

Sprinkle the Salt: RO-combiners with $\mathcal{O}(n)$ random bits

Rishiraj Bhattacharyya¹[0000–0002–0407–7780], Mridul Nandi²[0000–0002–1029–6576], and Anik Raychaudhuri³[0009–0006–2890–8522]

¹ University of Birmingham, UK, rishiraj.bhattacharyya@gmail.com

² Indian Statistical Institute, Kolkata, India, mridul.nandi@gmail.com

³ UPES, India, anikrc1@gmail.com

Abstract. At Crypto 2023, Dodis, Ferguson, Goldin, Hall, and Pietrzak [DFGHP23] introduced and constructed Random Oracle (RO) combiners, rejuvenating the well-studied area of cryptographic combiners. RO-combiners are (salted) hash function modes of operation over multiple underlying compression functions that achieve indistinguishability as long as at least one of the underlying compression functions is ideal.

Unfortunately, the security of RO-combiners turns out to be a multi-stage game, and the well-established indistinguishability results for hash function modes cannot be lifted to RO-combiners via the composition theorem. There have been two works aiming to construct RO-combiners from scratch. [DFGHP23] built an indistinguishable compression-function combiner, whose deployment requires a new hash implementation, limiting its scope. The follow-up work by Dodis, Goldin, and Hall [DGH25] at Eurocrypt 2025 constructed the first secure RO-combiner capable of handling a large but fixed-length message using Merkle-Damgård hashing. However, implementing their construction either requires a random salt larger than the message length (potentially exponential in the security parameter) or incurs up to an exponential number of calls to the underlying Merkle-Damgård hash.

In this paper, we introduce a new design principle for robust RO-combiners that yields efficient and *instantly deployable* solutions. At the same time, the principle is generic, enabling the construction of robust variable-input-length RO-combiners from a wide class of popular hashing modes.

Our main result constructs robust RO-combiners from standard *preimage-aware* hash constructions by salting each compression function call and applying a post-processor. The required salt size is $\mathcal{O}(n)$ for $n/2$ -bit security and is independent of the input message size. Moreover, the combiner makes only a single call to each of the underlying hash modes. We show instantiations using both linear Merkle-Damgård and tree hashing.

From an instantiation perspective, with only a simple modification to message padding, our variable-length Merkle-Damgård combiner can be readily obtained with SHA-256 or Blake2. The combiner requires just 1408 bits of randomness while achieving 128-bit security and a message processing rate of 320 bits per compression-function call.

1 Introduction

The Random Oracle heuristic, introduced by Bellare and Rogaway [9] and employed in numerous security proofs, asserts that if a cryptographic protocol using a monolithic Random Oracle $\mathcal{F}$ is secure, then it remains secure when $\mathcal{F}$ is replaced by a *suitable* cryptographic hash function.

Practical cryptographic hash functions, however, are not monolithic objects like a Random Oracle. They are usually built from primitives such as compression functions or permutations, composed via certain modes of operation such as Merkle-Damgård mode [23, 40], Tree mode [41] or Sponge mode [10]. Coron, Dodis, Malinaud, and Puniya [CDMP] [21], building on the framework of indistinguishability of random systems [37], proposed the notion of indistinguishability from a Random Oracle for a *hash function*, where the underlying primitive is considered a monolithic ideal object P (e.g., a fixed-input-length random oracle (FIL-RO) or a random permutation). One then proves that the final hash function H^P , obtained by applying a public hashing mode H to P , has a distribution *indistinguishable* from a Random Oracle. The same approach can then be applied when instantiating P due to composability. The composition theorem of indistinguishability states that if one instantiates P with a construction indistinguishable from an ideal primitive, the resulting hash function H remains indistinguishable from a Random Oracle in any single-stage security game (indistinguishability security game is also a single-stage security game). Starting with the work of CDMP [21], numerous results [5, 11, 12, 14, 22, 27, 28] have established indistinguishability of various hash constructions. Today, indistinguishability from a Random Oracle (also called PRO or pseudorandom oracle) is considered a strongly desirable, if not essential, property of a hash design.

The reader may notice that indistinguishability security for a hash function is very fragile: it requires the underlying primitive to be (indistinguishable from) an ideal primitive. Thus, any reported irregularity—even a minor or seemingly harmless bias—in any underlying primitive invalidates the entire proof (e.g. see [4]), casting doubt on the security of any protocol that employs the hash function.

Random Oracle combiners aim to address this dilemma. A Random Oracle combiner (RO-combiner), considered in [42] and formalized in [24], is a (salted) mode of operation C built from multiple instantiations of the same primitive (e.g., multiple compression functions). The combiner is called robust for PRO if C remains indistinguishable from a Random Oracle as long as *at least one* of the primitive instantiations is indistinguishable from the corresponding ideal object. The idea of an RO-combiner follows a well-established line of research on cryptographic combiners [6, 34, 38], particularly hash function combiners [18, 30, 39, 42, 44, 45], which aim to ensure security as long as one of the underlying components remains secure. Once we have a robust combiner for PRO, the composition theorem allows us to continue securely instantiating ideal objects using indistinguishable combiners. Following this modular approach, one should be able to instantiate a Random Oracle $\mathcal{F}$ using an RO-combiner hash construction based on two or more well-analyzed hash functions, without having

to worry about the effect of every reported bias in a primitive on widely deployed protocols.

But there is a catch. As observed by Dodis, Ferguson, Goldin, Hall, and Pietrzak [24], the robustness-for-PRO security game is not a single-stage game. Thus, the indistinguishability composition theorem does not apply *when constructing an RO-combiner*. In particular, the underlying hash functions may not be independent and could depend on one another. Even if we build a combiner hash from two well-studied hash functions, we cannot conclude robustness-for-PRO solely from the indistinguishability of the individual underlying hash functions. In fact, [DFGHP] [24] demonstrated this limitation through an attack on a concrete instantiation. It appears that a new indistinguishability proof is required from scratch for every combiner hash.

The post-combiner composability, on the other hand, still works. Once we have an RO-combiner, that combiner hash can then be used directly in cryptographic protocols instantiating a Random Oracle in any single-stage security game. To date, only two, albeit influential, works have addressed the design and analysis of RO-combiners from FIL-RO compression functions. In [24], [DFGHP] constructed an $(m+n)$ -bit-to- n -bit combiner compression function by xoring two $(2m+5n/2)$ -bit FIL-ROs using a total of $2m+3n$ bits of salt, achieving indistinguishability up to $2^{n/2}$ queries (birthday-bound security). One can then build a variable-input-length indistinguishable hash function by applying a suitable mode to this robust-for-PRO combiner compression function. Two years later, Dodis, Goldin, and Hall [DGH] [25] reported a major advancement. They showed that the xor of two Merkle-Damgård hashes with n -bit outputs, padded with independent random salts at the end of the messages,

$$\text{XOR-MD}_{Z_1, Z_2}^{f_1, f_2}(M) \stackrel{\text{def}}{=} \text{MD}^{f_1}(M \| Z_1) \oplus \text{MD}^{f_2}(M \| Z_2),$$

is a robust-for-PRO fixed-input-length combiner hash with n -bit output. The compression functions f_1, f_2 may have small, theoretically optimal domain size $(n + \mathcal{O}(1))$, and the random strings must satisfy $|Z_1| = |Z_2| \geq |M| + n/2$.

The problem of RO-combiner, however, is far from being settled. Although formalised as a two-stage game, the RO-combiner formulation in [24, 25] does not allow any function-dependent preprocessing to be done before finalizing the compressor implementations. In other words, the adversarial implementation must be fixed before the FIL-RO compression function is chosen. But standardized hash functions remain public for decades, allowing evaluations on numerous inputs. Any weaknesses found over this long offline period could be encoded in the implementation/distinguisher's algorithm and eventually exploited in the online phase. There has been a series of works [3, 20, 26, 31–33, 48] on the (feasibility of) preprocessing attacks against Random Oracle, salted hash functions and Merkle-Damgård based constructions. Any RO-combiner proposal that aims to support standardized hash functions must show some robustness against preprocessing attacks. Looking ahead, that is not the case with the XOR-MD combiner of [25]. The following natural questions arise.

- **Question 1 - Preprocessing Queries:** How to strengthen the definition of RO-combiners to allow preprocessing queries? Can we construct an efficient RO-combiner that remains robust against such preprocessing attacks?

Minimizing randomness while maintaining efficiency has been a well-recognised foundational requirement in theoretical cryptography. The fixed-input-length combiner proposal from [25] requires the random salts to be as long as the (possibly exponential-size) input message! While one could artificially reduce the salt-size by constructing a small-size combiner, the price would be that the overall number of calls to this combiner and the underlying hash functions grows linearly with the message length, leading to exponentially many calls overall. We know that an RO-combiner must have $\Omega(n)$ -bit salt [24]. Optimizing the randomness-efficiency tradeoff was posed as the main open question in [25].

- **Question 2 - Optimal Randomness-Efficiency Tradeoff:** Is there an RO-combiner hash with n -bit output that supports variable-length inputs, achieves $n/2$ -bit security using only $\mathcal{O}(n)$ random salt bits, and requires only a constant number of calls to the underlying hash function?

Finally, the sequential Merkle-Damgård paradigm is inherently inefficient in parallelizable settings. To fully exploit multi-core processors and SIMD architectures, many applications prefer tree hashing modes. However, no RO-combiner based on tree structures is currently known. Given that the security proof for XOR-MD combiner in [25] works only with Merkle-Damgård hash functions, the following question raised in [24], remains an open problem.

- **Question 3 - Generic Design:** Many practical hash functions employ different underlying structures. Does there exist a generic design of Random Oracle combiner that can be instantiated from general indifferentiable hash functions?

1.1 Our Contributions

We develop a conceptually simple technique to design hash combiners that **positively answers all the above open questions**. Furthermore, the security proofs are based on counting and probabilistic arguments; arguably simpler to verify. We start with formalising PRO-security of RO combiners in the presence of offline queries.

New Definition for Preprocessing. We formalize, in Section 3, a new model of security for random-oracle combiners where the adversarial implementation of a compression function can be fixed after making *significant* albeit bounded number of preprocessing queries to the other, untampered *ideal* compression function. An advice based on the query-answer transcript of this preprocessing phase is hardwired to the adversarial implementation whereas the whole transcript is provided to the online indistinguishability adversary. In the ideal world, the preprocessing queries are made to a first-stage simulator which has no access to the ideal variable-input-length Random Oracle $\mathcal{F}$. The second-stage simulator

which has access to $\mathcal{F}$ also gets the preprocessing phase query-answer transcript as an advice.

We consider four possible models based on the information and power available to the implementation.

1. We start with the **general** model considered in [24, 25], where the adversary *cannot* make any offline query to the ideal function before finalising the adversarial implementation.
2. The strongest definition we consider is the **strong non-uniform** model, where the implementor can make an exponential number of offline queries. The adversarial implementation is also allowed to make queries in the online phase.
3. The middle of the spectrum is a **succinct non-uniform model** where the implementation *does not get any access* to the ideal compression function in the online phase. The number of offline queries remains potentially exponential in the security parameter.
4. The weakest version is the **weak non-uniform model**, where the number of offline queries are *polynomially bounded*, and the implementation *does not get any access* to the ideal compression function in the online phase. We note that the weak model was conceived in [24], and finding an efficient combiner in this weak model was left as an open question.

How do our three new models compare with the general model? Our strong model is surely stronger than the general model, as the adversary gets the additional power of making a huge number of offline queries. The succinct model appears to be somewhat orthogonal to the general model. Indeed, the adversarial implementation can now work with some structures found through the huge number of offline query-responses, but it loses the ability to utilize any online, adaptive, salt-dependent query-responses. However, the offline query-response transcript provides the second-stage indistinguishability distinguisher with additional, useful information like a set of colliding inputs to the compression function. We establish this by demonstrating an attack against the XOR-MD combiner of [25] by making $2^{n/2}$ preprocessing queries and 2 online queries in the succinct model. The details are in section 4. In contrast, our main *generic* construction remains robust in the strong non-uniform model, secure up to 2^k many preprocessing queries using a k -bit random salt.

We note that our query-response transcript advice to the online adversary and the second stage simulator is in the same line as many recent works on preprocessing attacks and similar two-stage indistinguishability notions [32, 33, 46]. As the first-stage simulator has no access to the Random Oracle $\mathcal{F}$, the post-combiner indistinguishability composition theorem holds true in our stronger model as well (we present a composition theorem in Theorem 12). The caveat of the large space requirements of the reduction is not considered a barrier in the information-theoretic setting.

Envelope-then-XOR: Generic construction of RO-combiners from Various Hashing Modes. We present *generic* constructions of *Variable-Input-Length* RO-combiners that are secure in the *strong non-uniform* model, require $\mathcal{O}(n)$ salt and make a *single* call to each of the underlying hashes from FIL-

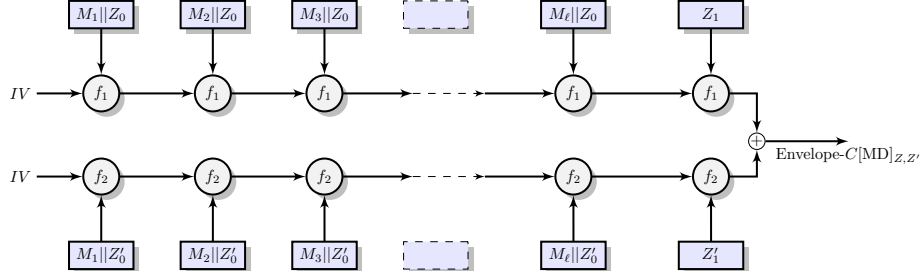


Fig. 1: Envelope- $C[MD]$: RO-combiner from $\mathcal{O}(n)$ salt and two calls to Merkle-Damgård Hash. For $\frac{n}{2}$ -bit security, $|Z_0| \geq \frac{3n}{4}$, $|Z_1| = m + |Z_0|$ where m is the rate, and f_1, f_2 are $(m + \frac{7n}{4})$ -bit-to- n -bit compression functions.

RO based *preimage-aware* hash functions. Our design principle embodies an *envelope-then-XOR* approach, which allows us to extend the characterization of traditional single-stage indistinguishability security⁴, with literature spanning works over two decades, to this "imperfect" primitive setting. Our constructions utilize two random salts, one short and one long. The short ones are padded in *every* compression call of the preimage-aware hash H , whereas the long ones are applied to an additional compressor evaluation on the output of H to create a VIL-RO via the "independent post-processor" framework [8, 28]. The following theorem establishes that the XOR of two such VIL-RO is indeed a secure RO-combiner.

Theorem 1 (Informal). *Let $f_1, f_2: \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$ be two compression functions. Let $Z_0, Z'_0 \xleftarrow{\$} \{0, 1\}^k$ and $Z_1, Z'_1 \xleftarrow{\$} \{0, 1\}^{m+k}$ and $Z = (Z_0, Z_1)$, $Z' = (Z'_0, Z'_1)$. Let H be a Preimage Aware hash function that employs a $(n+m)$ -to- n -bit hash function. The Envelope- $C[H]$ combiner, defined as follows, is a PRO-secure combiner. For $M \in \{0, 1\}^*$,*

$$\text{Envelope-}C[H]_{Z,Z'}^{f_1,f_2}(M) \stackrel{\text{def}}{=} f_1(H^{f_{(1,Z_0)}}(M), Z_1) \oplus f_2(H^{f_{(2,Z'_0)}}(M), Z'_1)$$

where $f_{(1,Z_0)} := f_1(\cdot, \cdot, Z_0)$ and $f_{(2,Z'_0)} := f_2(\cdot, \cdot, Z'_0)$.

As a consequence of the above theorem, we can now construct RO-combiners from a wide range of hashing modes, used across different applications. One notable example is the *first* VIL-RO-combiner from the popular tree mode mode of hashing. Our combiner can be readily implemented via modes supporting compression function salting (eg. tweaks in Skein, keyed-mode in Blake3 etc).

The main highlight of our work is a *Variable-Input-Length RO-combiner* from $\mathcal{O}(n)$ salt and only *two* calls to the *classical Merkle-Damgård hash* and a *simple message padding rule* (depicted in Figure 1).

⁴ Preimage-awareness is a weaker notion than indistinguishability. A Random Oracle is preimage-aware. Hence, any indistinguishable from RO hash is also preimage-aware.

XOR-MD combiner via Brined-HAIFA Padding of Messages. Consider the Brined-HAIFA padding of messages that works in the following way. Let $Z = (Z_0, Z_1)$ be random salt such that $Z_0 \xleftarrow{\$} \{0, 1\}^k$ and $Z_1 \xleftarrow{\$} \{0, 1\}^{m+k}$. Let $M = M_1 \| M_2 \| \dots \| M_\ell$ where $|M_i| = m$. The (m, k) -Brine padding of M with salt Z is defined by

$$\text{Brine}(M, Z) = (M_1 \| Z_0) \| (M_2 \| Z_0) \| \dots \| (M_\ell \| Z_0) \| Z_1$$

A learned reader may have observed, our Brine padding, carefully curated for the combiner design, has some resemblance with a plausible *counterless* salt-only version of the HAIFA padding [17]. However, generic HAIFA padding, as sketched in [17] and used in practice, does not imply combiner security.

Theorem 2 ((informal) PRO security of XOR combiner from Merkle-Damgård hash with Brine padding). *Let $f_1, f_2: \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$ be two compression functions. Let $Z_0, Z'_0 \xleftarrow{\$} \{0, 1\}^k$ and $Z_1, Z'_1 \xleftarrow{\$} \{0, 1\}^{m+k}$ and $Z = (Z_0, Z_1)$, $Z' = (Z'_0, Z'_1)$. The Envelope-C[MD] combiner, defined as follows, is a PRO-secure combiner. For $M \in \{0, 1\}^*$,*

$$\text{Envelope-C[MD]} \stackrel{\text{def}}{=} MD^{f_1}(\text{Brine}(M, Z)) \oplus MD^{f_2}(\text{Brine}(M, Z'))$$

When $k = \frac{3n}{4}$, we prove the above Envelope-C[MD] combiner achieves $\frac{n}{2}$ -bit security. Thus, the total randomness requirement is $3n + 2m$ for an $(m + \frac{7n}{4})$ -bit-to- n bit compression function. In popular MD hashes like SHA256 and Blake2s, the underlying compression functions are $3n$ -to- n bit compression functions with $n = 256$. Instantiating with SHA256 or Blake2s, the above combiner requires a total 1408 random bits with a message processing rate of 320 bits per function call.

Our envelope-then-XOR RO-combiners remain robust even in the strong model. Their efficiency and security of hold true when one of the compression functions is adversarially implemented after making up to exponential in the randomness parameter $\mathcal{O}(2^k)$ many offline queries to the other untampered compression function (Theorem 14).

Rate preserving RO-combiners. Our final contribution is to build RO-combiners that has message processing rate almost same as the underlying hash functions while still using $\mathcal{O}(n)$ salt. We present two scenarios where our Envelope-then-XOR combiner preserve the message processing rate. In both the settings, our construction works by applying the random salt only in the post-processing (envelope) block and without needing to salt the primitive calls of the underlying preimage-aware hash.

In the first design in Section 7, we employ a large-domain independent post-processor as the envelope. This combiner construction is proven secure in the general model of [24, 25], without considering any offline preprocessing queries.

In the second construction in Section 7.3, we use all identical compression functions as in our main strong model construction. However, we prove the security in a weak model (conceptualized in [24]) where the implementor can only make polynomially many offline queries before producing the adversarial implementation, *and* the adversarial implementation itself *does not get any additional*

access to the ideal compression function. We show that our envelope-then-xor combiner achieves PRO security by applying the random salt only in the last (envelope) block.

1.2 Technical Overview

Our main technical contribution is in developing a technique that *extends* established characterizations of indifferentiable hash functions to prove the security of RO-combiners. We focus our discussion in the general model from [24, 25] as the security for our constructions in the strong non-uniform model is achieved via a reduction from the general model.

The Indifferentiability Framework. We start by recalling the indifferentiability framework for hash functions [21]. In this framework, a distinguisher $\mathcal{A}$ is interacting with one of two worlds; the REAL world or the IDEAL world. In the REAL world, $\mathcal{A}$ has oracle access to a hash construction C which is built on some primitive P . The design of the construction is public, and $\mathcal{A}$ has also oracle access to P . In the IDEAL world, the hash construction is replaced by an ideal Variable Input Length Random Oracle (VIL-RO) $\mathcal{F}$. The role of P is performed by a simulator sim . The objective of the simulator is to simulate the working of P while maintaining consistency with $\mathcal{F}$ so that the distinguisher $\mathcal{A}$ cannot distinguish between the two worlds.

General theme of indifferentiability proofs: Uniqueness of Final Primitive Query. Indifferentiability security proofs for hash modes often pivot on proving the “uniqueness of final primitive query”. As long as the final query input for a hash chain is “fresh” and unique, the simulator can “program” the output in tune with the ideal VIL-RO function $\mathcal{F}$. Such an approach gets automatically carried out for the combiner $H^{f_1} \oplus H^{f_2}$ if f_1 and f_2 are two *independent* functions.

