practical, the implementation of the NIRP-construction, despite its efficiency, is conditioned on the existence of a one-way permutation with input length equal the one of existing block ciphers. Indeed, sufficiently-secure candidate one-to-one functions exist for similar input parameters (e.g., the discrete logarithm problem in properly chosen elliptic curves of prime order $q \approx 2^n$ can in general not be solved better than with running time roughly $\mathcal{O}(2^{n/2})$, i.e., the security of our constructions), but the fact that the block cipher expects n-bit inputs makes their use difficult.[16] However, we stress that such data-type conversion problems are common in practical constructions. For instance, when using an RSA-based trapdoor one-way permutation, the output of the TE-construction [23] must be (injectively) transformed into a string, and the result may be far from being random (attempting to extract random bits would destroy the injectivity property). It is our strong belief that these results should foster further research in designing good candidates for such central cryptographic primitives working at the bit level.

Acknowledgments. We thank Marc Fischlin, Ueli Maurer, Thomas Ristenpart, and the anonymous reviewers for valuable comments. Anja Lehmann is supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). Stefano Tessaro is supported by the Swiss National Science Foundation (SNF), project no. 200020-113700/1. The work described in this paper has been supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

References

1. E. Andreeva, G. Neven, B. Preneel, and T. Shrimpton, "Seven-property-preserving iterated hashing: ROX," in *ASIACRYPT 2007*, vol. 4833 of *LNCS*, pp. 130–146, Springer, 2006.
2. M. Bellare and T. Ristenpart, "Multi-property preserving hash domain extensions and the EMD transform," in *ASIACRYPT 2006*, vol. 4284 of *LNCS*, pp. 299–314, Springer, 2006.
3. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS '93*, ACM Press, 1993.
4. M. Bellare and P. Rogaway, "Optimal asymmetric encryption — how to encrypt with RSA," in *EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 92–111, Springer, 1995.
5. M. Bellare and P. Rogaway, "The exact security of digital signatures — how to sign with RSA and Rabin," in *EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 399–416, Springer, 1996.

[16] A natural candidate operating on n-bit strings arises from exponentiation in the multiplicative group of the extension field $GF(2^n)$: However, due to the existence of non-generic attacks, n has to be chosen large enough, i.e., around the same size as a reasonably secure RSA-modulo.

6. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "On the indifferentiability of the sponge construction," in *EUROCRYPT 2008*, vol. 4965 of *LNCS*, pp. 181–197, Springer, 2008.
7. R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *TCC 2007*, vol. 4392 of *LNCS*, pp. 61–85, Springer, 2007.
8. R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," in *STOC 1998*, pp. 209–218, ACM Press, 1998.
9. S. Contini, A. K. Lenstra, and R. Steinfeld, "VSH, an efficient and provable collision-resistant hash function," in *EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 165–182, Springer, 2006.
10. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgard revisited: How to construct a hash function," in *CRYPTO 2005*, vol. 3621 of *LNCS*, Springer, 2005.
11. I. Damgård, "A design principle for hash functions," in *CRYPTO '89*, vol. 435 of *LNCS*, pp. 416–427, Springer, 1990.
12. Y. Dodis, T. Ristenpart, and T. Shrimpton, "Salvaging merkle-damgård for practical applications," in *EUROCRYPT 2009*, vol. 5479 of *LNCS*, pp. 371–388, Springer, 2009.
13. M. Fischlin and A. Lehmann, "Multi-property preserving combiners for hash functions," in *TCC 2008*, vol. 4948 of *LNCS*, pp. 375–392, Springer, 2008.
14. M. Fischlin, A. Lehmann, and K. Pietrzak, "Robust multi-property combiners for hash functions revisited," in *ICALP 2008*, vol. 5126 of *LNCS*, pp. 655–666, Springer, 2008.
15. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is secure under the rsa assumption," in *CRYPTO 2001*, vol. 2139 of *LNCS*, Springer, 2001.
16. S. Halevi and P. Rogaway, "A tweakable enciphering mode," in *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 482–499, Springer, 2003.
17. NIST SHA-3 Competition. `http://csrc.nist.gov/groups/ST/hash/sha-3/`.
18. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen, "SWIFFT: A modest proposal for FFT hashing," in *FSE 2008*, vol. 5086 of *LNCS*, pp. 54–72, Springer, 2008.
19. U. Maurer, R. Renner, and C. Holenstein, "Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology," in *TCC 2004*, vol. 2951 of *LNCS*, pp. 21–39, Springer, 2004.
20. U. Maurer and S. Tessaro, "Domain extension of public random functions: Beyond the birthday barrier," in *CRYPTO 2007*, vol. 4622 of *LNCS*, pp. 187–204, Springer, 2007.
21. R. Merkle, "One way hash functions and DES," in *CRYPTO '89*, vol. 435 of *LNCS*, pp. 428–446, Springer, 1990.
22. R. Pass, "On deniability in the common reference string and random oracle model," in *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 316–337, Springer, 2003.
23. T. Ristenpart and T. Shrimpton, "How to build a hash function from any collision-resistant function," in *ASIACRYPT 2007*, vol. 4833 of *LNCS*, pp. 147–163, Springer, 2007.
24. P. Rogaway, "Formalizing human ignorance," in *Vietcrypt 2006*, vol. 4341 of *LNCS*, pp. 211–228, Springer, 2006.
25. T. Shrimpton and M. Stam, "Building a collision-resistant compression function from non-compressing primitives," in *ICALP 2008*, vol. 5126 of *LNCS*, pp. 643–654, Springer, 2008.