However, when $\mathcal{G}^f$, one of the two compression functions, is arbitrarily dependent on the other function f , such a uniqueness property cannot be argued in general. Indeed, for the xor combiner $\text{MD}^f(M\|Z) \oplus \text{MD}^{\mathcal{G}^f}(M\|Z')$, as DGH [25] showed, an $\text{MD}^{\mathcal{G}^f}(M', Z')$ may compute some $\text{MD}^f(M, Z)$. As a result, the simulator needs to go into a recursion to remain consistent with $\mathcal{F}$. DGH [25] required Z, Z' to be longer than M, M' to ensure the recursion terminates.

We develop a conceptually different approach. The central theme of our technique is to design a combiner where this uniqueness of the final primitive query of a hash chain holds, even when f_1 and f_2 are *not independent*. At first glance, this may appear impossible. But as we show, with the proper use of “random salts”, the condition is not only feasible, but also achievable for *practical* hashes, with a very little loss in efficiency. The salts are sampled after the combiner C and the dependent compression function $\mathcal{G}$ have been fixed, and our technique will infuse a functional independence between the two underlying hash components, $\mathcal{G}^f$ and f via the randomness of the salt.

Envelope Design of Hash Functions and their Indifferentiability. We start by recalling the “envelope design” approach for constructing indifferentiable

hash functions [8, 28]. We are given two FIL-ROs f, R , where f is an $(n + m)$ -bit-to- n -bit compression function, and R is an n -bit-to- n -bit function. The hash design comprises applying R as an independent post-processor on the output of an n -bit hashing mode H^f .

$$\text{Envelope-}H^{f,R}(M) = R(H^f(M))$$

The indifferentiability proof involves the notion of Preimage-Awareness [28]. Roughly speaking, preimage-awareness guarantees that an adversary's commitment to a hash output is binding to a unique, extractable preimage. In other words, the adversary can produce at most a single preimage of any hash output, and that preimage too is extractable from the f queries made until the commitment.

Indifferentiability of Envelope- H follows from the preimage-awareness of the hashing mode H using the uniqueness of the final primitive (in this case, the FIL-RO R) query paradigm. The simulator would simulate the function f perfectly honestly via lazy sampling: on a new query, it would return a response sampled uniformly at random. When an $R(y)$ query comes, the simulator will extract an M from the f -query response transcript such that $H^f(M) = y$, and set the output of the $R(Y)$ query to be equal to $\mathcal{F}(M)$. If no such M can be extracted, the simulator will return a uniform random response.

The above simulator perfectly simulates the working of f and R while maintaining the required consistency with the VIL-RO output $\mathcal{F}$ in all but two situations. The first one occurs if for two distinct M, M' we get $H^f(M) = y = H^f(M')$: the simulator cannot respond to the $R(y)$ query while simultaneously being consistent with $\mathcal{F}(M)$ as well as $\mathcal{F}(M')$. The second one occurs when no M could be extracted from the f -query-response transcript such that $H^f(M) = y$, but the adversary later produces a concrete preimage M : in this case, the simulator responded with a random output rather than $\mathcal{F}(M)$. However, both of these two conditions are exactly the two winning conditions of preimage-awareness. [28] argued that if an adversary can make this simulator inconsistent, they can break the preimage-awareness of H^f . In other words, until the adversary can break the preimage-awareness of H^f , the input to the final primitive R remains unique and thus the output can be programmed to maintain consistency with $\mathcal{F}$.

$\mathcal{G}^f$ -influence and Characterising Combiner Indifferentiability. Now consider the following XOR combiner with two FIL-ROs (f, R) and the corresponding adversarial implementations $(\mathcal{G}_f, \mathcal{G}_R)$ along with a preimage-aware hash function H . The combiner computes

$$\text{Post-C}[H]_{Z, Z'}^{f, R, \mathcal{G}_f, \mathcal{G}_R}(M) = R(H^f(M), Z) \oplus \mathcal{G}_R(H^{\mathcal{G}_f}(M), Z')$$

Note, both $\mathcal{G}_f$ and $\mathcal{G}_R$ have oracle access to (f, R) . Let us try to build a simulator for the Post-C[H] combiner following the idea described above. The simulator would simulate the function f perfectly honestly by returning a response sampled uniformly at random. When an $R(y, Z)$ query comes, the simulator will similarly extract an M from the f -query response transcript such that $H^f(M) = y$. The

next task is to compute $t = \mathcal{G}_R(H^{\mathcal{G}_f}(M), Z')$ so that the $R(y)$ query response can be set to $t \oplus \mathcal{F}(M)$.

Here comes the complexity arising from the dependent functions. In order to compute t , the simulator needs to respond to the (f, R) queries made by $\mathcal{G}_R$ and $\mathcal{G}_f$. Observe, the simulator's response to these queries made by $\mathcal{G}_R/\mathcal{G}_f$ needs to be exactly same as if the queries were made by the indistinguishability distinguisher. As a result, if the simulator needs to compute $\mathcal{G}_R(H^{\mathcal{G}_f}(M'), Z')$ for some M' when answering the R queries made by $\mathcal{G}_R/\mathcal{G}_f$, the simulator goes into a recursion!

Our main technical abstraction is in the precise characterization of this recursion condition that we call $\mathcal{G}^f$ -influence. If either of the adversarial implementations $\mathcal{G}_R/\mathcal{G}_f$ makes an $R(y', Z)$ query for which the simulator needs to compute $\mathcal{G}_R(H^{\mathcal{G}_f}(M'), Z')$, then it must hold that $H^f(M') = y$. Furthermore, the situation is arising when simulator is computing $\text{Envelope-}H_{Z'}^{\mathcal{G}_f, \mathcal{G}_R}(M) = \mathcal{G}_R(H^{\mathcal{G}_f}(M), Z')$. Taking everything together, we find that the adversary is able to find (not necessarily distinct) M, M' such that during the computations of $\text{Envelope-}H_{Z'}^{\mathcal{G}_f, \mathcal{G}_R}(M)$ and $\text{Envelope-}H_Z^{f, R}(M') = R(H^f(M'), Z)$, the same $R(y, Z)$ query was made. In Figure 6, we formalise this task into the $\mathcal{G}^f$ -influence game.

Suppose this $\mathcal{G}^f$ -influence condition does not happen. Then the simulator knows that the R queries made by $\mathcal{G}_R/\mathcal{G}_f$ will not be on an input (y, Z) such that $R(y, Z)$ becomes the final query of some $\text{Envelope-}H_Z^{f, R}(M')$ computation. In other words, conditioned on $\mathcal{G}^f$ -influence not happening, the simulator does not need to query $\mathcal{F}$ for any of the R queries made by $\mathcal{G}_R/\mathcal{G}_f$.

We get a simple simulator algorithm that *never* goes into a recursion. For queries made either directly by the adversary, or the adversarial implementations $\mathcal{G}_R/\mathcal{G}_f$, the simulator simulates the function f perfectly honestly by returning a response sampled uniformly at random. Similarly, for R queries made by $\mathcal{G}_R/\mathcal{G}_f$, the simulator returns a uniform random response. Only when the adversary submits a $R(y, Z)$ query with the correct salt Z , the simulator extracts an M from the f -query response transcript such that $H^f(M) = y$, computes $t = \mathcal{G}_R(H^{\mathcal{G}_f}(M), Z')$, and returns $\mathcal{F}(M) \oplus t$. If no such M can be extracted, a uniform random response is returned. For all other R queries, a uniform random response is returned.

This simple simulator is sufficient for the PRO-security of our combiner. In Theorem 13, we show that an indistinguishability distinguisher against the $\text{Post-C}[H]$ combiner either breaks the preimage-awareness or wins the $\mathcal{G}^f$ -influence game.

Bounding $\mathcal{G}^f$ -influence: Merkle-Damgård with large postprocessor.

How do we bound the probability of $\mathcal{G}^f$ -influence event? For a warm-up, let us start with the Merkle-Damgård hash mode with a large postprocessor. Suppose, we have the Merkle-Damgård hash based on a $(n + m)$ -bit-to- n -bit compression function f . Let R be a $(\frac{5n}{2} + m)$ -to- n -bit compression function, independent from f . Consider the salted hash $H_Z^{f, R}(M) = R(\text{MD}^f(M), Z)$, where Z is a

$(\frac{3n}{2} + m)$ -bit salt. The XOR combiner (Figure 2) with (f, R) and adversarial implementations $(\mathcal{G}_f, \mathcal{G}_R)$ then computes

$$\text{Post-C}[\text{MD}]_{Z, Z'}^{f, R, \mathcal{G}_f, \mathcal{G}_R}(M) = R(\text{MD}^f(M), Z) \oplus \mathcal{G}_R(\text{MD}^{\mathcal{G}_f}(M), Z')$$

We argue that neither any $\mathcal{G}_f$ nor any $\mathcal{G}_R$ computation can query $R(\cdot, Z)$ with

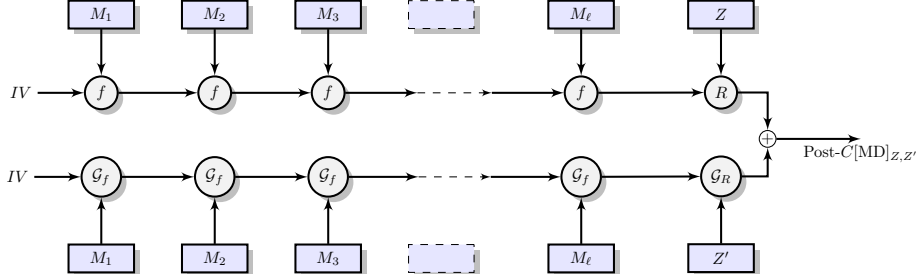


Fig. 2: Generic Post-C[MD] combiner

any significant probability during any $\mathcal{G}_R(\text{MD}^{\mathcal{G}_f}(M), Z')$ evaluation. All we need is counting. The input-size of $\mathcal{G}_f$ is $(n + m)$ bits, and for each of these $(n + m)$ -bit inputs, $\mathcal{G}_f$ makes at most $q_{\mathcal{G}}$ many queries. So, for any function pair (f, R) , across all the inputs, there can be at most $q_{\mathcal{G}} \times 2^{(n+m)}$ many ways $\mathcal{G}_f$ can query R ; implying there can be at most $q_{\mathcal{G}} \times 2^{(n+m)}$ many choices of $\hat{Z}$ such that $R(\cdot, \hat{Z})$ appears in this query table. The description of $\mathcal{G}$ was fixed before independently sampling the functions (f, R) and the salt Z . The probability that the sampled salt Z appears in a R query in the $\mathcal{G}_f$ query table for any fixed (and thus for independently and uniformly sampled) (f, R) is $\frac{q_{\mathcal{G}} 2^{(n+m)}}{2^{(n+m+n/2)}} = \frac{q_{\mathcal{G}}}{2^{n/2}}$. As the non-salt input of $\mathcal{G}_R$ is only n -bit, the probability in that case is even less. So, with probability at least $(1 - \frac{q_{\mathcal{G}}}{2^{n/2}})$, the sampled functions (f, R) and the sampled salt Z are such that the final query input of a $H_Z^{f, R}$ hash evaluation is not influenced by *any* $H_{Z'}^{\mathcal{G}_f, \mathcal{G}_R}$ computation. We get $\frac{n}{2}$ -bit PRO security via the existing proof of preimage-awareness of MD^f .

The Challenge: Merkle-Damgård with single compression function. We turn our attention to our Envelope- $C[\text{MD}]$ combiner from Figure 1 where the Merkle-Damgård mode is instantiated only with a single compression function. Suppose one of the chain is instantiated with the $(n + m + k)$ -bit-to- n -bit FIL-RO compression f , and the other uses an adversarially implemented compression function $\mathcal{G}^f$.

Our Brined-HAIFA padding applies a fixed salt Z_0 in each message block. As a result, the non-salt input size of f and $\mathcal{G}^f$ becomes restricted. This idea creates a similarity with the large postprocessor (with salt Z_1) case analyzed above, which immediately gives us $n/2$ -bit security with $|Z_0| = k = \frac{3n}{2}$ and

$|Z_1| = \frac{3n}{2} + m$. Unfortunately, these parameters are not acceptable as the message processing rate is only m -bits per call while employing a $(5n/2 + m)$ -to- n -bit compression function f . For $3n$ -to- n functions like in SHA256/SHA512, the message processing rate m is only $\frac{n}{2}$.

Our main technical quest is to make the parameters practical by improving the rate. It turns out the practical parameters can be achieved via careful probability analysis. Recall our Brined-Haifa Merkle-Damgård hash padding where the last block is a random salt Z_1 . If a $\mathcal{G}^f$ evaluation has to influence a $\text{MD}_Z^f(M)$ hash output, then it has to query $f(h, Z_1)$ for $h = \text{MD}^{f(\cdot, Z_0)}(M)$. Such a $\mathcal{G}^f$ evaluation either (a) *computes* an $h = \text{MD}^{f(\cdot, Z_0)}(M)$ itself before querying $f(h, Z_1)$, or (b) it “works out” a valid (h, Z_1) pair from the input and its f queries. In the first case, that particular $\mathcal{G}^f$ evaluation queries both $f(\cdot, Z_0)$ and $f(\cdot, Z_1)$. The probability of this event is easy to bound using entropy/counting arguments. The second case requires more careful handling. We write a rough idea below.

If, as in the second case, a $\mathcal{G}^f$ evaluation does not query $f(\cdot, Z_0)$ before querying $f(h, Z_1)$, the distribution of such chaining value h and the FIL-RO function $f(\cdot, Z_0)$ are independent. So, if we make a list L of the chaining values y that appear in such $f(y, Z_1)$ queries in the full $\mathcal{G}^f$ query table, the probability that any $f(\cdot, Z_0)$ query (made by the adversary or $\mathcal{G}^f$) output lands in that list L is at most $\frac{|L|}{2^n}$. Taking union bound over all σ many $f(\cdot, Z_0)$ queries in the PRO game transcript, the probability of a non-fresh input in the final primitive query of any $H_Z^f(M)$ evaluation is at most $\frac{\sigma|L|}{2^n}$.

Bounding $|L|$. The remaining step is to bound the size of L . A straightforward counting argument finds that with probability at least $\frac{q_{\mathcal{G}}}{2^{k-n/2}}$, the size of L is at most $2^{\frac{n}{2}}$. The resulting parameter of $k = n$ for $\frac{n}{2}$ -bit security, however, would have been sub-optimal. By a fine grained analysis on the distribution of $|L|$, we achieve $n/2$ -bit security with $k = \frac{3n}{4}$ leading to total randomness of $|Z_0| + |Z_1| = m + \frac{3n}{2}$ bits, and the domain size of the compression function f as $m + \frac{7n}{4}$ bits.

As a result of our analysis, for $3n$ -to- n functions like SHA256, the message processing rate of our Envelope-C[H] combiner becomes $\frac{5n}{4}$. In Section 1.3, we compare the parameters with standard designs in detail.

Handling preprocessing queries. Now we move to the strong non-uniform model where the adversary can make offline queries. The idea to salt every compressor query provides a natural robustness to such preprocessing attacks. If none of the offline queries match with the “valid” inputs of the ideal hash chain, then the offline query transcript has no effect on the security analysis presented thus far. The probability that the offline query transcript contains a query on $f(\cdot, Z_0)$ or $f(\cdot, Z_1)$ is at most $\frac{2q_1}{2^k}$. Hence, a reduction from the general model (without offline queries) can simulate the offline query-response transcript by sampling at random, and the games remain indistinguishable for up to 2^k offline queries.

1.3 Instantiation Efficiency.

How efficient are our combiners when instantiated with the compression functions from popular hash functions? Modern compression functions (compressors) in MD based modes often reserve additional space beyond the message-blocks and state that could be used for the salting. For example, in case of modes using HAIFA padding [17], the salt Z_0 can replace the counters and other flags. For an exact comparison with practical hash functions, consider the generic specifications of MD hashing in SHA256/512 [1], BLAKE (SHA3 submission [7]), BLAKE2 [47], BLAKE3 [43] with n -bit-state compressor and maximum input message length $2^{n/4}$ bits. Our Envelope-C[MD] combiner's message-processing rate is

- equal to that of any plausible XOR-BLAKE-mode combiner (HAIFA $2n$ -bit message block, $n/4$ -bit counter, $n/2$ -bit additional tags).
- $3/4$ times the rate of any plausible XOR-BLAKE2-mode combiner (HAIFA $2n$ -bit message block, $n/4$ -bit counter).
- $5/8$ times the rate of any plausible XOR-SHA256-mode or XOR-SHA512-mode combiner ($2n$ -bit message block, no additional bits for counter/tags).

For example, consider the compressor used in SHA512, $f: \{0, 1\}^{1536} \rightarrow \{0, 1\}^{512}$. It is a $3n$ -to- n bit compression function with $n = 512$. Our salted Envelope-C[MD] XOR combiner (Figure 1) requires total randomness of 1408 bits per hash function, setting $|Z_0| = 384$ bits and $|Z_1| = 1024$ bits. Each compression function call thus processes $\frac{5 \times 512}{4} = 640$ bits of message data, compared to 1024 bits of standardized specification.

Similarly, for the tree mode, instantiated with the corresponding compression functions, our XOR-Envelope-Tree combiner's message processing rate is equal to that of BLAKE3 (keyed) tree hash mode. Our combiner can be implemented via a larger compression function (using $3n$ -to- n -bit compression function in a binary tree). In that case, the efficiency loss factor is at most $1/3$ compared to a ternary tree hash.

Rate comparison with [25]. Our variable-input-length mode using $3n$ -to- n -bit compressor and $2.75n$ -bit random salt has message processing rate approximately $5n/4$ -bits per call per hash . The fixed-input-length RO-combiner from [25] with *salt-size more than the message-length* has rate less than n bit per call per hash. *Directly turning it into a variable length scheme will cause further loss of inefficiency for shorter messages.* Restricted to $2.75n$ -bit salt, a derived variable-input-length combiner conceived in [25] would have rate less than $n/2$ bits per call per hash *while making potentially exponentially many calls to the underlying hash function.* We calculate the rate as follows. $2.75n$ bit salt implies that each call of the XOR-MD combiner can handle up to $2.25n$ bit message. Among these, only $1.25n$ bit can be used for a new message-block because n bit needs to be reserved for the previous combiner call result. Given that the total salted message size for each XOR-MD call is $5n$ bits, the combiner must make 3 calls to the $3n$ -to- n compressor. Hence 3 compression-function calls are required for processing $1.25n$ message bits, leading to a rate of less than $n/2$ bits per compression call.

1.4 Other Related Works.

Combiners for cryptographic applications were first studied in [35,36]. Hash combiners focusing on collision resistance have been studied extensively in [18,19,45]. In particular, it is known that a collision-resistant combiner implies an output size at least twice that of the underlying hash functions. In [30], multi-property preserving combiners were investigated, including a version of indistinguishability. However, the model considered there is much weaker than the ones considered in this paper.

Mittelbach [42] introduced the notion of a Cryptopia combiner (later revised in [39]), which overcomes the lower bound on the output length of a collision-resistant combiner by assuming that one of the underlying hash functions is ideal. Although the paper envisioned that the input hash functions of the combiner could be replaced by constructions indistinguishable from a hash function, [24] demonstrated that this is not the case. Furthermore, when instantiated with real-world hash functions, the number of black-box calls the Cryptopia combiner needs to make to the underlying hash function is proportional to the message length, which limits its efficiency.

Our approach of query bounded preprocessing phase for two-stage indistinguishability has been considered in [15,16,32,46]. The work of [32] did not consider adversarial implementation, and naturally treatments and techniques therein are inadequate for combiner security. The works of [15,16,46] looked at subversion resilient hashing and their model is significantly stronger. It is unclear whether any practical hash design can achieve birthday-bound security in their setting.

2 Notations and Preliminaries

Notations. If S is a discrete set, $|S|$ denotes the size of the set. We write $x \xleftarrow{\$} S$ to denote the process of choosing x uniformly at random from a set S and independently from all other random variables defined so far. Let $\{0,1\}^*$ be the set of all binary strings. For a positive integer n , the term $\{0,1\}^n$ denotes the set of all n -bit binary strings. For a positive integer l , we use $[l]$ and $[l]$ to denote the sets $\{0,1,\dots,l\}$ and $\{1,\dots,l\}$ respectively. If x and y are two strings, $x||y$ denotes the concatenated string. $F_{m,n}$ denotes the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$. $f : \mathcal{D}_f \times \{0,1\}^k \rightarrow \mathcal{R}_f$ denotes a family of 2^k many functions from $\mathcal{D}_f$ to $\mathcal{R}_f$. We often use the shorthand f_z to denote the function family $\{f_z := f(\cdot, z)\}_{z \in \{0,1\}^k}$, particularly when the function family is given as oracles. We often use the notation $f(*, z)$ to denote an f -evaluation where the last element of the input is z .

Adversaries and Transcripts. An algorithm $\mathcal{A}$ having access to oracles $\mathcal{O}_1, \dots, \mathcal{O}_k$ is called a $(q_1, \dots, q_k)$ -query algorithm if for all $i \in [k]$, $\mathcal{A}$ makes at most q_i many queries to its i^{th} oracle. The random coin of $\mathcal{A}$ is independent of the random coins of the oracles. The adversaries considered in this paper are computationally unbounded algorithms. The complexities of these algorithms are measured solely on query complexity, the number of queries they make. Adversarial queries and the corresponding responses are stored by a tuple of pairs τ ,

called a transcript. We also denote $\tau_{\mathcal{O}}$ to denote the part of the transcript containing all query responses to the oracle $\mathcal{O}$. We call an oracle $\mathcal{O}$ *primitive* if the joint distribution of the responses does not depend on the order of the queries. In this case, we write the transcript as a multi-set $\{(x_1, y_1), \dots, (x_k, y_k)\}$. For a fixed transcript $\tau = ((x_1, y_1), \dots, (x_k, y_k))$, the event that the oracle $\mathcal{O}$ returns y_i on the i th query x_i , for all i , is denoted as $\tau \leftarrow \mathcal{O}$. *The randomness of the event only comes from the oracle $\mathcal{O}$ and not from the algorithm $\mathcal{A}$ to which it interacts.*

Definition 3 (Distinguishing Advantage). Let F^k and G^k be two k -tuples of randomized oracle algorithms for some positive integer k . We define the advantage of an adversary $\mathcal{A}$ at distinguishing F^k from G^k as

$$\Delta_{\mathcal{A}}(F^k ; G^k) \stackrel{\text{def}}{=} |\Pr[\mathcal{A}^{F_1, F_2, \dots, F_k} = 1] - \Pr[\mathcal{A}^{G_1, G_2, \dots, G_k} = 1]|.$$

Indifferentiability. An oracle construction $C^P(\cdot, \cdot)$ with a random salt, first fixes the salt Z (chosen randomly from a set, called salt space). Afterwards, on input M , the construction C interacts with the oracle $\mathcal{O}$, and finally, it returns an output, denoted as $C_Z^{\mathcal{O}}(M)$ or $C^{\mathcal{O}}(M, Z)$. We can similarly define an oracle construction with more than one oracle $\mathcal{O}_1, \dots, \mathcal{O}_k$. The domain of M is also called the domain of the construction. An oracle construction C is called $\mathcal{F}$ -compatible if the domains and ranges of C and $\mathcal{F}$ (an ideal primitive) are the same. Now we recall the definition of indifferentiability of an oracle construction [21, 37] in our terminology.

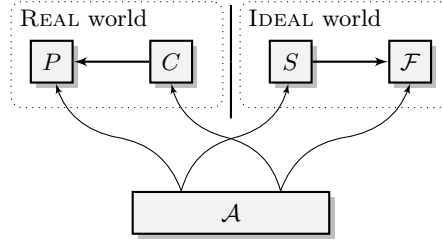


Fig. 3: The indifferentiability security notion.

Definition 4 (Indifferentiability). Let $\mathcal{F}$ be an ideal primitive and C^P be an $\mathcal{F}$ -compatible oracle construction. C is said to be $((q_P, q_C, q_{\text{sim}}), \epsilon)$ -indifferentiable from an ideal primitive $\mathcal{F}$ if there exists a q_{sim} -query algorithm $S^{\mathcal{F}}$ (called simulator) such that for any (q_P, q_C) -query algorithm $\mathcal{A}$, it holds that

$$\text{Adv}_{C, \mathcal{F}, S}^{\text{pro}}(\mathcal{A}) := \Delta_{\mathcal{A}(Z)}((P, C^P(\cdot, Z)) ; (S^{\mathcal{F}}(Z, \cdot), \mathcal{F})) < \epsilon.$$

where Z is the random salt of the construction C , chosen uniformly from the salt space and provided to the adversary $\mathcal{A}$ and the simulator S . When the oracle P is replaced by multiple oracles, the simulator S simulates all of them.

When $\mathcal{F}$ is a random oracle, we call $\mathcal{A}$ PRO (pseudo random oracle) adversary and the construction is called PRO (pseudo random oracle) secure.

COMPLETE DISTINGUISHER. We call $\mathcal{A}$ *complete*, if it makes all necessary P queries to compute $C^P(M_i, Z)$ for all M_i queried to the construction oracle. In case it is not complete, we assume that $\mathcal{A}$ makes all remaining P queries after it is over with its usual interaction. This would not reduce the advantage, and it would only cost some additional P queries. In this paper, without loss of generality, we assume all PRO adversaries are complete.

Hash Function and Preimage Awareness

Definition 5 (Hash Modes). A n -bit hash mode is an oracle algorithm H which, on an input $M \in \{0, 1\}^*$ (and a possibly random seed Z) and with access of a primitive oracle P , returns an n -bit hash value h . We write the hash output as $H^P(M) = h$ or $H_Z^P(M) = h$ in case of a presence of a seed Z .

A hash mode H^P is called *simple*, if the hash output h matches the output of some P query made by H . This is in the same spirit as the GDE mode of [13].

PREIMAGE AWARENESS. In [28], authors introduced the preimage awareness game and the notion of preimage aware extractor. Given a $h \in \{0, 1\}^n$, a transcript τ , and an optional random seed Z the extractor algorithm Ex returns $M \in \{0, 1\}^* \cup \{\perp\}$. We write this as $\text{Ex}(h, \tau, Z) = M$.

An extractor is called *correct* for a hash-mode H_Z^P if for all Z, h and $\tau \leftarrow P$, it holds that

$$\text{Ex}(h, \tau, Z) = M \neq \perp \implies H_Z^P(M) = h$$

Throughout the rest of the paper, we only consider correct extractors.

PRA ADVANTAGE. Let Z be a random salt. The PrA game $\text{PrA}[\mathcal{A}(Z), P, \text{Ex}(Z)]$ is defined as follows:

1. $\mathcal{A}$ interacts with P and an extractor oracle Ex . The queries to the extractor oracle are called commitment.
2. Throughout the interaction, $\mathcal{A}$ commits $h_1, \dots, h_t$, and receives $M_1, \dots, M_t$, where $M_i = \text{Ex}(h_i, \tau_i, Z)$ where τ_i is the transcript of interaction between $\mathcal{A}$ and P until commitment h_i .
3. Finally $\mathcal{A}$ returns a message M .

We say that $\mathcal{A}$ wins **PrA game** for a n -bit hash mode H_Z^P against an extractor Ex if it holds that 1. for some i , $H_Z^P(M) = h_i$ and 2. $M \neq M_i$. We denote the probability that $\mathcal{A}$ wins the $\text{PrA}[\mathcal{A}(Z), P, \text{Ex}(Z)]$ game as $\text{Adv}_{H, \text{Ex}}^{\text{pra}}(\mathcal{A})$.

Envelope Hash Modes and Indifferentiability

Definition 6 (Envelope Hash Mode). Let H^P be a n -bit (possibly salted via randomness Z) hash mode based on a primitive P . Let $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a function independent from P . We define the Envelope- H hash as the composition of H^P and R , i.e. $\text{Envelope-H}_Z^{P,R}(M) := R(H_Z^P(M))$.

The following theorem was proved in [28]. If H^P is Preimage Aware, and R is an independent FIL-RO, then the Envelope- $H^{P,R}$ hash is a PRO.

Theorem 7 (PRO security of Envelope Hash [28]). Let H^P be a Preimage Aware (PRA) hash mode with a correct extractor Ex , and $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$, $\mathcal{F} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be random oracles. There exists sim such that for any PRO adversary $\mathcal{A}$, there exists a PRA adversary $\mathcal{B}$ such that

$$\text{Adv}_{\text{Envelope-H}, \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \text{Adv}_{H, P, \text{Ex}}^{\text{pra}}(\mathcal{B})$$

An Envelope- H hash can be instantiated using a single FIL-RO primitive f via domain separation (e.g. fixing the last bit to be 0 for the internal function and setting the last bit to be 1 for the envelope function).

Other Useful Lemmas

Lemma 8 (Markov Inequality (Theorem 1.6.4 in [29])). Let X be a non-negative random variable and $a > 0$ be a real number. Then it holds that $\Pr[X \geq a] \leq \frac{\mathbb{E}(X)}{a}$.

3 Hash Combiners and PRO Security

In this section we recall the definition of hash combiner and present our definitions of PRO security for hash combiners.

Definition 9 (Hash Combiner). A P -primitive hash-combiner $C_Z^{f_1, f_2} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ is an algorithm keyed by randomness Z , with oracle access to two candidate P -primitives f_1 and f_2 , that implements a n -bit hash mode.

We call a hash combiner $C_Z^{f_1, f_2}$ symmetric if for all $M \in \{0, 1\}^*$, we have $C_Z^{f_1, f_2}(M) = C_Z^{f_2, f_1}(M)$.

Implementer. A (q_1, q_G) -implementer is a q_1 -query oracle algorithm $\mathcal{A}_1^\mathcal{O}$ with the transcript of oracle query-responses denoted by $\tilde{\tau}$. $\mathcal{A}_1$ outputs the description of the adversarial implementation $\mathcal{G}^\mathcal{O}$ (denoted by $\langle \mathcal{G} \rangle$), and a state $\text{st} = \tilde{\tau}$. The algorithm $\mathcal{G}^\mathcal{O}$ makes at most q_G many queries to its oracle. A transcript-dependent advice $\alpha(\tilde{\tau})$ is hardwired in $\mathcal{G}$.

Definition 10 (Combiner Distinguisher). We say that a pair $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms $((q_1, q_G), (q_P, q_C))$ -combiner distinguisher for a combiner $C_Z^{f, \mathcal{G}^f}$ if

1. $\mathcal{A}_1$ is a (q_1, q_G) -implementer for f ;
2. $\mathcal{A}_2(\langle \mathcal{G} \rangle, \mathbf{st}, Z)$ is a (q_P, q_C) -query indistinguishability distinguisher where Z is the random salt of the combiner construction, provided as input.

We revisit the definition of PRO security of a symmetric hash combiner, as defined in [24], to include offline queries. The definition is based on the following two-stage distinguishing game. At the beginning, the ideal primitives f and $\mathcal{F}$ are sampled. The (q_1, q_G) -implementer $\mathcal{A}_1$ with oracle access to f produces the subverted implementation $\mathcal{G}^f$ along with a state $\mathbf{st}$. After this preprocessing phase, a random salt Z is sampled and published as the salt of the construction C . Finally, $\mathcal{A}_2$ is invoked with the description of the implementation $\langle \mathcal{G} \rangle$, the state $\mathbf{st}$, and the random salt Z as input. In the real world, $\mathcal{A}_2$ interacts with the f (same from the first stage) and the construction $C_Z^{f, \mathcal{G}^f}(\cdot)$. In the ideal world, the oracle f is simulated by a two stage simulator $\mathbf{sim} = (\mathbf{sim}_1, \mathbf{sim}_2)$. The first stage simulator $\mathbf{sim}_1$ has *no* oracle access to $\mathcal{F}$. The second stage simulator $\mathbf{sim}_2$ gets the transcript $\tilde{r}$, the random salt Z as inputs, along with oracle access to the adversarial implementation $\mathcal{G}$, and the random oracle $\mathcal{F}$. The simulator is aimed to simulate f so that the behaviour of $(f, C_Z^{f, \mathcal{G}^f})$ is as close as $(\mathbf{sim}_2, \mathcal{F})$ to the distinguisher $\mathcal{A}_2$.

Definition 11 (PRO Security of Symmetric Hash Combiner). Let $C_Z^{f_1, f_2}$ be a P -primitive hash-combiner. Let f be a P -primitive and $\mathcal{F}$ be a random oracle. We say that C is $((q_1, q_G), (q_P, q_C), q_{\text{sim}}, \varepsilon)$ random oracle secure if there exists a q_{sim} -query simulator $\mathbf{sim}_2$ with access to $\langle \mathcal{G} \rangle$ such that for any $((q_1, q_G), (q_P, q_C))$ -combiner distinguisher adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that

$$\text{Adv}_{C, \mathcal{F}, \mathbf{sim}_2}^{\text{pro}}(\mathcal{A}_2) \leq \varepsilon$$

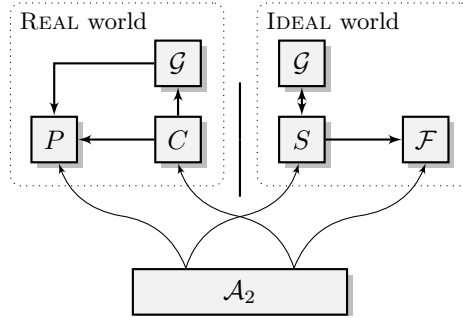


Fig. 4: PRO security of Hash Combiners

For the rest of the paper, we view $f_1 = f$, $f_2 = \mathcal{G}^f$ and present a hash combiner C as a hash mode $C_{Z, Z'}^{f, \mathcal{G}^f}$. In this paper, we consider four models of combiner distinguisher as listed below.

1. The **general** model, studied in [24, 25] has $q_1 = 0$ and $q_G > 0$. For the rest of the paper, the general model is embodied by a $(q_G, (q_P, q_C))$ -combiner distinguisher.
2. the **strong** non-uniform model has $q_1 > 0$, and $q_G > 0$.
3. the **succinct** non-uniform model has $q_1 > 0$, and $q_G = 0$.
4. the **weak** non-uniform model has $q_1 = \mathcal{O}(\text{poly}(n))$, and $q_G = 0$.

In order to establish the validity of our models, we include the following composition theorem. The theorem and the proof is focused on the strong non-uniform model, but can be easily adapted to any of the other models as well.

Theorem 12 (Composition Theorem for a strong model RO-Combiner).

Let $\mathcal{P}$ be a cryptosystem with oracle access to a VIL-RO $\mathcal{F}$. Let f be an FIL-RO and C be an $\mathcal{F}$ -compatible RO-combiner such that C is $((q_1, q_G), (q_P, q_C), q_{sim}, \varepsilon)$ random oracle secure. Then the cryptosystem $\mathcal{P}^C$ in the f -model is as secure as $\mathcal{P}^{\mathcal{F}}$ in the $\mathcal{F}$ model. For every (q_1, σ) -query adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a σq_{sim} query adversary $\mathcal{B}$ such that for any security game G ,

$$\left| \Pr \left[(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{A}_1^f; Z \xleftarrow{\$} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G \left(P^{C_Z^{f, \mathcal{G}^f}}, \mathcal{A}_2^f(Z, \tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \right) \right] \right. \\ \left. - \Pr \left[(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{B}; Z \xleftarrow{\$} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G \left(P^{\mathcal{F}}, \mathcal{B}^{\mathcal{F}}(Z, \tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \right) \right] \right| \leq \varepsilon$$

where $\sigma \leq q_P + q_G q_C \ell_{max}$

Proof. We construct the adversary $\mathcal{B}$ in the $\mathcal{F}$ model. Let $\text{sim} = (\text{sim}_1, \text{sim}_2)$ be the simulator from the strong model PRO security of C . $\mathcal{B}$ runs $\mathcal{A}_1$: answering queries via sim_1 and stores the output $(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle)$. When $\mathcal{B}$ gets invoked in game G with input Z along with its state, $\mathcal{B}$ invokes $\mathcal{A}_2$ on $(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle)$. The queries of f oracle is handled by sim_2 , which is executed by $\mathcal{B}$ itself with the stored $\tilde{\tau}$ and $\langle \mathcal{G} \rangle$ and its own access to oracle $\mathcal{F}$. Note, query complexity of $\mathcal{B}$ is bounded by the query complexity of the simulator times the number queries to the simulator.

We now construct a strong model adversary $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ against the RO-Combiner C . $\mathcal{D}_1$ has oracle access to $\mathcal{O}_f$, $\mathcal{D}_2$ has oracle access to $(\mathcal{O}_f, \mathcal{O}_C)$. $\mathcal{D}_1$ runs $\mathcal{A}_1$: answers all the queries via its oracle, and outputs $(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle)$ committing $\langle \mathcal{G} \rangle$. The distinguisher $\mathcal{D}_2$ receives input Z on invocation. $\mathcal{D}_2$ runs game G ; when the cryptosystem P makes an oracle query, it is answered via $\mathcal{O}_C$. When G calls $\mathcal{A}_2$, $\mathcal{D}_2$ invokes $\mathcal{A}_2$ and answers $\mathcal{A}_2$'s queries by its own oracles. Finally, when the game G outputs a bit b , $\mathcal{D}_2$ outputs the same b .

When $(\mathcal{O}_f, \mathcal{O}_C) = (f, C_Z)$, $\mathcal{D}$ exactly simulates the game

$$\left[(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{A}_1^f; Z \xleftarrow{\$} \{0, 1\}^{\lambda(n)}; G \left(P^{C_Z^{f, \mathcal{G}^f}}, \mathcal{A}_2^{f, P}(Z, \tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \right) \right]$$

In other words

$$\Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_Z^{f, \mathcal{G}^f}} \right) = 1 \right] = \\ \Pr \left[(\tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{A}_1^f; Z \xleftarrow{\$} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G \left(P^{C_Z^{f, \mathcal{G}^f}}, \mathcal{A}_2^{f, P}(Z, \tilde{\tau}, \text{st}, \langle \mathcal{G} \rangle) \right) \right]$$

When $(\mathcal{O}_f, \mathcal{O}_C) = (\text{sim}, \mathcal{F})$, the adversary $\mathcal{D}$ exactly simulates the game

$$\left[(\tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{B}; Z \stackrel{s}{\leftarrow} \{0, 1\}^{\lambda(n)}; G(P^{\mathcal{F}}, \mathcal{B}^{\mathcal{F}}(Z, \tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle)) \right]$$

Hence,

$$\begin{aligned} & \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2, \mathcal{F}} \right) = 1 \right] = \\ & \Pr \left[(\tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{B}; Z \stackrel{s}{\leftarrow} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G(P^{\mathcal{F}}, \mathcal{B}^{\mathcal{F}}(Z, \tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle)) \right] \end{aligned}$$

Thus

$$\begin{aligned} & \left| \Pr \left[(\tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{A}_1^f; Z \stackrel{s}{\leftarrow} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G \left(P^{C_Z^{f, \mathcal{G}^f}}, \mathcal{A}_2^{f, P}(Z, \tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle) \right) \right] \right. \\ & \quad \left. - \Pr \left[(\tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle) \leftarrow \mathcal{B}; Z \stackrel{s}{\leftarrow} \{0, 1\}^{\lambda(n)}; 1 \leftarrow G(P^{\mathcal{F}}, \mathcal{B}^{\mathcal{F}}(Z, \tilde{\tau}, \mathbf{st}, \langle \mathcal{G} \rangle)) \right] \right| \\ & = \left| \Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_Z^{f, \mathcal{G}^f}} \right) = 1 \right] - \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2, \mathcal{F}} \right) = 1 \right] \right| \\ & \leq \varepsilon \end{aligned}$$

□

4 XOR-MD from [25] is insecure against large preprocessing queries

Claim. XOR-MD is insecure in the strong/succinct non-uniform model for $2^{n/2}$ preprocessing queries.

Proof. The main idea of the attack is to exploit the deterministic first block (with $IV = 0^n$, say) used in the construction. A collision found in the offline stage with IV as the chaining value, can be used to mount collision attack on the hash functions, leading to distinguishing attack on the Combiner. Recall the XOR-MD combiner from [25]

$$\text{XOR-MD}_{Z, Z'}^{f_1, f_2}(M) \stackrel{\text{def}}{=} \text{MD}^{f_1}(M \| Z) \oplus \text{MD}^{f_2}(M \| Z'),$$

Let $f: \{0, 1\}^{n+m} \rightarrow \{0, 1\}^n$ be an FIL-RO compression function. For ease of explanation, we assume $m > \frac{n}{2}$. The attack works for any $m > 0$ with simple adjustment.

In the preprocessing stage $\mathcal{A}_1$

- makes $2^{n/2}$ queries to get two $\frac{n}{2}$ -bit messages x_1, x_2 such that

$$f(IV, x_1 \| 0^{m-\frac{n}{2}}) = f(IV, x_2 \| 0^{m-\frac{n}{2}}).$$

$\mathcal{A}_1$ aborts if no such pair exists. By birthday bound, $\mathcal{A}_1$ succeeds in finding such pair with probability $\frac{1}{2}$.

- creates an adversarial implementation $\mathcal{G}^f: \{0,1\}^{n+m} \rightarrow \{0,1\}^n$ that works as follows. On input 1^{n+m} , $\mathcal{G}^f$ outputs $x_1||x_2$, and for all other input, $\mathcal{G}^f$ outputs 0^n

The PRO distinguisher $\mathcal{A}_2$, on input $\langle \mathcal{G} \rangle$ and the salts Z, Z' , evaluates $\mathcal{G}(1^{n+m})$ to extract $x_1||x_2$, and sets $M_1 = x_1||0^{m-\frac{n}{2}}$ and $M_2 = x_2||0^{m-\frac{n}{2}}$. $\mathcal{A}_2$ queries the oracle $\mathcal{O}_C$ on inputs M_1 and M_2 and outputs 0 if $\mathcal{O}_C(M_1) = \mathcal{O}_C(M_2)$, outputs 1 otherwise.

As $f(IV, M_1) = f(IV, M_2)$, we have

$$\text{MD}^f(M_1||Z_1) = \text{MD}^f(M_2||Z_1)$$

On the other hand, as $IV = 0^n$, $\mathcal{G}^f(IV, M_1) = \mathcal{G}^f(IV, M_2) = 0^n$. Once the chaining value becomes 0^n , every subsequent $\mathcal{G}^f$ evaluation in the $\text{MD}^{\mathcal{G}^f}$ computations output 0^n .

$$\text{MD}^{\mathcal{G}^f}(M_1||Z') = \text{MD}^{\mathcal{G}^f}(M_2||Z')$$

We conclude, $C_{Z,Z'}^{f,\mathcal{G}^f}(M_1) = C_{Z,Z'}^{f,\mathcal{G}^f}(M_2)$. However, $\mathcal{F}(M_1)$ will be same as $\mathcal{F}(M_2)$ only with negligible probability. $\square$

One interesting property of the above attack is that the second stage adversary (distinguisher) $\mathcal{A}_2$ does not need any additional advice from $\mathcal{A}_1$ beyond the implementation $\langle \mathcal{G} \rangle$. Moreover, adversarial implementation $\mathcal{G}^f$ does not make any query to f .

5 Envelope-then-XOR Combiner and PRO Security

5.1 The Envelope-then-XOR Combiner.

Primitives. Let $n, m, k > 0$ be the security parameter, message rate, and a randomness parameter, respectively. The primitives we consider here are a FIL-RO $f: \{0,1\}^{n+m+k} \rightarrow \{0,1\}^n$ and a corresponding adversarially implemented function $\mathcal{G}^f$, where each invocation of $\mathcal{G}^f$ can make $q_{\mathcal{G}}$ queries to f .

Random Salt. The basic hash mode uses one $(m+k)$ -bit random string $Z_1 \xleftarrow{\$} \{0,1\}^{m+k}$ and one k -bit random string $Z_0 \xleftarrow{\$} \{0,1\}^k$. Throughout, we assume Z_0 is different from the last k -bits of Z_1 . Looking ahead, we shall consider the event (Z_0 is same as the last k -bits of Z_1) as Bad. Clearly, the probability of this bad event is 2^{-k} , which is negligible for a reasonable choice of k .

Combiner. We define the compression functions $f_{Z_0}: \{0,1\}^{n+m} \rightarrow \{0,1\}^n$ and $f_{Z_1}: \{0,1\}^n \rightarrow \{0,1\}^n$ from f via domain separation, by fixing the last bits of the f inputs to Z_0 and Z_1 respectively:

$$f_{Z_0} := f(\cdot, \cdot, Z_0) \quad f_{Z_1} := f(\cdot, Z_1).$$

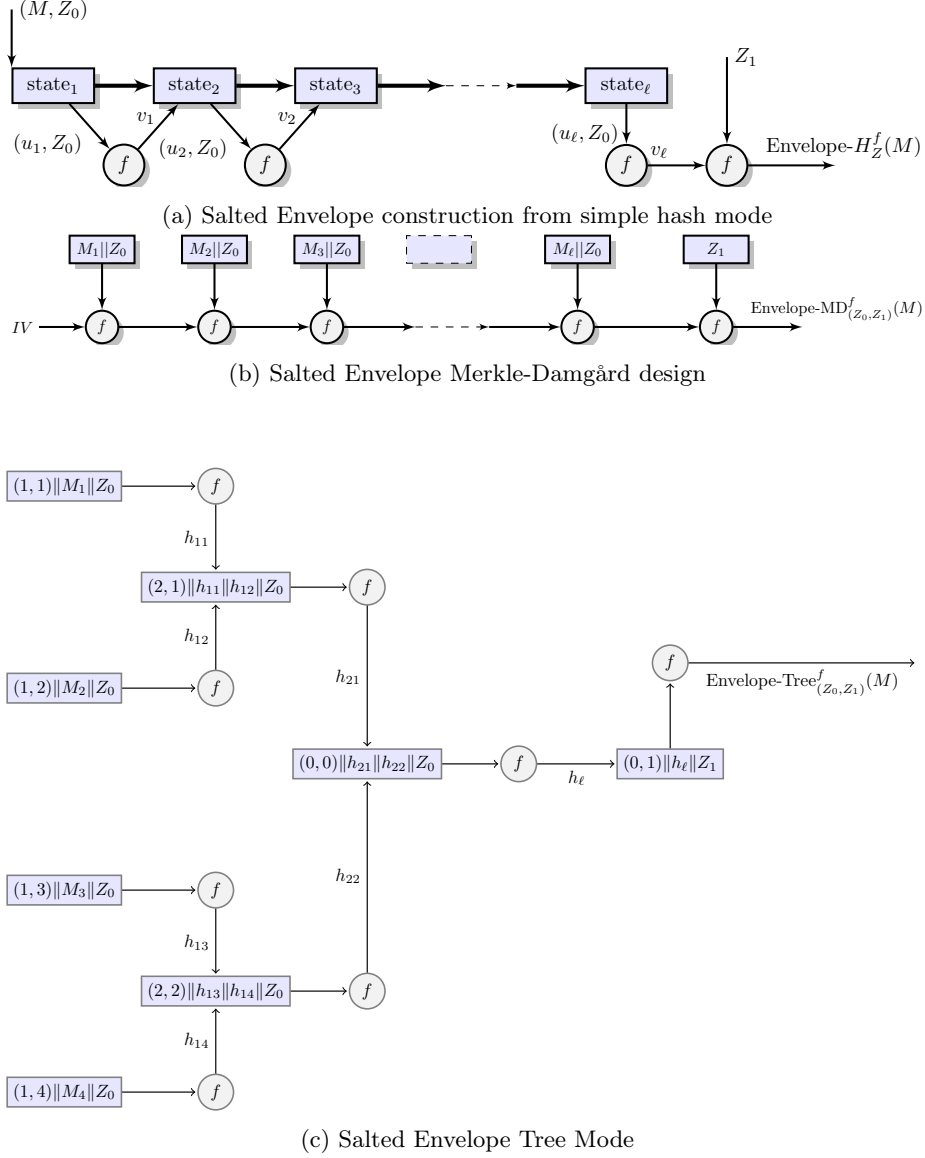


Fig. 5: The Salted Envelope- $H_{(Z_0, Z_1)}^f$ designs. Figure 5a shows the Envelope construction from a simple hash mode. Figure 5b shows the Merkle-Damgård instantiation, Figure 5c shows the Tree mode instantiation.

If f is a random function and Z_0 is different from the last k -bits of Z_1 , then the restricted functions f_{Z_0} and f_{Z_1} are independent (as the domains are separated). Similarly, we define the salted adversarial implementations

$$\mathcal{G}^f_{Z'_0} := \mathcal{G}^f(\cdot, \cdot, Z'_0) \quad \mathcal{G}^f_{Z'_1} := \mathcal{G}^f(\cdot, \cdot, Z'_1)$$

Let $H^{\hat{f}}: \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a *preimage-aware simple* hashing mode using a FIL-RO compression function $\hat{f}: \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^n$. We define the envelope- H mode with salt $Z = (Z_0, Z_1)$ as the composition of $H^{f_{Z_0}}$ and f_{Z_1} , i.e.

$$\text{Envelope-}H_Z^f(M) := f_{Z_1}(H^{f_{Z_0}}(M)) = f(H^{f(\cdot, \cdot, Z_0)}(M), Z_1)$$

The evaluation through the adversarial implementation will similarly be

$$\text{Envelope-}H_{Z'}^{\mathcal{G}^f}(M) := \mathcal{G}^f_{Z'_1}(H^{\mathcal{G}^f_{Z'_0}}(M)) = \mathcal{G}^f(H^{\mathcal{G}^f(\cdot, \cdot, Z'_0)}(M), Z'_1)$$

Examples of some salted Envelope- $H_{(Z_0, Z_1)}^f$ designs can be found in Figure 5. The Envelope- $C[H]$ xor combiner with salts $Z = (Z_0, Z_1)$ and $Z' = (Z'_0, Z'_1)$ is the xor of two Envelope- H evaluations, one through f and the other through $\mathcal{G}$.

$$\text{Envelope-}C[H]_{Z, Z'}^{f, \mathcal{G}}(M) = \text{Envelope-}H_Z^f(M) \oplus \text{Envelope-}H_{Z'}^{\mathcal{G}^f}(M)$$

Overall randomness requirement of the Envelope-then-XOR combiner is

$$|Z| + |Z'| = 2(|Z_0| + |Z_1|) = 2(k + m + k) = 4k + 2m \quad (1)$$

Game $\mathcal{G}^f$ -influence($\langle \mathcal{G} \rangle$, Envelope- $C[H]$)	Event Bad-Salt
$f \xleftarrow{\$} F_{(n+m+k), n}$	$f \xleftarrow{\$} F_{(n+m+k), n}$
$Z_0, Z'_0 \xleftarrow{\$} \{0, 1\}^k; Z_1, Z'_1 \xleftarrow{\$} \{0, 1\}^{m+k}$	$(\langle \mathcal{G} \rangle, \tilde{\tau}) \leftarrow \mathcal{A}_1^f$
$Z = (Z_0, Z_1); Z' = (Z'_0, Z'_1)$	$Z_0, Z'_0 \xleftarrow{\$} \{0, 1\}^k$
$(M, M') \leftarrow \mathcal{A}^f(Z, Z')$	$Z_1, Z'_1 \xleftarrow{\$} \{0, 1\}^{m+k}$
$Q_G = \{h \mid \text{Envelope-}H_{Z'}^{\mathcal{G}^f}(M') \text{ queries } f(h, Z_1)\}$	$Z = (Z_0, Z_1); Z' = (Z'_0, Z'_1)$
$Q_H = \{h \mid \text{Envelope-}H_Z^f(M) \text{ queries } f(h, Z_1)\}$	if Offline Transcript $\tilde{\tau}$ contains query
if $Q_G \cap Q_H \neq \emptyset$ return 1	$f(*, *, Z_0)$ or $f(*, *, Z_1)$
else return 0	Bad-Salt = 1
	else Bad-Salt = 0

Fig. 6: The $\mathcal{G}^f$ -influence game and the Bad-Salt event.

5.2 $\mathcal{G}^f$ -influence game and PRO security of Envelope Combiner

In this section, we introduce the security game $\mathcal{G}^f$ -influence. An adversary wins the $\mathcal{G}^f$ -influence game if they can produce two messages M, M' such that during the $\text{Envelope-H}_{Z'}^{\mathcal{G}^f}(M')$ evaluation, the $\mathcal{G}^f$ queries the last chaining block of $\text{Envelope-H}_Z^f(M)$ (i.e. the envelope query to $f(*, Z_1)$).

Formally, the game is defined in Figure 6. The $\mathcal{G}^f$ -influence advantage of an adversary $\mathcal{A}$ is defined as

$$\text{Adv}_{\mathcal{G}^f, C}^{\text{influence}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr [\mathcal{G}^f\text{-influence}[C]^{\mathcal{A}} = 1]$$

We establish the PRO security of our envelope combiner via two theorems.

PRO security of Envelope Combiner in the General Model. The first theorem, which establishes PRO security in the *general* model (where $q_1 = 0, q_G > 0$), says that unless a $\mathcal{G}^f$ evaluation queries the last chaining block of some $\text{Envelope-H}_Z^f(M)$, the preimage awareness of H is sufficient to establish the indistinguishability of our envelope-then-XOR combiner against a *general* combiner-distinguisher.

Theorem 13 (PRO security of Envelope Combiner in general model). *Let H^f be a Preimage Aware (PRA) hash mode with extractor Ex and maximum message length ℓ . Let $\text{Envelope-C}[H]$ be the Envelope- H xor combiner. There exists a simulator sim such that for any $(q_G, (q_P, q_C))$ -combiner distinguisher adversary $\mathcal{A}$, there exists a PRA adversary $\mathcal{B}$ and a $\mathcal{G}^f$ -influence adversary $\mathcal{C}$ such that*

$$\text{Adv}_{\text{Envelope-C}[H], \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \text{Adv}_{H, P, \text{Ex}}^{\text{pra}}(\mathcal{B}) + \text{Adv}_{\mathcal{G}^f, \text{Envelope-C}[H]}^{\text{influence}}(\mathcal{C}) + \frac{1}{2^k}$$

where $\mathcal{B}$ and $\mathcal{C}$ makes at most $2(q_P q_G(\ell + 1) + q_C q_G(\ell + 1)^2)$ many queries to their oracles and $q_{\text{sim}} = q_P$.

We refer the reader to Section 6.1 for the proof.

PRO security of Envelope Combiner in the Strong Non-Uniform Model

The next theorem establishes the PRO security of Envelope-then-XOR combiner in the strong non-uniform model. Before stating the theorem, we introduce the event Bad-Salt (See Figure 6). Recall that the strong model adversary makes q_1 many queries to the function f in the offline phase before finalising the adversarial implementation $\mathcal{G}^f$. The random salts Z_0, Z'_0, Z_1, Z'_1 are sample independently thereafter at the start of the online mode. We say the Bad-Salt event happens if the offline query transcript $\tilde{\tau}$ contains any query on $f(*, *, Z_0)$ or $f(*, Z_1)$. Given that $Z_0 \stackrel{\$}{\leftarrow} \{0, 1\}^k$ and $Z_1 \stackrel{\$}{\leftarrow} \{0, 1\}^{m+k}$ are independent from $\tilde{\tau}$, we conclude,

$$\Pr [\text{Bad-Salt}] \leq \frac{2|\tilde{\tau}|}{2^k} = \frac{2q_1}{2^k}$$

The following theorem argues that unless the event Bad-Salt happens, PRO security of our Envelope-then-XOR combiner in the general model *implies* the PRO security in the strong non-uniform model.

Theorem 14 (PRO security of Envelope Combiner in the strong model).

Let $\text{Envelope-C}[H]$ be the Envelope- H xor combiner. There exists a two stage simulator $\text{sim} = (\text{sim}_1, \text{sim}_2)$ such that for any strong model $((q_1, q_G), (q_P, q_C))$ -combiner distinguisher adversary $\mathcal{D}$ and any simulator S , there exists a general model $(q_G, (q_P, q_C))$ -combiner distinguisher adversary $\mathcal{A}$ such that

$$\text{Adv}_{\text{Envelope-C}[H], \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{D}) \leq \text{Adv}_{\text{Envelope-C}[H], \mathcal{F}, S}^{\text{pro}}(\mathcal{A}) + \frac{2q_1}{2^k}$$

We include the proof in Section 6.3.

Bounding $\mathcal{G}^f$ -influence advantage of Envelope- $C[H]$ combiners. The following theorem summarises the $\mathcal{G}^f$ -influence security of the Envelope- $C[H]$ xor combiner for a simple hash mode H .

Theorem 15. Let $Z_0, Z'_0 \in \{0, 1\}^k$ and $Z_1, Z'_1 \in \{0, 1\}^{m+k}$. Let $H^{\hat{f}}$ be a n -bit simple hashing mode using a $(n+m)$ -to- n -bit FIL-RO compression function $\hat{f}$. Let f be a $(n+m+k)$ -to- n -bit FIL-RO compression function with the adversarial implementation $\mathcal{G}^f$. Consider the envelope- H mode with salt $Z = (Z_0, Z_1)$ and the Envelope- $C[H]$ xor combiner. It holds that, for any adversary $\mathcal{C}$ making at most $\sigma = 2(q_P q_G(\ell + 1) + q_C q_G(\ell + 1)^2)$ blocks of queries

$$\text{Adv}_{\mathcal{G}^f, \text{Envelope-C}[H]}^{\text{influence}}(\mathcal{C}) \leq \frac{q_G^2}{2^{2k-n}} + \frac{2\sigma^2}{2^n} + \frac{2(n+1)q_G\sigma}{2^k}$$

We prove the above theorem in Section 6.2.

Corollary of Theorem 15: PRO security of Envelope-then-XOR combiners based on Popular Modes We immediately get the following theorems by applying Theorems 13 to 15, and PRA security of Merkle-Damgård mode [28].

Theorem 16 (PRO security of Envelope Merkle-Damgård Combiner).

Let f be an $(n+k+m)$ -bit-to- n -bit compression function. Let MD^f be the Merkle-Damgård hash mode with maximum allowed message length ℓ . Let $\text{Envelope-C}[MD]$ be the salted Envelope-MD xor combiner with total salt of $2(2k+m)$ bits. Then for any $((q_1 q_G), (q_P, q_C))$ -strong non-uniform model combiner distinguisher adversary $\mathcal{A}$, it holds that

$$\text{Adv}_{\text{Envelope-C}[MD], \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \mathcal{O} \left(\frac{q_G^2}{2^{2k-n}} + \frac{\sigma^2}{2^n} + \frac{(n+1)q_G\sigma}{2^k} + \frac{q_1}{2^k} \right)$$

where $\sigma = 2(q_P q_G(\ell + 1) + q_C q_G(\ell + 1)^2)$ and $q_{\text{sim}} = q_P$.

For tree hashing, we apply Theorems 13 to 15 along with Theorem 1 of [22] and PRA security of indifferentiable constructions [28].

Theorem 17 (PRO security of Envelope Tree Combiner). *Let f be an $(tn+k+2\log \ell)$ -bit-to- n -bit compression function for $t \geq 2$. Let T^f be an indifferentiable secure t -ary tree hash mode and maximum allowed message length ℓ . Let Envelope-C[T] be the Envelope-Tree xor combiner with total salt of $2((t-1)n+2k)$ bits. Then for any $((q_1q_G), (q_P, q_C))$ -strong non-uniform model combiner distinguisher adversary $\mathcal{A}$, it holds that*

$$\text{Adv}_{\text{Envelope-C}[T], \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \mathcal{O}\left(\frac{q_G^2}{2^{2k-n}} + \frac{\sigma^2}{2^n} + \frac{(n+1)q_G\sigma}{2^k} + \frac{q_1}{2^k}\right)$$

where $\sigma = 2(q_Pq_G(\ell+1) + q_Cq_G(\ell+1)^2)$ and $q_{\text{sim}} = q_P$.

5.3 Candidate Instantiations

In this section, we instantiate our combiner via popular hash functions.

Merkle-Damgård based Hashes. *Instantiations.* We observe that for Merkle-Damgård based hashes, salting each message block can be achieved simply via our Brine *message padding* algorithm. The message is divided into blocks of m bits, the k -bit random string Z_0 is padded with each message block to make a $m+k$ size message block. Thus, our mode (and the combiner) can be directly implemented in popular Merkle-Damgård based modes *without making any changes* to the implementation.

Parameters. From Equation 1, the total randomness used in the XOR combiner is $(4k+2m)$ bits for an $(n+m+k)$ -bit-to- n bit compression function. In order to get $\frac{n}{2}$ -bit security from Theorem 16, we can set $k = \frac{3n}{4}$. Thus, the total randomness requirement is $(3n+2m)$ for an $(m+\frac{7n}{4})$ -bit-to- n bit compression function. Specifically, for $3n$ -to- n bit compression function f , the rate m can be $\frac{5n}{4}$, and the total randomness required by our combiner is $(3n+\frac{5n}{2})$ bits. Similarly, for $2n$ -to- n bit compression function f , m can be $\frac{n}{4}$, and the total randomness required by our combiner is $(3n+\frac{n}{2})$ bits.

Two most popular hash functions that employ Merkle-Damgård modes are SHA256 and Blake2. In SHA256 and Blake2s, the underlying compression function is $f: \{0,1\}^{768} \rightarrow \{0,1\}^{256}$; it is a $3n$ -to- n bit compression function with $n = 256$. Our salted Envelope-MD xor combiner (see Figure 5b) thus requires total randomness of $\frac{11 \times 256}{2} = 1408$ bits. Each compression function call thus processes $\frac{5 \times 256}{4} = 320$ bits of message data, compared to 512 bits of standardized specification. This rate leads to efficiency slow down by a factor of 1.6.

In case of SHA512, Blake2b, the compression function is $f: \{0,1\}^{1536} \rightarrow \{0,1\}^{512}$, again a $3n$ -to- n bit function. Our salted Envelope-MD uses $\frac{11 \times 512}{2} = 2816$ bits; processing $\frac{5 \times 512}{4} = 640$ bits of message block in every compression function call.

Tree based Hashes. *Instantiations.* The Salted Tree mode of Figure 5c uses HAIFA padding, the design is based on Blake3, a tree based hashing mode

supported in OpenZFS, Linux Kernel 5.17 and IETF Internet-Draft. Blake3 is extremely fast on CPUs (SIMD, AVX2, AVX-512, NEON) and scales well to multiple cores, reported to be considerably faster than SHA256/SHA3. Blake 3 compression function, at the tree node level ⁵, allows “keying” the function with a 256-bit salt (called the keyed hash mode [43]). Thus, our salted tree mode can be directly instantiated using a Blake3-like structure with one additional call to the compression function.

Parameters. From Equation 1, the total randomness used in the tree mode is $2((t-1)n + 2k)$ for an $(tn + k + 2\log \ell)$ -bit-to- n -bit compression function $n + k + m$ -bit-to- n bit compression function. In order to get birthday security from Theorem 17, we can set $k = \frac{3n}{4}$ leading to total randomness use to be $(2t+1)n$. Setting $t = 2$ and $n = 256$, we get the randomness requirement of $5 \times 256 = 1280$ bits. The rate in this case remains almost the same as the basic tree hash, while the required compression function input is k -bit larger than what is required for a t -ary tree with HAIFA mode. In general, instantiating our salted envelope tree combiner with $tn + 2\log \ell$ -to- n -bit compression function and $(t-1)$ -ary basic hash tree with HAIFA padding requires $2tn$ total random bits. In that case, the efficiency slowdown would be of a factor of $\frac{t}{t-2}$ compared to the standard t -ary hash with HAIFA padding.

6 Proofs

6.1 Proof of Theorem 13

Overview. We start with an overview. Theorem 13 is similar in spirit to the PRA to PRO Theorem of [28] (restated in Theorem 7). For ease of understanding we recall the main idea of Theorem 7 and the proof. Roughly speaking, PrA security of a hashing mode H^f ensures that for any target point y , at most a single message M can be extracted from the f -query transcript such that $H^f(M) = y$. In case, no such message can be extracted at the moment, additional f queries will not help the adversary to construct any such M . Authors of [28] proved that the Envelope- H hash achieved via a composition of the PrA hash H^f with an FIL-RO R achieves PRO security. The simulator design is conceptually simple. The f -queries are answered honestly, sampling points uniformly at random. For every R query y , the PrA extractor is invoked on y to get a preimage M , and the random oracle output $\mathcal{F}(M)$ is returned. If no preimage is found (extractor returns $\perp$), a uniform random response is returned. The PrA security ensures that the adversary will not be able to find any new preimage of y , thus the simulator response remains consistent.

In our Envelope- H construction, $H^{f_{z_0}}$ implements a PrA hash whereas the FIL-RO function f_{z_1} implements the envelope function R . In order to ensure

⁵ The node level compression function of Blake3 is further instantiated via a linear mode. We restrict our attention only to the node level compression function here. In other words, the FIL-RO f is the node level compression function, and so is the adversarial implementation $\mathcal{G}^f$.

Oracle sim-main (h, x_1, x_2) if $x_2 = Z_0$ sim-query-Z_0 (h, x_1) if $x_1 x_2 = Z_1$ sim-query-Z_1 (h) if $(h, x_1, y) \in \tau_f[x_2]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[x_2] = \tau_f[x_2] \cup \{(h, x_1, y)\}$ return y Algorithm sim-query-Z_0 (h, m, Z_0) if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ return y Algorithm f-Z_1 ($M', h, Z_1, \text{flag}_g$) if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y	Algorithm sim-query-Z_1 (h) if $\text{flag}_g = 1$ // $\mathcal{G}^f$ query computing $H^{\mathcal{G}^f}(M_G)$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ if $\text{flag}_g = 0$ // Query from the PRO-adversary $\mathcal{A}$ $M = \text{Ex}(h, \tau_f[Z_0])$ if $M = \perp$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ if $M \neq \perp$ $\text{flag}_g = 1$ $M_G = M$ $t = \text{Envelope-}H_{Z'}^{\mathcal{G}^{\text{sim}}}(M)$ $\text{flag}_g = 0$ $M_G = \perp$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $\mathcal{F}(M) \oplus t$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ return $\mathcal{F}(M) \oplus t$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y if $(*, h, y, *) \notin \tau_f[Z_1]$ // the remaining case $y = \mathcal{F}(M) \oplus t$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, 0)\}$ return y endif // Ending $(*, h, y, *) \notin \tau_f[Z_1]$ case endif // Ending $M \neq \perp$ case endif // Ending $\text{flag}_g = 0$ case
--	--

Fig. 7: General Model Simulator for Envelope Combiner: Initialized with $Z = (Z_0, Z_1)$, $Z' = (Z'_0, Z'_1)$, $\tau_f = []$, $\text{flag}_g = 0$. The simulator is inconsistent in boxed Conditions. The first boxed condition implies $\mathcal{G}^f$ -influence =1. The second condition implies a PRA attack.

that the envelope function f_{Z_1} remains independent from $H^{f_{Z_0}}$, we need that Z_0 , and the last k -bits of Z_1 are different. The probability of the Bad event of collision in those k bits is $\frac{1}{2^k}$. So, with all but $\frac{1}{2^k}$ probability, Envelope-H design instantiated with random salt $Z = (Z_0, Z_1)$ is a PRO secure hash.

Now, let us look at our Envelope-C[H] combiner. The simulator follows the same principle as above. In all cases except when answering f_{Z_1} queries, the simulator algorithm (described in Figure 7) can reply perfectly honestly; if the query-answer is not in the transcript, sample a uniform random element $y \xleftarrow{\$} \{0, 1\}^n$, update the transcript and return y . However, when the adversary

makes a $f_{Z_1}(h) = f(h, Z_1)$ query, the simulator output needs to stay consistent with $\mathcal{F}$ oracle output. The simulator checks whether $h = H^{f_{Z_0}}(M)$ for some M known to the adversary. In other words, the simulator needs to extract out $M \leftarrow \text{Ex}(h, \tau_f[Z_0])$, and then program the output as $\mathcal{F}(M) \oplus \text{Envelope-}H_{Z_1}^{\mathcal{G}^f}(M)$. While doing so, the simulator can become inconsistent if the output has already been programmed to a different value. In that case one of the following two events happen:

- (a) Bad_G : $(M', h, y, 1) \in \tau_f[Z_1]$. For the $f(h, Z_1)$ made by the adversary, the extractor outputs $\text{Ex}(h, \tau_f[Z_0]) \neq \perp$, but $f(h, Z_1)$ was set and thus already queried by the adversarial implementation $\mathcal{G}^f$. This is the winning condition of the $\mathcal{G}^f$ -influence game.
- (b) Bad_{PRA} : $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$. For a $f(h, Z_1)$ made by the adversary, the extractor output $\text{Ex}(h, \tau_f[Z_0]) \neq \perp$. However, $f(h, Z_1)$ has already been set by some previous query (with $\text{flag}_g = 0$), and M' , the extractor output at that time was different from M , the current extractor output. This is a winning condition of the PrA game.

We prove the theorem via game playing technique. Looking ahead, the interesting hybrids are the intermediate games $G2, G3$ (Figure 11) where the following holds

$$\begin{aligned} \Delta_{\mathcal{A}(Z, Z')}(\text{REAL}, G2) &= 0 & \Delta_{\mathcal{A}(Z, Z')}(G3, \text{IDEAL}) &= 0 \\ \Delta_{\mathcal{A}(Z, Z')}(G2, G3) &\leq \Pr[\text{Bad}_G] + \Pr[\text{Bad}_{\text{PRA}}] \end{aligned}$$

Description of Games. Game REAL: In this game, the adversary is interacting with (C, f) .

Game G0: Oracle C checks whether the envelope query $f(h, Z_1)$ is already set to be some y in the transcript. If yes, proceed with the same value without querying $f(h, Z_1)$ again. In addition, the game introduces some flag to remember whether a $f(h, Z_1)$ query is made during an evaluation of $\mathcal{G}^f$ (with the corresponding message M) or it is made by the adversary. There is no change in the output.

$$\Delta_{\mathcal{A}(Z, Z')}(\text{REAL}, G0) = 0$$

Game G1: In this game, algorithm $f\text{-query}(h, Z_1)$ introduces few additional bookkeeping codes and conditions, to be used in the future games. In particular, for every query made either directly by the adversary or oracle C , the extractor is queried to find potential a pre-image M . If such M exists (i.e. $h = H^{f_{Z_0}}(M)$), we compute $t' = \text{Env-}H^{\mathcal{G}}(M)$. Note, the process does not go into a recursion as the queries made by $\mathcal{G}^f$ is handled by the $\text{flag}_g = 1$ condition.

Once the process is complete, the game checks if $f(h, Z_1)$ is already set to be some y in the transcript. If yes, proceed with the same value. Otherwise, it samples a random value y , updates the transcript and returns y .

REAL, G0 // Boxed commands are executed only in G0 return $b = \mathcal{A}^{C_{(Z, Z')}, f\text{-main}}$ Oracle $C_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{f_{Z_0}}(M)$ $t' = \text{Env-}H^{\mathcal{G}}(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $y \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ return $y \oplus t'$ else // Case $M = M'$ return $y \oplus t'$ endif endif $y = f\text{-query}(h, Z_1)$ return $y \oplus t'$ Oracle $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y	Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $\text{Env-}H^{\mathcal{G}}(M)$ $\text{flag}_g = 1 \quad M_G = M$ $h' = H^{G_{Z'_0}^{f\text{-main}}}(M)$ $t' = \mathcal{G}^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0 \quad M_G = \perp$ Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M_G, h, y, \text{flag}_g)\}$ return y endif if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(\perp, h, y, \text{flag}_g)\}$
---	--

Fig. 8: Game REAL, G0. Initialized with $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1), \tau_f = \emptyset, \text{flag}_g = 0$

While there are many additional f evaluations, the outputs of f are always sampled honestly. Moreover, the transcript is never over-written. Hence, the output distribution is unchanged.

$$\Delta_{\mathcal{A}(Z, Z')} (G0, G1) = 0$$

Game G2: This game is identical to G1 except in one place. In algorithm $f\text{-query}(h, Z_1)$, if the extractor returns an $M \neq \perp$, and the output of (h, Z_1) is not found in the transcript, even after the computation of $t' = \text{Env-}H^{\mathcal{G}}(M)$, then the return value is set as $\mathcal{F}(M) \oplus t'$. Recall $\mathcal{F}$ is a random oracle. As the output of $f(h, Z_1)$ was not set in the transcript, and the extractor is correct (i.e. it never returned M for any other query), $\mathcal{F}(M)$ has not been queried before. So, until that point, the transcript is independent from $\mathcal{F}(M)$. We conclude t' is independent from

Games $G0, \boxed{G1}$ // Boxed commands are executed only in $G1$ return $b = \mathcal{A}^{C_{(Z,Z')}, f\text{-main}}$ Oracle $C_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{fZ_0}(M)$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $y \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ return $y \oplus t'$ else // Case $M = M'$ return $y \oplus t'$ endif endif $y = f\text{-query}(h, Z_1)$ return $y \oplus t'$ Oracle $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $\text{Env-}H^G(M)$ $\text{flag}_g = 1 \quad M_G = M$ $h' = H^{G^{f\text{-main}}_{Z'_0}}(M)$ $t' = G^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0 \quad M_G = \perp$	Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M_G, h, y, \text{flag}_g)\}$ return y endif $M = \text{Ex}(h, \tau_f[Z_0])$ $\boxed{\text{if } M \neq \perp}$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, *) \in \tau_f[Z_1]$ $\boxed{\wedge M = M'}$ return y if $(M', h, y, *) \in \tau_f[Z_1]$ $\boxed{\wedge M \neq M'}$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\boxed{\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}}$ return y endif $\boxed{\text{else}} // M = \perp \wedge \text{flag}_g = 0$ if $(M', h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(\perp, h, y, \text{flag}_g)\}$ return y endif
---	---

Fig. 9: Game $G0, \boxed{G1}$. Initialized with $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1), \tau_f = \emptyset, \text{flag}_g = 0$

$\mathcal{F}(M)$. Thus the output distribution remains unchanged.

$$\Delta_{\mathcal{A}(Z, Z')} (G1, G2) = 0$$

Game G3: In this game we introduce the bad events. For any $C(M)$ query or any $f(h, Z_1)$ query for which the extractor has returned a $M \neq \perp$, after the

Games G1, G2 // Boxed commands are executed only in G2 return $b = \mathcal{A}^{C_{(Z, Z'), f\text{-main}}}$ Oracle $C_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{f_{Z_0}}(M)$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $y \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ return $y \oplus t'$ else // Case $M = M'$ return $y \oplus t'$ endif endif $y = f\text{-query}(h, Z_1)$ return $y \oplus t'$ Oracle $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $\text{Env-}H^G(M)$ $\text{flag}_g = 1$ $M_G = M$ $h' = H^{G^{f\text{-main}}_{Z'_0}}(M)$ $t' = G^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0$ $M_G = \perp$	Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M_G, h, y, \text{flag}_g)\}$ return y endif $M = \text{Ex}(h, \tau_f[Z_0])$ if $M \neq \perp$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, *) \in \tau_f[Z_1] \wedge M = M'$ return y if $(M', h, y, *) \in \tau_f[Z_1] \wedge M \neq M'$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $y = \mathcal{F}(M) \oplus t'$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}$ return y endif else // $M = \perp \wedge \text{flag}_g = 0$ if $(M', h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(\perp, h, y, \text{flag}_g)\}$ return y endif
---	--

Fig. 10: Game G1, G2. Initialized with $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1), \tau_f = \emptyset, \text{flag}_g = 0$

computation of $t' = \text{Env-}H^G(M)$, we check whether $f(h, Z_1)$ is already set in the transcript. If yes, and it was set with $\text{flag}_g = 1$ (i.e. set during some $\text{Env-}H^G$ evaluation) the flag Bad_G is set. If it was set with $\text{flag}_g = 0$, but with some other $M' \neq M$, the flag Bad_{PRA} is set. For these bad events, game G3 “implicitly” considers the corresponding $f(h, Z_1)$ to be $\mathcal{F}(M) \oplus t'$, and outputs accordingly. There are two non-bad code changes in G3. During the evaluation of $C(M)$,

Games G2, G3 // Boxed commands are executed only in G3 return $b = \mathcal{A}^{C_{(Z, Z')}, f\text{-main}}$ Oracle $C_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{f_{Z_0}}(M)$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ Bad_G = 1 return $\mathcal{F}(M)$ $\text{Bad}_G = 1$ return $y \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ $\text{Bad}_{PRA} = 1$ return $\mathcal{F}(M)$ if $M \neq M'$ $\text{Bad}_{PRA} = 1$ return $y \oplus t'$ else // Case $M = M'$ return $\mathcal{F}(M)$ return $y \oplus t'$ endif return $\mathcal{F}(M)$ return $f\text{-query}(h, Z_1) \oplus t'$ Oracle $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $\text{Env-}H^G(M)$ $\text{flag}_g = 1$ $M_G = M$ $h' = H^{G^{f\text{-main}}_{Z'_0}}(M)$ $t' = G^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0$ $M_G = \perp$	Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ $M = \text{Ex}(h, \tau_f[Z_0])$ if $M = \perp \wedge \text{flag}_g = 0$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ else // $M \neq \perp \wedge \text{flag}_g = 0$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ Bad_G = 1 return $\mathcal{F}(M) \oplus t'$ $\text{Bad}_G = 1$ return y if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ Bad_{PRA} = 1 return $\mathcal{F}(M) \oplus t'$ $\text{Bad}_{PRA} = 1$ return y if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y $y = \mathcal{F}(M) \oplus t'$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}$ return y endif Algorithm $f\text{-}Z_1(M', h, Z_1, \text{flag}_g)$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y
--	--

Fig. 11: Game G2, G3. Initialized with $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1), \tau_f = \emptyset, \text{flag}_g = 0$

for the remaining cases when $f(h, Z_1)$ is already set in the transcript with the message M , or $f(h, Z_1)$ has not been set, the oracle C still returns $\mathcal{F}(M)$. If Bad flag has not been set, in both these cases $f(h, Z_1)$ evaluation would indeed return $\mathcal{F}(M) \oplus t'$, making the $C(M) = \mathcal{F}(M)$. Thus we conclude, conditioned on

Games G3, IDEAL // Boxed commands are executed only in the IDEAL GAME return $b = \mathcal{A}^{C_{(Z, Z')}, f\text{-main}}$ Oracle $C_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ return $\mathcal{F}(M)$ $h = H^{fZ_0}(M)$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $\mathcal{F}(M)$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ return $\mathcal{F}(M)$ else // Case $M = M'$ return $\mathcal{F}(M)$ endif return $\mathcal{F}(M)$ Oracle $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $\text{Env-}H^G(M)$ $\text{flag}_g = 1 \quad M_G = M$ $h' = H^{G^{fZ'_0}}(M)$ $t' = G^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0 \quad M_G = \perp$	Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ $M = \text{Ex}(h, \tau_f[Z_0])$ if $M = \perp \wedge \text{flag}_g = 0$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ else // $M \neq \perp \wedge \text{flag}_g = 0$ $t' = \text{Env-}H^G(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ $\text{Bad}_G = 1$ return $\mathcal{F}(M) \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ $\text{Bad}_{\text{PRA}} = 1$ return $\mathcal{F}(M) \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y $y = \mathcal{F}(M) \oplus t'$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}$ return y endif Algorithm $f\text{-}Z_1(M', h, Z_1, \text{flag}_g)$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y
--	---

Fig. 12: Game G3, Ideal. Initialized with $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1), \tau_f = \emptyset, \text{flag}_g = 0$

not Bad, the output distribution of G2 and G3 are identical. By the fundamental lemma of game playing proof,

$$\Delta_{\mathcal{A}(Z, Z')}(G2, G3) \leq \Pr[\text{Bad}_G] + \Pr[\text{Bad}_{\text{PRA}}]$$

Game IDEAL: The only change in the IDEAL game is that oracle $C(M)$ directly returns $\mathcal{F}(M)$ without any other computation. Oracle $f\text{-main}$ and other sub-

routines constitute the simulator. As $C(M)$ was always returning $\mathcal{F}(M)$ in the game G3, and the oracle C evaluation was not changing any game state, the output distributions remain unchanged.

$$\Delta_{\mathcal{A}(Z, Z')}(\text{IDEAL}, G3) = 0.$$

Bad Events, and their connection with PRA and $\mathcal{G}^f$ -influence

$\mathcal{G}^f$ -influence adversary $\mathcal{C}$. Suppose we have a PRO adversary $\mathcal{A}$. We describe a $\mathcal{G}^f$ -influence adversary $\mathcal{C}$ (based on the algorithm in the game G2 in Figure 11) which will have oracle access to the FIL-RO f . The complete description of $\mathcal{C}$ can be found in Figure 13. On input $(\mathcal{G}^f, Z, Z')$, algorithm $\mathcal{C}$ will run $\mathcal{A}$ with the salt (Z, Z') and provide two interfaces, oracle C and Oracle f -main. To answer the construction queries, $\mathcal{C}$ executes the oracle C of the game $G2$. The primitive $f(h, Z_1)$ queries made by $\mathcal{A}$ as well as $\mathcal{G}^f$ are answered following the algorithm f -query(h, z_1) in the game $G2$ in Figure 11. Remaining primitive queries are forwarded to the f oracle, while maintaining a transcript of this interaction in the (M, h, y, flag_g) format.

If the flag Bad_G is set, then for some construction query $C_{Z, Z'}(M)$ with $h = H^{f_{Z_0}}(M)$ or for some f -query(h, Z_1) with $M = \text{Ex}(h, \tau_f[Z_0])$, there is an entry in $\tau_f[Z_1]$ with $(M', h, y, 1)$. $\mathcal{C}$ outputs M and M' . If the Bad_G is never set before $\mathcal{A}$ terminates, $\mathcal{C}$ outputs two random messages $M, M' \in \{0, 1\}^m$.

The $(M', h, y, 1)$ entry in $\tau_f[Z_1]$ that sets $\text{Bad}_G = 1$ was set by the subroutine call f -Z₁($M', h, Z, 1$) which was made by f -query(h, Z_1). The $\text{flag}_g = 1$ implies that this f -query(h, Z_1) call was made during the computation of $\text{Env-}H^{\mathcal{G}}(M')$ query. This proves that $h \in Q_G \cup Q_H$ in the $\mathcal{G}^f$ -influence game of Figure 6.

$$\Pr[\text{Bad}_G] \leq \Pr[\mathcal{G}^f\text{-influence}[\mathcal{C}]^{\mathcal{A}} = 1]$$

PRA adversary $\mathcal{B}$. Suppose we have a PRO adversary $\mathcal{A}$. We describe a PRA adversary $\mathcal{B}$ (based on the algorithm in the game G2 in Figure 11) which will have oracle access to the FIL-RO $f_{Z, 0}$. On input Z_0 , algorithm $\mathcal{B}$ samples Z'_0, Z_1, Z'_1 to create the salt $(Z = (Z_0, Z_1); Z' = (Z'_0, Z'_1))$. In addition, $\mathcal{B}$ generates the adversarial implementation $\langle \mathcal{G} \rangle$ by running $\mathcal{A}_1$. $\mathcal{B}$ then executes $\mathcal{A}$ with the salt (Z, Z') and provide two interfaces, oracle C and Oracle f -main. To answer the construction queries, $\mathcal{C}$ executes the oracle C of the game $G2$. The primitive $f(h, Z_1)$ queries made by $\mathcal{A}$ as well as $\mathcal{G}^f$ are answered following the algorithm f -query(h, Z_1) in the game $G2$ in Figure 11. Note, the extractor queries made during the f -query(h, Z_1) evaluations denote the commitment made by $\mathcal{B}$ in the PRA game. The remaining f queries including the primitive $f(h, m, Z_0)$ queries are forwarded to the oracle f . Remaining primitive queries are simulated by $\mathcal{B}$ following the algorithms in Figure 11. The complete description of $\mathcal{B}$ is included in Figure 14.

If the flag Bad_{PRA} is set, then for some construction query $C_{Z, Z'}(M)$ with $h = H^{f_{Z_0}}(M)$ or for some f -query(h, Z_1) with $M = \text{Ex}(h, \tau_f[Z_0])$ with $M \neq \perp$,

$\mathcal{G}^f$ -adversary $\mathcal{C}^f(\langle \mathcal{G} \rangle, Z, Z')$ $b = \mathcal{A}^{C, f\text{-main}}(Z, Z')$ $M, M' \xleftarrow{\$} \{0, 1\}^m$ // Case Bad_G is still 0 return (M, M') Oracle $\mathbf{C}_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{f_{Z_0}}(M)$ $t' = \text{Env-}H^{\mathcal{G}}(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ $\text{Bad}_G = 1$ Halt and Output (M, M') if $(M', h, y, 0) \in \tau_f[Z_1]$ return $y \oplus t'$ return $f\text{-query}(h, Z_1) \oplus t'$ Algorithm $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y = f(h, x_1, x_2)$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $f\text{-}Z_1(M', h, Z_1, \text{flag}_g)$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y = f(h, Z_1)$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y Algorithm $\mathcal{F}(M)$ if $(M, y) \in \tau\text{-}F$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau\text{-}F = \tau\text{-}F \cup \{(M, y)\}$ return y	Algorithm $f\text{-query}(h, Z_1)$ $M = \text{Ex}(h, \tau_f[Z_0])$ if $\text{flag}_g = 1$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ if $M = \perp \wedge \text{flag}_g = 0$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ else // $M \neq \perp \wedge \text{flag}_g = 0$ $t' = \text{Env-}H^{\mathcal{G}}(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ $\text{Bad}_G = 1$ Halt and Output (M, M') if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ return y $y = \mathcal{F}(M) \oplus t'$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}$ return y endif Algorithm $f\text{-query}(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y = f(h, m, Z_0)$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ Algorithm $\text{Env-}H^{\mathcal{G}}(M)$ $\text{flag}_g = 1$ $M_G = M$ $h' = H^{\mathcal{G}^f_{Z'_0}}(M)$ $t' = \mathcal{G}^f(h', Z'_1)$ $\text{flag}_g = 0$ $M_G = \perp$
---	---

Fig. 13: $\mathcal{G}^f$ -influence adversary $\mathcal{C}$ with an oracle access to f .

there is an entry in $\tau_f[Z_1]$ with $(M', h, y, 0)$ and $M \neq M'$. $\mathcal{C}$ outputs (M, h) . If the $\text{Bad}_{P_{RA}}$ is never set before $\mathcal{A}$ terminates, $\mathcal{B}$ outputs a random message $M \in \{0, 1\}^m$.

PRA Adversary $\mathcal{B}^f()$ $\langle \mathcal{G} \rangle = \mathcal{A}_1(n)$ $Z_0, Z'_0 \xleftarrow{\\$} \{0, 1\}^k; Z_1, Z'_1 \xleftarrow{\\$} \{0, 1\}^{m+k}$ $Z = (Z_0, Z_1); Z' = (Z'_0, Z'_1)$ $b = \mathcal{A}^{C, f\text{-main}}(Z, Z')$ Commit $h \xleftarrow{\\$} \{0, 1\}^n$ // Case Bad_{PRA} is still 0 $M \xleftarrow{\\$} \{0, 1\}^m$ Output (M, h) Oracle $\mathbf{C}_{Z=(Z_0, Z_1), Z'=(Z'_0, Z'_1)}(M)$ $h = H^{f-Z_0}(M)$ $t' = \text{Env-}H^{\mathcal{G}}(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $y \oplus t'$ if $(M', h, y, 0) \in \tau_f[Z_1]$ if $M \neq M'$ $\text{Bad}_{PRA} = 1$ Halt and Output M else // Case $M = M'$ return $y \oplus t'$ endif return $f\text{-query}(h, Z_1) \oplus t'$ Algorithm $f\text{-main}(h, x_1, x_2)$ if $x_2 = Z_0$ $f\text{-query}(h, x_1, Z_0)$ if $x_1 x_2 = Z_1$ $f\text{-query}(h, Z_1)$ if $(h, x_1, x_2, y) \in \tau_f$ return y $y = f(h, x_1, x_2)$ $\tau_f = \tau_f \cup \{(h, x_1, x_2, y)\}$ return y Algorithm $f\text{-}Z_1(M', h, Z_1, \text{flag}_g)$ if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y = f(h, Z_1)$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y	Algorithm $f\text{-query}(h, Z_1)$ if $\text{flag}_g = 1$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ Commit h to get $M = \text{Ex}(h, \tau_f[Z_0])$ if $M = \perp \wedge \text{flag}_g = 0$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ else // $M \neq \perp \wedge \text{flag}_g = 0$ $t' = \text{Env-}H^{\mathcal{G}}(M)$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return y if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ $\text{Bad}_{PRA} = 1$ Halt and Output M $y = \mathcal{F}(M) \oplus t'$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, \text{flag}_g)\}$ return y endif Algorithm $f\text{-}Z_0(h, m, Z_0)$ if $(h, m, y) \in \tau_f[Z_0]$ return y $y = f(h, m, Z_0)$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ return y Algorithm $\text{Env-}H^{\mathcal{G}}(M)$ $\text{flag}_g = 1$ $M_G = M$ $h' = H^{\mathcal{G}^{f\text{-main}}_{Z'_0}}(M)$ $t' = \mathcal{G}^{f\text{-main}}(h', Z'_1)$ $\text{flag}_g = 0$ $M_G = \perp$ Algorithm $\mathcal{F}(M)$ if $(M, y) \in \tau\text{-}F$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau\text{-}F = \tau\text{-}F \cup \{(M, y)\}$ return y
---	---

Fig. 14: PRA adversary $\mathcal{B}^f$. The implementation $\mathcal{G}$ is generated by adversary $\mathcal{A}_1$

The $(M', h, y, 0)$ entry in $\tau_f[Z_1]$ that sets $\text{Bad} = 1$ can be of two type. If $M' = \perp$, it was set by a subroutine call $f\text{-}Z_1(\perp, h, Z, 1)$ made by $f\text{-query}(h, Z_1)$ where $\text{Ex}(h, \tau_f[Z_0])$ output was $\perp$. Otherwise, it was set during a previous $f\text{-}$

query(h, Z_1) call where the extractor output was $M' \neq M$. Thus $\mathcal{B}$ has successfully found another M such that $h = H^{f_{Z_0}}(M)$.

$$\Pr[\text{Bad}_{PRA}] \leq \mathbf{Adv}_{H, \text{Ex}}^{\text{pra}}(\mathcal{B})$$

Query Bounds of $\mathcal{B}$ and $\mathcal{C}$. Suppose the PRO adversary $\mathcal{A}$ makes q_P, q_C many queries to the oracles P and C respectively. This implies, $\mathcal{A}$ makes at most $\hat{\sigma} = (q_P + q_C \ell)$ blocks of queries, where ℓ is the maximum block-length of the message processed by hash H . Then the number of queries made by $\mathcal{B}$ and $\mathcal{C}$ (the extended transcript including the queries required to compute $\text{Envelope-}H_Z^f$ and $\text{Envelope-}H_{Z'}^{\mathcal{G}^f}$) is at most $(\hat{\sigma} + \hat{\sigma} q_G(\ell + 1))$. This is because $H^{\mathcal{G}^f}$ evaluation can get triggered at most σ times, and for each of the messages, $\mathcal{G}^f$ gets evaluated at most $(\ell + 1)$ many times. For each of these evaluations, $\mathcal{G}^f$ makes at most q_G many queries. Thus, the total number of queries in the extended transcript can be bounded by $\hat{\sigma}(1 + q_G(\ell + 1))$. Replacing the value of $\hat{\sigma} = (q_P + q_C \ell)$, we get the maximum number of queries to be at most $2(q_P q_G(\ell + 1) + q_C q_G(\ell + 1)^2)$.

This concludes the proof of Theorem 13. $\square$

6.2 Proof of Theorem 15: Bounding $\mathcal{G}^f$ -influence advantage of $\text{Envelope-}C[H]$ combiners

The implementation $\mathcal{G}^f$ evaluations happen through the combiner C . On input a message M consisting of m -bit message blocks $M_1, \dots, M_\ell$, the combiner C evaluates $\mathcal{G}^f$ sequentially. For example, for Envelope-MD combiner, C evaluates $\mathcal{G}^f$ on $h'_0 = \mathcal{G}^f(\text{IV}, M_0, Z'_0)$, $h'_1 = \mathcal{G}^f(h'_0, M_1, Z'_0)$, and so on, and finally on $h_e = \mathcal{G}^f(h_\ell, Z'_1)$. Recall that $Z_0, Z'_0 \in \{0, 1\}^k$ and $Z_1, Z'_1 \in \{0, 1\}^{m+k}$.

GF-Bad transcript We will bound the probability of the event that the transcript is GF-Bad, that is during some $\text{Envelope-}H_{Z'}^{\mathcal{G}^f}(M')$ evaluation, the adversarial compression function $\mathcal{G}^f$ queries $f(h^*, Z_1)$ such that h^* is the output of some f_{Z_0} query in the transcript.

Recall, for $\mathcal{G}^f$ -influence game to output 1, the evaluation $\text{Envelope-}H_Z^f(M)$ needs to query $f(h^*, Z_1)$ such that the same $f(h^*, Z_1)$ has been queried by $\mathcal{G}^f$ during the $\text{Envelope-}H_{Z'}^{\mathcal{G}^f}(M')$ evaluation. As H is a simple hash mode (see Definition 5), the chaining value h^* in $f(h^*, Z_1)$ query must be the output of some f_{Z_0} evaluation. That would imply that the transcript is GF-Bad. In other words,

Lemma 18.

$$\mathbf{Adv}_{\mathcal{G}^f, \text{Envelope-}C[H]}^{\text{influence}}(\mathcal{C}) \leq \Pr[\text{GF-Bad}]$$

In the rest of the section we shall bound $\Pr[\text{GF-Bad}]$.

Bounding $\Pr[\text{GF-Bad}]$

Types of GF-Bad transcript. Recall, during some Envelope- $H_{Z'}^{\mathcal{G}^f}(M')$ evaluation, the Envelope- $H_{Z'}^{\mathcal{G}^f}$ mode queries $\mathcal{G}^f(x, Z'_0)$ (for some $x \in \{0, 1\}^{n+m}$) or $\mathcal{G}^f(x, Z'_1)$ (for some $x \in \{0, 1\}^n$). Recall the following notations we use. $f_{Z_0} := f(\cdot, \cdot, Z_0)$ $f_{Z_1} := f(\cdot, \cdot, Z_1)$. A GF-Bad transcript can be one of the following two types.

1. $T1$: there exists $\mathcal{G}^f(x, Z'_b)_{b \in \{0,1\}}$ evaluation that queries both f_{Z_0} and f_{Z_1} .
2. $T2$: All the $\mathcal{G}^f(x, Z'_0)$ and $\mathcal{G}^f(x, Z'_1)$ evaluations that query f_{Z_1} do not query f_{Z_0} .

Bounding $\Pr[\text{GF-Bad} \wedge T1]$

Lemma 19. *It holds that*

$$\Pr[\exists x \in \{0, 1\}^{n+m} \text{ s.t. } \mathcal{G}^f(x, Z'_0) \text{ queries both } f_{Z_0} \text{ and } f_{Z_1}] \leq \frac{q_{\mathcal{G}}^2}{2^{2k-n}}$$

$$\Pr[\exists x \in \{0, 1\}^n \text{ s.t. } \mathcal{G}^f(x, Z'_1) \text{ queries both } f_{Z_0} \text{ and } f_{Z_1}] \leq \frac{q_{\mathcal{G}}^2}{2^{2k+m-n}}$$

where the probabilities are taken over f, Z_0, Z_1, Z'_0, Z'_1

Proof. We prove the case for $\mathcal{G}^f(x, Z'_0)$ queries. The case for $\mathcal{G}^f(x, Z'_1)$ queries holds analogously. Fix a (f, Z'_0, Z'_1) . For each $x \in \{0, 1\}^{n+m}$, define

$$Q_x := \{(Z_0, Z_1) \in \{0, 1\}^k \times \{0, 1\}^{m+k} \mid \mathcal{G}^f(x, Z'_0) \text{ queries both } f_{Z_0} \text{ and } f_{Z_1}\}$$

and

$$Q := \bigcup_{x \in \{0, 1\}^{n+m}} Q_x$$

As each $\mathcal{G}^f(x, Z'_0)$ makes at most $q_{\mathcal{G}}$ number of queries, for each x

$$|Q_x| \leq q_{\mathcal{G}}^2$$

Thus, taking sum over all possible x ,

$$|Q| = \sum_{x \in \{0, 1\}^{n+m}} |Q_x| \leq 2^{n+m} \times q_{\mathcal{G}}^2$$

Finally,

$$\Pr[(Z_0, Z_1) \in Q] \leq \frac{2^{n+m} \times q_{\mathcal{G}}^2}{2^{m+2k}} = \frac{q_{\mathcal{G}}^2}{2^{2k-n}}$$

□

The following corollary is immediate from Lemma 19.

Corollary 20. *It holds that*

$$\Pr[\text{GF-Bad} \wedge T1] \leq \Pr[T1] \leq \frac{q_{\mathcal{G}}^2}{2^{2k-n}} \quad (2)$$

Bounding $\Pr[\mathbf{GF}\text{-Bad} \wedge T2]$ We prove the bound for $\mathcal{G}^f(x, Z'_0)$ queries. The bounds for $\mathcal{G}^f(x, Z'_1)$ queries are bound in the identical way, and are in fact bounded above by the bounds for $\mathcal{G}^f(x, Z'_0)$ queries.

If $\mathbf{GF}\text{-Bad} \wedge T2$ holds then there must be some $f(*, *, Z_0)$ query output y in the transcript such that for some x , $\mathcal{G}^f(x, Z'_0)$ queries $f(y, Z_1)$. In order the bound the probability of such event, we first find the f_{z_1} query sets of $\mathcal{G}^f$. We define the following set for every choice of (Z, Z', f) .

$$B_{(Z, Z', f)} \stackrel{\text{def}}{=} \{x \in \{0, 1\}^{n+m} \mid \mathcal{G}^f(x, Z'_0) \text{ queries } f_{z_1} \text{ but does not query } f_{z_0}\}$$

Now we define the query set as

$$L_{(Z, Z', f)} \stackrel{\text{def}}{=} \{y \in \{0, 1\}^n \mid \exists x \in B_{(Z, Z', f)}, \mathcal{G}^f(x, Z'_0) \text{ queries } f(y, Z_1)\}$$

Note, that both the sets are defined over all possible input to $\mathcal{G}^f$, and is not dependent on a particular adversary. The following lemma says that the set $L_{(Z, Z', f)}$ is invariant of changing f_{z_0} . Thus the function f_{z_0} has full entropy even conditioned on $L_{(Z, Z', f)}$.

Lemma 21. *Fix Z'_0, Z_0, Z_1 . Fixing f for all but $f(\cdot, \cdot, Z_0)$, $L_{(Z, Z', f)}$ is same for all choices of $f(\cdot, \cdot, Z_0)$.*

Proof. Consider any two functions $g_a, g_b: \{0, 1\}^{n+m} \rightarrow \{0, 1\}^n$.

Let $f_a: \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$ be a function where $f(\cdot, \cdot, Z_0) = g_a(\cdot, \cdot)$. Let f_b be identical to f_a except $f(\cdot, \cdot, Z_0) = g_b(\cdot, \cdot)$.

Consider any $x \in B_{(Z, Z', f_a)}$. As $\mathcal{G}$ is deterministic, the query list of $\mathcal{G}^{f_a}(x, Z'_0)$ and $\mathcal{G}^{f_b}(x, Z'_0)$ are identical as long as the responses received from f are identical. As $\mathcal{G}^f(x, Z'_0)$ does not query $f(*, *, Z_0)$, and f_a is same as f_b for all other set of inputs, for all the queries, the responses received by $\mathcal{G}^{f_a}(x, Z'_0)$ and $\mathcal{G}^{f_b}(x, Z'_0)$ are identical. We get the following claim.

Claim. An element $x \in B_{(Z, Z', f_a)}$ if and only if $x \in B_{(Z, Z', f_b)}$.

The Lemma follows. $\square$

Bad Transcript. The final ingredient is to bound the probability of an $f(*, *, Z_0)$ query output to be in $L_{(Z, Z', f)}$. For that, we define the notion of bad transcript τ_f -Bad as follows. A $f(*, *, Z_0)$ query (made by the adversary or $\mathcal{G}^f$) output y sets τ_f -Bad, if.

1. Collision in the output of $f(*, *, Z_0)$ queries.
2. Collision with the first n -many bits of the input of any existing f query in the transcript.
3. $y \in L_{(Z, Z', f)}$

Recall, $\mathbf{GF}\text{-Bad} \wedge T2$ implies there is some $f(*, *, Z_0)$ query output y in the transcript such that for some x , $\mathcal{G}^f(x, Z'_0)$ queries $f(y, Z_1)$. Further, transcript type T2 implies that $x \in B_{(Z, Z', f)}$ and $y \in L_{(Z, Z', f)}$. We get the following lemma.

Lemma 22. *It holds that*

$$\Pr[\mathbf{GF}\text{-Bad} \wedge T2] \leq \Pr[\tau_f\text{-Bad}]$$

Bounding the size of $L_{Z,Z',f}$ We say a (Z, Z', f) is i -slotted for $i \in (n]$ if

$$2^i \leq |L_{Z,Z',f}| < 2^{i+1}$$

Fix f, Z_0, Z'_0 . For each $x \in \{0, 1\}^{n+m}$, $\mathcal{G}^f(x, Z'_0)$ makes at most $q_{\mathcal{G}}$ many queries to f . Hence

$$\sum_{Z_1 \in \{0,1\}^{m+k}} |L_{Z,Z',f}| \leq q_{\mathcal{G}} 2^{n+m}$$

Taking the average over choices of Z_1 ,

$$\mathbb{E}_{Z_1} [|L_{Z,Z',f}|] \leq \frac{q_{\mathcal{G}} 2^{n+m}}{2^{m+k}} = q_{\mathcal{G}} 2^{n-k}$$

Applying Markov inequality,

$$\begin{aligned} \Pr_{Z_1} [(Z, Z', f) \text{ is } i \text{ slotted}] &= \Pr_{Z_1} [2^i \leq |L_{Z,Z',f}| < 2^{i+1}] \leq \Pr_{Z_1} [2^i \leq |L_{Z,Z',f}|] \\ &\leq \frac{\mathbb{E}_{Z_1} [|L_{Z,Z',f}|]}{2^i} \\ &\leq \frac{q_{\mathcal{G}} 2^{n-k}}{2^i} \end{aligned}$$

We conclude, when the probability over (Z, Z', f) , for every $i \in (n]$,

$$\Pr_{(Z,Z',f)} [(Z, Z', f) \text{ is } i \text{ slotted}] \leq \frac{q_{\mathcal{G}} 2^{n-k}}{2^i}$$

For any element $x \in \{0, 1\}^{n+m}$, it holds that

$$\begin{aligned} &\Pr_{(Z,Z',f)} [f_{Z_0}(x) \in L_{Z,Z',f}] \\ &= \sum_{i=0}^n \Pr_{(Z,Z',f)} [f_{Z_0}(x) \in L_{Z,Z',f} \wedge (Z, Z', f) \text{ is } i \text{ slotted}] \\ &= \sum_{i=0}^n \Pr_{f_{Z_0}} [f_{Z_0}(x) \in L_{Z,Z',f} \mid (Z, Z', f) \text{ is } i \text{ slotted}] \cdot \Pr_{(Z,Z',f)} [(Z, Z', f) \text{ is } i \text{ slotted}] \\ &\leq \sum_{i=0}^n \frac{2^{i+1}}{2^n} \cdot \frac{q_{\mathcal{G}} 2^{n-k}}{2^i} = \sum_{i=0}^n \frac{2q_{\mathcal{G}} 2^{n-k}}{2^n} = \frac{2(n+1)q_{\mathcal{G}}}{2^k} \end{aligned}$$

Bounding $\Pr[\tau_f\text{-Bad}]$ Recall that the extended transcript for adversary $\mathcal{C}$ has at most σ blocks of queries, where $\sigma = 2(q_P q_{\mathcal{G}}(\ell+1) + q_C q_{\mathcal{G}}(\ell+1)^2)$. Then we bound

$$\Pr[\tau_f\text{-Bad}] \leq \sum_{i=1}^{\sigma} \left(\frac{i-1}{2^n} + \frac{i-1}{2^n} + \frac{2(n+1)q_{\mathcal{G}}}{2^k} \right) = \frac{2\sigma^2}{2^n} + \frac{2(n+1)q_{\mathcal{G}}\sigma}{2^k}$$

Bounding Pr [GF-Bad] We bound the probability of GF-Bad as

$$\begin{aligned} \Pr [\text{GF-Bad}] &= \Pr [\text{GF-Bad} \wedge T1] + \Pr [\text{GF-Bad} \wedge T2] \\ &\leq \frac{q_G^2}{2^{2k-n}} + \frac{2\sigma^2}{2^n} + \frac{2(n+1)q_G\sigma}{2^k} \end{aligned}$$

where $\sigma = 2(q_P q_G(\ell+1) + q_C q_G(\ell+1)^2)$ is the maximum number of queries in the transcript of $\mathcal{C}$. This completes the proof of Theorem 15. $\square$

6.3 Proof of Theorem 14: Strong-model PRO security from general model PRO security.

In this section, we establish the security of our Envelope-C[H] combiner when the implementor $\mathcal{A}_1$ makes significant number of preprocessing queries before finalizing the adversarial implementation $\mathcal{G}$.

Proof. Let $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ where $\mathcal{D}_1$ is a (q_1, q_G) -implementor, and $\mathcal{D}_2$ is the indistinguishability distinguisher. The general combiner-distinguisher adversary $\mathcal{A}_1$ (who itself does not have access to f /simulator S) invokes $\mathcal{D}_1$, answers all the q_1 many queries through the algorithm sim_1 which answers each new query by sampling responses independently and uniformly at random. When $\mathcal{D}_1$ outputs $\tilde{\tau}$ and $\mathcal{G}^f_{\mathcal{D}}$, the adversary $\mathcal{A}_1$ outputs $\tilde{\tau}$ as state and produces its own adversarial implementation $\mathcal{G}^f_{\mathcal{A}}$ that also has $\tilde{\tau}$ hardwired. $\mathcal{G}^f_{\mathcal{A}}$ runs $\mathcal{G}^f_{\mathcal{D}}$ on the given input, if the query made by $\mathcal{G}^f_{\mathcal{D}}$ is in $\tilde{\tau}$, same response is given, otherwise, $\mathcal{G}^f_{\mathcal{A}}$ forwards the query to $\mathcal{A}_2$'s own primitive oracle.⁶

The distinguishing adversary $\mathcal{A}_2$ submits the implementation $\mathcal{G}^f_{\mathcal{A}}$ and receives the random salt $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1)$. Then it runs $\text{sim}_2(\tilde{\tau}, Z, Z')$. If $\tilde{\tau}$ includes any query $f(*, *, Z_0)$ or $f(*, *, Z_1)$ (Bad-Salt event), sim_2 and $\mathcal{A}_2$ halt and abort. $\mathcal{A}$ invokes $\mathcal{D}_2$ with input $(\mathcal{G}^f_{\mathcal{D}}, \tilde{\tau}, Z, Z')$. Queries made by $\mathcal{D}_2$ is handled by sim_2 as follows. If it is in $\tilde{\tau}$, same answer is given. Else, the query is forwarded to $\mathcal{A}_2$'s own oracle, and response is fed back in. So, by construction

$$\mathcal{A}_1 = \mathcal{D}_1^{\text{sim}_1} \quad \mathcal{A}_2^{\mathcal{O}_f, \mathcal{O}_C} = \mathcal{D}_2^{\text{sim}_2^{\mathcal{O}_f, \mathcal{O}_C}}$$

Suppose $(\mathcal{O}_f, \mathcal{O}_C) = (f, C_{Z, Z'}^{f, \mathcal{G}^f_{\mathcal{A}}})$. We observe that if Bad-Salt does not happen, $\mathcal{A}$ and $\mathcal{G}^f_{\mathcal{A}}$ (with the help of their own oracles) perfectly simulates the oracles for $\mathcal{D}$ and $\mathcal{G}^f_{\mathcal{D}}$. In other words,

$$\Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_{Z, Z'}^{f, \mathcal{G}^f_{\mathcal{D}}}} \right) = 1 \mid \neg \text{Bad-Salt} \right] = \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2^f, C_{Z, Z'}^{f, \mathcal{G}^f_{\mathcal{A}}}} \right) = 1 \mid \neg \text{Bad-Salt} \right]$$

Hence

$$\left| \Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_{Z, Z'}^{f, \mathcal{G}^f_{\mathcal{D}}}} \right) = 1 \right] - \Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{f, C_{Z, Z'}^{f, \mathcal{G}^f_{\mathcal{A}}}} \right) = 1 \right] \right| \leq \Pr [\text{Bad-Salt}]$$

⁶ Note, $\mathcal{D}_2$ and $\mathcal{G}^f_{\mathcal{D}}$ may indeed repeat queries from $\tilde{\tau}$ as we are considering complete adversaries.

Oracle sim-first-phase (h, x_1, x_2) // $h = n, x_1 = m, x_2 = k$ $L_0 = L_0 \cup x_2; L_1 = L_1 \cup x_1 \parallel x_2$ if $(h, x_1, y) \in \tau_f[x_2]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[x_2] = \tau_f[x_2] \cup \{(h, x_1, y)\}$ return y Oracle sim-main (h, x_1, x_2) if $x_2 = Z_0$ sim-query-Z0 (h, x_1) if $x_1 \parallel x_2 = Z_1$ sim-query-Z1 (h) if $(h, x_1, y) \in \tau_f[x_2]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[x_2] = \tau_f[x_2] \cup \{(h, x_1, y)\}$ return y Algorithm sim-query-Z0 (h, m, Z_0) if $(h, m, y) \in \tau_f[Z_0]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_0] = \tau_f[Z_0] \cup \{(h, m, y)\}$ return y Algorithm f-Z1 ($M', h, Z_1, \text{flag}_g$) if $(*, h, y, *) \in \tau_f[Z_1]$ return y $y \xleftarrow{\\$} \{0, 1\}^n$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M', h, y, \text{flag}_g)\}$ return y	Oracle sim-init-second-phase (τ_f, Z, Z') Parse $Z = (Z_0, Z_1), Z' = (Z'_0, Z'_1)$ if $Z_0 \in L_0$ or $Z_1 \in L_1$ Bad-Salt = 1; return $\perp$ Algorithm sim-query-Z1 (h) if $\text{flag}_g = 1$ // $\mathcal{G}^f$ query computing $H^{\mathcal{G}^f}(M_G)$ return $f\text{-}Z_1(M_G, h, Z_1, \text{flag}_g)$ if $\text{flag}_g = 0$ // Query from the PRO-adversary $\mathcal{A}$ $M = \text{Ex}(h, \tau_f[Z_0])$ if $M = \perp$ return $f\text{-}Z_1(M, h, Z_1, \text{flag}_g)$ if $M \neq \perp$ $\text{flag}_g = 1 \quad M_G = M$ $t = \text{Envelope-}H_{Z'}^{\mathcal{G}^{\text{sim}}}(M)$ $\text{flag}_g = 0 \quad M_G = \perp$ if $(M', h, y, 1) \in \tau_f[Z_1]$ return $\mathcal{F}(M) \oplus t$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M \neq M'$ return $\mathcal{F}(M) \oplus t$ if $(M', h, y, 0) \in \tau_f[Z_1] \wedge M = M'$ return y if $(*, h, y, *) \notin \tau_f[Z_1]$ // the remaining case $y = \mathcal{F}(M) \oplus t$ $\tau_f[Z_1] = \tau_f[Z_1] \cup \{(M, h, y, 0)\}$ return y endif // Ending $(*, h, y, *) \notin \tau_f[Z_1]$ case endif // Ending $M \neq \perp$ case endif // Ending $\text{flag}_g = 0$ case
--	--

Fig. 15: Strong Model Simulator for Envelope Combiner: first-stage simulator answers honestly. Second-stage simulator initially checks for Bad-Salt condition, and aborts if true; else runs the general model simulator with state τ_f (offline phase transcript).

When $(\mathcal{O}_f, \mathcal{O}_C) = (S, \mathcal{F})$, by construction

$$\Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{S, \mathcal{F}} \right) = 1 \right] = \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2^S, \mathcal{F}} \right) = 1 \right]$$

By triangle inequality we have

$$\begin{aligned}
& \left| \Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_{Z, Z'}^f, \mathcal{G}_{Z, Z'}^f} \right) = 1 \right] - \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2^S, \mathcal{F}} \right) = 1 \right] \right| \\
& \leq \left| \Pr \left[\left(\mathcal{D}_1^f, \mathcal{D}_2^{f, C_{Z, Z'}^f, \mathcal{G}_{Z, Z'}^f} \right) = 1 \right] - \Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{f, C_{Z, Z'}^f, \mathcal{G}_{Z, Z'}^f} \right) = 1 \right] \right| + \\
& \quad \left| \Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{f, C_{Z, Z'}^f, \mathcal{G}_{Z, Z'}^f} \right) = 1 \right] - \Pr \left[\left(\mathcal{D}_1^{\text{sim}_1}, \mathcal{D}_2^{\text{sim}_2^S, \mathcal{F}} \right) = 1 \right] \right| \\
& \leq \Pr[\text{Bad-Salt}] + \left| \Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{f, C_{Z, Z'}^f, \mathcal{G}_{Z, Z'}^f} \right) = 1 \right] - \Pr \left[\left(\mathcal{A}_1, \mathcal{A}_2^{S, \mathcal{F}} \right) = 1 \right] \right|
\end{aligned}$$

We derive

$$\Delta_{\mathcal{D}(Z, Z')}(\text{REAL}, \text{IDEAL}) \leq \Delta_{\mathcal{A}(Z, Z')}(\text{REAL}, \text{IDEAL}) + \Pr[\text{Bad-Salt}]$$

We get Theorem 14 as $\Pr[\text{Bad-Salt}] \leq \frac{q_1}{2^{k+m}} + \frac{q_1}{2^k} < \frac{2q_1}{2^k}$. $\square$

7 Rate-Preserving Combiners

7.1 General Model

In this section, we present a rate-preserving combiner secure in the general model (i.e. $q_1 = 0, q_G > 0$). In practice, there are many designs where the envelope post-processor has larger domain than the internal compression functions. Towards this, consider two FIL-RO $f_0 : \{0, 1\}^m \rightarrow \{0, 1\}^n$ and $f_1 : \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$ and the adversarially implemented primitives $\mathcal{G}_0^{f_0, f_1} : \{0, 1\}^m \rightarrow \{0, 1\}^n$ and $\mathcal{G}_1^{f_0, f_1} : \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$. While we present the result in terms of Merkle-Damgård mode, the combiner will work with any preimage-aware hash mode. Let MD denote the Merkle-Damgård hash mode. The Post-C[MD] xor combiner with salts $Z, Z' \xleftarrow{\$} \{0, 1\}^{m+k}$ is then defined as

$$\text{Post-C[MD]}_{Z, Z'}^{f_0, f_1, \mathcal{G}_0, \mathcal{G}_1}(M) = f_1 \left(\text{MD}^{f_0}(M), Z \right) \oplus \mathcal{G}_1 \left(\text{MD}^{\mathcal{G}_0}(M), Z' \right)$$

Theorem 23 (PRO security of Large Postprocessor Merkle-Damgård Combiner). *There exists a simulator sim such that any $(q_G, (q_P, q_C))$ -PRO adversary $\mathcal{A}$, it holds that*

$$\text{Adv}_{\text{Post-C[MD]}, \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \mathcal{O} \left(\frac{\sigma^2}{2^n} + \frac{q_G}{2^k} \right)$$

where $\sigma = 2(q_P q_G(\ell + 1) + q_C q_G(\ell + 1)^2)$, q_G is the number of queries made by $\mathcal{G}^f$ and $q_{\text{sim}} = q_P$.

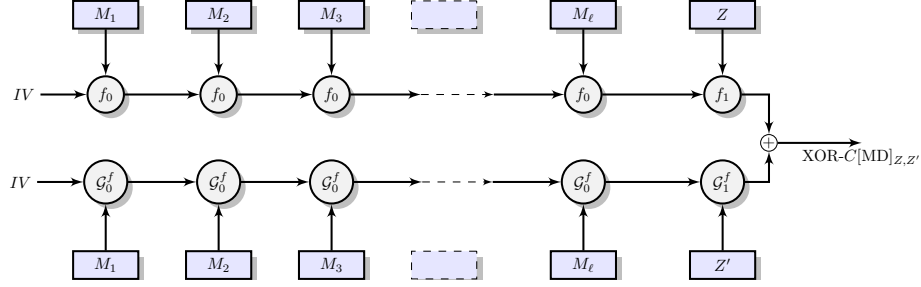


Fig. 16: Rate preserving RO-combiner XOR-C[MD]. The variant Post-C[MD] in Section 7 provides general model security with $f_0, \mathcal{G}_0 : \{0, 1\}^m \rightarrow \{0, 1\}^n$, $f_1, \mathcal{G}_1 : \{0, 1\}^{n+m+k} \rightarrow \{0, 1\}^n$ and random salts $Z, Z' \xleftarrow{\$} \{0, 1\}^{n+m+k}$. For weak-model security, weak-envelope-C combiner from Section 7.3 requires $f_0, f_1, \mathcal{G}_0, \mathcal{G}_1 : \{0, 1\}^{n+k} \rightarrow \{0, 1\}^n$ with random salts $Z, Z' \xleftarrow{\$} \{0, 1\}^k$.

7.2 Proof of Theorem 23.

In order to prove Theorem 23, following Theorem 13, it is sufficient to bound the $\text{Adv}_{\mathcal{G}^f, \text{Post-C[MD]}}^{\text{influence}}$ as the PRA advantage of any σ -query adversary against the MD^{f_0} is $\frac{\sigma^2}{2^n}$. The following lemma shows the required bound.

Lemma 24. *It holds that, for any adversary $\mathcal{C}$*

$$\text{Adv}_{\mathcal{G}^f, \text{Post-C[MD]}}^{\text{influence}}(\mathcal{C}) \leq \frac{q_{\mathcal{G}}}{2^k}$$

Proof. We will bound the probability of the event that the transcript is Bad, that is during some $\text{Post-MD}_{Z', \mathcal{G}_1}^{f_0, \mathcal{G}_1}(M')$ evaluation, the adversarial compression functions $\mathcal{G}_0$ or $\mathcal{G}_1$ query $f(h, Z)$ such that h is the output of some f query in the transcript.

We shall prove the following claim.

Claim. It holds that

$$\begin{aligned} \Pr \left[\exists x \in \{0, 1\}^m \text{ s.t. } \mathcal{G}_0^{f_0, f_1}(x) \text{ queries } f_1(*, Z) \right] &\leq \frac{q_{\mathcal{G}}}{2^k} \\ \Pr \left[\exists x \in \{0, 1\}^n \text{ s.t. } \mathcal{G}_1^{f_0, f_1}(x, Z') \text{ queries } f_1(*, Z) \right] &\leq \frac{q_{\mathcal{G}}}{2^k} \end{aligned}$$

Proof (of claim). For proof of the claim, we first fix f_0, f_1, Z' and define the query set of $\mathcal{G}_0$ as

$$Q[\mathcal{G}_0, x] := \{z \in \{0, 1\}^{m+k} \mid \mathcal{G}_0^{f_0, f_1}(x) \text{ queries } f_1(*, z)\}$$

Then we define $Q[\mathcal{G}_0] := \cup_{x \in \{0, 1\}^m} Q[\mathcal{G}_0, x]$. As $\mathcal{G}_0$ makes $q_{\mathcal{G}}$ many queries

$$|Q[\mathcal{G}_0]| \leq 2^m q_{\mathcal{G}}$$

We conclude

$$\Pr_Z [Z \in Q[\mathcal{G}_0]] \leq \frac{2^m q_{\mathcal{G}}}{2^{m+k}} = \frac{q_{\mathcal{G}}}{2^k}$$

Analogously, it holds that

$$\Pr_Z [Z \in Q[\mathcal{G}_1]] \leq \frac{2^n q_{\mathcal{G}}}{2^{m+k}} = \frac{q_{\mathcal{G}}}{2^{k+m-n}} \leq \frac{q_{\mathcal{G}}}{2^k}$$

The last inequality holds as $m > n$. $\square$

The lemma follows from the observation that conditioned on the event that for no $x \in \{0, 1\}^m$, $\mathcal{G}_0^{f_0, f_1}(x)$ and for no $y \in \{0, 1\}^n$, $\mathcal{G}_1^{f_0, f_1}(y)$ query $f_1(*, Z)$, probability of Bad is zero. Hence

$$\text{Adv}_{\mathcal{G}^f, \text{Post-}C[MD]}^{\text{influence}}(\mathcal{C}) \leq \frac{q_{\mathcal{G}}}{2^k}$$

$\square$

7.3 Rate Preserving Combiner in the Weak Model

Finally, we consider a weaker model of adversarial implementations and corresponding random oracle combiners, originally conceived in [24]. In this model, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a combiner adversary with $q_1 = \text{poly}(n)$, $q_{\mathcal{G}} = 0$. Recall, we also provide $\mathcal{A}_2$ with $\tilde{\tau}$, the transcript of the interaction of $\mathcal{A}_1$ with f as the advice string. The simulator also gets $\tilde{\tau}$ to remain consistent.

Primitives. The primitives we consider here are a FIL-RO $f: \{0, 1\}^{n+k+1} \rightarrow \{0, 1\}^n$ and an adversarially implemented function $\mathcal{G}: \{0, 1\}^{n+k+1} \rightarrow \{0, 1\}^n$. We define the compression functions $f_0, f_1: \{0, 1\}^{n+k} \rightarrow \{0, 1\}^n$ from f via domain separation, by fixing the last bit of the inputs to f ; $f_0 := f(\cdot, \cdot, 0)$ $f_1 := f(\cdot, \cdot, 1)$.

The Construction. Let $H^{\hat{f}}: \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a *preimage-aware* hashing mode for some extractor Ex using a FIL-RO compression function $\hat{f}: \{0, 1\}^{n+k} \rightarrow \{0, 1\}^n$. Let $Z \xleftarrow{\$} \{0, 1\}^k$ be a random string. We define the envelope- H mode as the composition of H^{f_0} and $f_1(\cdot, Z)$.

$$\text{Weak-Envelope-}H_Z^f(M) := f_1(H^{f_0}(M), Z).$$

The Weak-Envelope- H XOR combiner C is the xor of two Weak-Envelope- H evaluations, one through f and the other through $\mathcal{G}$.

$$C_{Z, Z'}^{f, \mathcal{G}}(M) = \text{Weak-Envelope-}H_Z^f(M) \oplus \text{Weak-Envelope-}H_{Z'}^{\mathcal{G}}(M)$$

Security Theorem 25 summarizes the security of the Weak-Envelope- H XOR combiner.

Theorem 25. *Let $\mathcal{A}$ be a $(q_1, 0)$ implementer. There exists a simulator $\text{sim} = (\text{sim}_1, \text{sim}_2)$ such that $(f, C_{Z, Z'}^{f, \mathcal{G}})$ is indistinguishable from $(\text{sim}, \mathcal{F})$ and $q_{\text{sim}} = q_P$. In other words, for every indistinguishability adversary $\mathcal{A}$ making (q_P, q_C) queries to its oracles, there exist a PRA adversary $\mathcal{B}$ such that*

$$\text{Adv}_{C, \mathcal{F}, \text{sim}}^{\text{pro}}(\mathcal{A}) \leq \frac{q_1}{2^k} + \text{Adv}_{H, \text{Ex}}^{\text{pra}}(\mathcal{B})$$

For simplicity of argument, we consider a hybrid model where every execution of $\mathcal{G}^f$ makes the primitive queries present in $\tilde{\tau}$. Clearly, the success probability of a PRO adversary $\mathcal{A}$ is no worse in this hybrid model.

As Z, Z' is sampled after $\tilde{\tau}$ is fixed, we have $\Pr[(*, Z, *) \in \tilde{\tau}] \leq \frac{|\tilde{\tau}|}{2^k} = \frac{q_1}{2^k}$. Hence, $\text{Adv}_{\mathcal{G}^f, \text{Envelope-}C[H]}^{\text{influence}} \leq \frac{q_1}{2^k}$ which, following Theorem 13, implies Theorem 25.

7.4 Proof of Theorem 25

Simulator Description The simulator sim works in two stages. At the first stage, sim_1 returns the $y \xleftarrow{\$} \{0, 1\}^n$ for the x queries of the adversary. In the second stage sim_2 gets the random salts Z, Z' and checks whether there exists any x queried to sim_1 such that $x = (\cdot, Z, 1)$. If there exist such an x it outputs $\perp$ and aborts. Otherwise, it takes the help of the extractor Ex to answer the adversary's queries. We formally describe the simulator in Figure 17

7.5 Description of Games

We define a sequence of Games through which we go from the Real World to the Ideal World. Game **REAL** describes the real world where the adversary interacts with f and $C_{Z, Z'}^{f, \mathcal{G}}$. Game **IDEAL** describes the ideal world where the adversary $\mathcal{A}$ interacts with a random oracle F and the simulator sim . Game G_2 describes the intermediate world, where we introduce a PRA adversary $\mathcal{B}$ for H^{f_0} for any PRO adversary $\mathcal{A}$ of $C_{Z, Z'}^{f, \mathcal{G}}$. $\mathcal{B}$ makes all the queries that $\mathcal{A}$ makes, but in the second stage, if it gets a query $x = (\cdot, Z, 1)$ to its P oracle instead of calling f , it simulates the response by itself through calling its extractor oracle (The detailed description is given in Figure 19). At the end of all the queries made by $\mathcal{A}$ for all M queries to C , it computes all the intermediate values and then checks if $\exists M$ s.t $H^{P(M)} = x$, $x = (\cdot, Z, 1)$ was queried by $\mathcal{A}$ to C , and when $\mathcal{B}$ called Ex with input x , and the relevant transcript, it returned $M' \neq M$. If there exists such an M , it outputs the first such M . It is obvious that in this case, $\mathcal{B}$ wins the PRA Game against H^{f_0} .

7.6 Transcript Description

At the end of its query response phase, the adversary $\mathcal{A}$ submits all the queries made to the C oracle. For these M , all the intermediate P computations are done and made public to the $\mathcal{A}$. We define the whole transcript as $\Gamma = (\Gamma_P, \Gamma_C)$.

7.7 Bad Events

We list two bad events in the game G_2

1. Bad_τ : There exist x_i queried in the first stage such that $x_i = (\cdot, Z, 1)$
2. Bad_{PRA} : There exists M, M' such that $H^P(M) = x_j$ where x_j was the query for which M' was returned, and $M \neq M'$.

Preprocessing Stage	Online Stage
$\text{sim}_1\text{-main}(h, x, b)$	$\text{sim}_2\text{-init}(\tau_1, Z, Z')$
$\text{// Simulator has no access to } \mathcal{F}$ $\text{// } h \in \{0, 1\}^n$ $\text{// } x \in \{0, 1\}^k, b \in \{0, 1\}$ if $(h, x, y) \in \tau_1[b]$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_1[b] = \tau_1[b] \cup \{(h, x, y)\}$ return y	$\text{// Simulator has access to } \mathcal{G}, \mathcal{F}$ if $(*, Z, y) \in \tau_1[1]$ return $\perp$ $\tau_2 = \emptyset \quad \tau_f = \tau_1 \cup \tau_2$ $L = \emptyset$ $\text{sim}_2\text{-main}(h, x, b)$ if $(x, b) = (Z, 1)$ $M = \text{Ex}(h, \tau[0])$ if $(M', h, x, y) \in L$ if $M \neq M' \quad \text{Bad}_{\text{PRA}} = 1$ return y endif if $M \neq \perp$ $t = \text{Envelope-}H_{Z'}^{\mathcal{G}}(M)$ $y = \mathcal{F}(M) \oplus t$ else $\text{// } M = \perp$ $y \xleftarrow{\$} \{0, 1\}^n$ endif $\tau_2[1] = \tau_2[1] \cup \{(h, x, y)\}$ $L = L \cup (M, h, x, y)$ else $\text{// } b = 0 \vee x \neq z$ if $(h, x, y) \in \tau_f[b]$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_2[b] = \tau_2[b] \cup \{(h, x, y)\}$ endif return y

Fig. 17: The Simulator for Envelope Combiner in the Weak Model: Initialized with Z, Z' .

Game REAL, G0	Online Stage
<u>Preprocessing Stage</u>	<u>game-init(τ_1, Z, Z')</u>
<u>$O_f(h, x, b)$</u>	$\tau_2 = \emptyset \quad \tau_f = \tau_1 \cup \tau_2$ $L = \emptyset$
$\text{// } h \in \{0, 1\}^n$ $\text{// } x \in \{0, 1\}^k, b \in \{0, 1\}$ if $(h, x, y) \in \tau_1[b]$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_1[b] = \tau_1[b] \cup \{(h, x, y)\}$ return y	<u>$C_{Z, Z'}(M)$</u> $h = \text{MD}^{f_0}(M)$ $h' = \text{MD}^{\mathcal{G}}(M)$ $t = \mathcal{G}(h', Z', 1)$ if $(h, Z, y) \in \tau_f[1]$ return $y \oplus t$ $y = f(h, Z, 1)$ return $y \oplus t$
	<u>$f(h, x, b)$</u> if $(h, x, y) \in \tau_f[b]$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_2[b] = \tau_2[b] \cup \{(h, x, y)\}$ return y

Fig. 18: Oracles and Subroutines in the REAL game in the Weak Model: Initialized with $\tau_1 = \tau_2 = \emptyset, \tau_f = \tau_1 \cup \tau_2, Z, Z'$.

Game G2, G3	
Preprocessing Stage	$f(h, x, b)$ $\text{// } h \in \{0, 1\}^n \text{ // } x \in \{0, 1\}^k, b \in \{0, 1\}$ if $(*, h, x, y) \in \tau_1[b]$ return y $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_1[b] = \tau_1[b] \cup \{(\perp, h, x, y)\}$ return y
Online Stage	
$\text{game-init}(\tau_1, Z, Z')$ $\tau_2 = \emptyset \quad \tau_f = \tau_1 \cup \tau_2$ $L = \emptyset$ $C_{Z, Z'}(M)$ $h = \text{MD}^{f_0}(M)$ $t = \text{Envelope-}H_{Z'}^G(M)$ if $(M', h, Z, y) \in \tau_f[1] \wedge M = M'$ return $y \oplus t$ if $(M', h, Z, y) \in \tau_f[1] \wedge M \neq M'$ $\text{Bad}_{PRA} = 1$ return $\mathcal{F}(M)$ return $y \oplus t$ $y = f(h, Z, 1)$ return $y \oplus t$	$f(h, x, b)$ if $(x, b) = (Z, 1)$ if $(*, *, Z, y) \in \tau_1[1]$ $\text{Bad}_\tau = 1$ return $\perp$ return y $M = \text{Ex}(h, \tau[0])$ if $(M', h, x, y) \in \tau_f[1]$ if $M \neq M' \quad \text{Bad}_{PRA} = 1$ return y endif if $M \neq \perp$ $t = \text{Envelope-}H_{Z'}^G(M)$ $y = \mathcal{F}(M) \oplus t$ else $\text{// } M = \perp$ $y \xleftarrow{\$} \{0, 1\}^n$ endif $\tau_2[1] = \tau_2[1] \cup \{(M, h, x, y)\}$ else $\text{// } b = 0 \vee x \neq Z$ if $(h, x, y) \in \tau_f[b] \quad \text{return } y$ $y \xleftarrow{\$} \{0, 1\}^n$ $\tau_2[b] = \tau_2[b] \cup \{(\perp, h, x, y)\}$ endif return y

Fig. 19: Algorithms in the G2 game in the Weak Model: Initialized with $\tau_1 = \tau_2 = \emptyset, \tau_f = \tau_1 \cup \tau_2, Z, Z'$.

7.8 Bounding The Probability of Bad Events:

Proposition 26. $\Pr_{G_2}[\text{Bad}_\tau] \leq \frac{q_1}{2^k}$

Proof. There are at most q_1 queries made in the first stage. As Z is sampled uniformly at random from $\{0, 1\}^k$, the probability that one of those queries was of the form $(\cdot, Z, 1)$ is bounded by $\frac{q_1}{2^k}$.

Proposition 27. *For any adversary $\mathcal{A}$ of the PRO game of (C^f, f) making q_P primitive and q_C construction queries, there exists an adversary $\mathcal{B}$ of the PRA game of H^f s.t $\Pr_{G_2}[\text{Bad}_{PRA}] \leq \text{Adv}_{H^f, \text{Ex}}^{\text{pra}}(\mathcal{B})$. $\mathcal{B}$ makes at most $q_1 + q_P + q_C \cdot l_{\max}$ primitive queries, and at most q_C extraction queries.*

Proof. For any adversary $\mathcal{A}$ making at most q_P query to f in the second stage, $\mathcal{B}$ makes at most $q_1 + q_P + q_C \cdot l_{\max}$ queries to f . As $\mathcal{A}$ makes at most q_C construction queries $\mathcal{B}$ makes q_C extraction queries. The proposition follows from the definition of the PRA game.

7.9 Ratio of Good Transcripts

Lemma 28. *Let G_2 be defined as in Figure 19. It holds that*

$$\Delta_{\mathcal{A}(Z, Z')}(\text{REAL}, G_2) = 0 \quad (3)$$

$$\Delta_{\mathcal{A}(Z, Z')}(\text{IDEAL}, G_2) \leq \Pr[\text{Bad}_\tau] + \Pr[\text{Bad}_{PRA}] \quad (4)$$

The proof of this lemma is identical to the proof in Section 6.1. We skip the proof.

From Lemma 28, by triangle inequality we get

$$\text{Adv}_{C, \mathcal{F}, S}^{\text{pro}}(\mathcal{A}) < \Pr[\text{Bad}_\tau] + \Pr[\text{Bad}_{PRA}]$$

This completes the Proof of Theorem 25. $\square$

8 Conclusion

In Table 1, we compare the security bounds for these constructions as well as the number of queries needed for the attacks mentioned. Note, we do not have anything better than the online-only attack against Envelope- $C[H]$ even when preprocessing queries are allowed.

We conclude with potential future research directions:

- Does there exist an efficient RO-Combiner for variable-length input using compression functions of the type $\{0, 1\}^{n+\mathcal{O}(1)} \rightarrow \{0, 1\}^n$ and $\mathcal{O}(n)$ salt? As most popular hash modes use at least $2n$ -bit-to- n -bit compression functions, we consider this is mainly for theoretical interest.
- While we present matching attacks to the presented combiners for certain parameters, the bounds may not be tight for *all* parameters. Combiner designs achieving optimality across all the parameters remain an interesting open problem.

Construction	Security Bound	Attack Query Complexity with preprocessing	Attack Query Complexity without preprocessing
XOR-MD [25]	$\frac{\mathcal{O}(q_G q_P l + q_C l^2)^2}{2^n}$	$2^{n/2}$ preprocessing query, $\mathcal{O}(1)$ online query	$2^{n/2}$ online queries
Envelope- $C[H]$ (This Paper)	$\frac{\mathcal{O}(q_G q_P l + q_C l^2)^2}{2^n} + \frac{q_1}{2^k} + \frac{q_G^2}{2^{2k-n}}$	??	$2^{n/2}$ online queries
Post-C[MD] (This Paper)	$\frac{\mathcal{O}(q_G q_P l + q_C l^2)^2}{2^n}$	$2^{n/2}$ preprocessing query, $\mathcal{O}(1)$ online query	$2^{n/2}$ online queries
Envelope-MD Combiner for weak model (This Paper)	$\frac{\mathcal{O}(q_1 + q_P + q_C l)^2}{2^n}$	$2^{n/2}$ online queries	$2^{n/2}$ online queries

Table 1: Table comparing security bounds and attacks for combiners

- The constructions from [24, 25] as well as this paper are applicable for hash modes built upon compression functions. In practice, there exist hash functions like SHA3 [2], which are built upon permutations. The main open challenge remaining is designing a random oracle combiner using hash modes built upon permutations. As the adversarial implementation could make inverse queries to the independent permutation, a new technique might be required to prove security in this case.

Acknowledgements. We thank the reviewers of this and the earlier versions of the work for their insightful suggestions. RB acknowledges the support from UKRI via EPSRC grant EP/Y001680/1. At the time of this work, AR was affiliated to the University of Birmingham, supported by EPSRC grant EP/Y001680/1. MN is supported by MATRICS scheme MTR/2023/000302 of Science and Engineering Research Board, India.

References

1. Specifications for the secure hash standard. Federal Information Processing Standards Publication FIPS PUB 180-2, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA (Aug 2002), available at <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-2.pdf>
2. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication FIPS PUB 202, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA (Aug 2015), available at <https://doi.org/10.6028/NIST.FIPS.202>
3. Akshima, Cash, D., Drucker, A., Wee, H.: Time-space tradeoffs and short collisions in merkle-damgård hash functions. In: Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara,

- CA, USA, August 17-21, 2020, Proceedings, Part I. pp. 157–186 (2020), https://doi.org/10.1007/978-3-030-56784-2_6
4. Andreeva, E., Luykx, A., Mennink, B.: Provable security of BLAKE with non-ideal compression function. In: Knudsen, L.R., Wu, H. (eds.) *Selected Areas in Cryptography*, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. *Lecture Notes in Computer Science*, vol. 7707, pp. 321–338. Springer (2012), https://doi.org/10.1007/978-3-642-35999-6_21
 5. Andreeva, E., Mennink, B., Preneel, B.: On the indistinguishability of the grøstl hash function. In: Garay, J.A., Prisco, R.D. (eds.) *Security and Cryptography for Networks*, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. *Proceedings. Lecture Notes in Computer Science*, vol. 6280, pp. 88–105. Springer (2010), https://doi.org/10.1007/978-3-642-15317-4_7
 6. Asmuth, C.A., Blakley, G.R.: An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers & Mathematics with Applications* **7**(6), 447–450 (1981)
 7. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: Blake: Submission to the NIST SHA-3 competition. Tech. rep., NIST SHA-3 Competition (2008), <https://www.aumasson.jp/blake/>
 8. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. *Cryptology ePrint Archive*, Paper 2006/399 (2006), <https://eprint.iacr.org/2006/399>
 9. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73 (1993), <https://doi.org/10.1145/168588.168596>
 10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.: *Sponge functions*. *ECRYPT Hash Workshop 2007* (01 2007)
 11. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indistinguishability of the sponge construction. In: *Advances in Cryptology - EUROCRYPT 2008*, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. *Proceedings*. pp. 181–197 (2008), https://doi.org/10.1007/978-3-540-78967-3_11
 12. Bhattacharyya, R., Mandal, A.: On the indistinguishability of fugue and luffa. In: López, J., Tsudik, G. (eds.) *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings. Lecture Notes in Computer Science*, vol. 6715, pp. 479–497 (2011). https://doi.org/10.1007/978-3-642-21554-4_28, https://doi.org/10.1007/978-3-642-21554-4_28
 13. Bhattacharyya, R., Mandal, A., Nandi, M.: Indistinguishability characterization of hash functions and optimal bounds of popular domain extensions. In: *Progress in Cryptology - INDOCRYPT 2009*, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. *Proceedings*. pp. 199–218 (2009), https://doi.org/10.1007/978-3-642-10628-6_14
 14. Bhattacharyya, R., Mandal, A., Nandi, M.: Security analysis of the mode of JH hash function. In: Hong, S., Iwata, T. (eds.) *Fast Software Encryption*, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers. *Lecture Notes in Computer Science*, vol. 6147, pp. 168–191. Springer (2010), https://doi.org/10.1007/978-3-642-13858-4_10

15. Bhattacharyya, R., Nandi, M., Raychaudhuri, A.: Crooked indistinguishability of enveloped XOR revisited. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13143, pp. 73–92. Springer (2021). https://doi.org/10.1007/978-3-030-92518-5_4, https://doi.org/10.1007/978-3-030-92518-5_4
16. Bhattacharyya, R., Nandi, M., Raychaudhuri, A.: Subversion resilient hashing: Efficient constructions and modular proofs for crooked indistinguishability. IEEE Trans. Inf. Theory **69**(5), 3302–3315 (2023), <https://doi.org/10.1109/TIT.2023.3238115>
17. Biham, E., Dunkelman, O.: A framework for iterative hash functions - HAIFA. IACR Cryptol. ePrint Arch. p. 278 (2007), <http://eprint.iacr.org/2007/278>
18. Boneh, D., Boyen, X.: On the impossibility of efficiently combining collision resistant hash functions. In: Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings. pp. 570–583 (2006), https://doi.org/10.1007/11818175_34
19. Canetti, R., Rivest, R.L., Sudan, M., Trevisan, L., Vadhan, S.P., Wee, H.: Amplifying collision resistance: A complexity-theoretic treatment. In: Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. pp. 264–283 (2007), https://doi.org/10.1007/978-3-540-74143-5_15
20. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.P.: Random oracles and non-uniformity. In: Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I. pp. 227–258 (2018), https://doi.org/10.1007/978-3-319-78381-9_9
21. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. pp. 430–448 (2005), https://doi.org/10.1007/11535218_26
22. Daemen, J., Mennink, B., Assche, G.V.: Sound hashing modes of arbitrary functions, permutations, and block ciphers. IACR Trans. Symmetric Cryptol. **2018**(4), 197–228 (2018), <https://doi.org/10.13154/tosc.v2018.i4.197-228>
23. Damgård, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer (1990). https://doi.org/10.1007/0-387-34805-0_39
24. Dodis, Y., Ferguson, N., Goldin, E., Hall, P., Pietrzak, K.: Random oracle combiners: Breaking the concatenation barrier for collision-resistance. In: Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II. pp. 514–546 (2023), https://doi.org/10.1007/978-3-031-38545-2_17
25. Dodis, Y., Goldin, E., Hall, P.: Random oracle combiners: Merkle-damgård style. In: Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I. pp. 274–302 (2025), https://doi.org/10.1007/978-3-031-91107-1_10
26. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic

- Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. pp. 473–495 (2017), https://doi.org/10.1007/978-3-319-56614-6_16
27. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In: Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22–25, 2009, Revised Selected Papers. pp. 104–121 (2009), https://doi.org/10.1007/978-3-642-03317-9_7
 28. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26–30, 2009. Proceedings. pp. 371–388 (2009), https://doi.org/10.1007/978-3-642-01001-9_22
 29. Durrett, R.: Probability: Theory and Examples. Cambridge University Press, 5th edn. (2019)
 30. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust multi-property combiners for hash functions. *J. Cryptol.* **27**(3), 397–428 (2014), <https://doi.org/10.1007/s00145-013-9148-7>
 31. Ghoshal, A., Komargodski, I.: On time-space tradeoffs for bounded-length collisions in merkle-damgård hashing. In: Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part III. pp. 161–191 (2022), https://doi.org/10.1007/978-3-031-15982-4_6
 32. Ghoshal, A., Komargodski, I., Segev, G.: Offline-online indifferentiability of cryptographic systems. In: Theory of Cryptography - 23rd International Conference, TCC 2025, Aarhus, Denmark, December 1–5, 2025, Proceedings, Part II. pp. 3–33 (2025), https://doi.org/10.1007/978-3-032-12293-3_1
 33. Ghoshal, A., Tessaro, S.: The query-complexity of preprocessing attacks. In: Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part II. pp. 482–513 (2023), https://doi.org/10.1007/978-3-031-38545-2_16
 34. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings. pp. 96–113 (2005), https://doi.org/10.1007/11426639_6
 35. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. pp. 96–113. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
 36. Herzberg, A.: On tolerant cryptographic constructions. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. pp. 172–190. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
 37. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19–21, 2004, Proceedings. pp. 21–39 (2004), https://doi.org/10.1007/978-3-540-24638-1_2
 38. Meier, R., Przydatek, B.: On robust combiners for private information retrieval and other primitives. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO

- 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4117, pp. 555–569. Springer (2006), https://doi.org/10.1007/11818175_33
39. Mennink, B., Preneel, B.: Breaking and fixing cryptophia’s short combiner. In: Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings. pp. 50–63 (2014), https://doi.org/10.1007/978-3-319-12280-9_4
 40. Merkle, R.C.: Secrecy, Authentication, and Public Key Systems. Ph.D. thesis, Stanford University (1979), available at <https://www.merkle.com/papers/Thesis1979.pdf>
 41. Merkle, R.C.: A certified digital signature. Communications of the ACM **33**(4), 40–46 (1989). <https://doi.org/10.1145/65346.65356>
 42. Mittelbach, A.: Cryptophia’s short combiner for collision-resistant hash functions. In: Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings. pp. 136–153 (2013), https://doi.org/10.1007/978-3-642-38980-1_9
 43. O’Connor, J., Aumasson, J.P., Neves, S., Wilcox-O’Hearn, Z.: Blake3 — one function, fast everywhere: Specifications, analysis, and design rationale. Specification Technical report, BLAKE3 Team (2020), <https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf>
 44. Pietrzak, K.: Non-trivial black-box combiners for collision-resistant hash-functions don’t exist. In: Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings. pp. 23–33 (2007), https://doi.org/10.1007/978-3-540-72540-4_2
 45. Pietrzak, K.: Compression from collisions, or why CRHF combiners have a long output. In: Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. pp. 413–432 (2008), https://doi.org/10.1007/978-3-540-85174-5_23
 46. Russell, A., Tang, Q., Yung, M., Zhou, H.: Correcting subverted random oracles. In: Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. pp. 241–271 (2018), https://doi.org/10.1007/978-3-319-96881-0_9
 47. Saarinen, M.J.O.: The blake2 cryptographic hash and message authentication code (mac). RFC 7693 (Nov 2015). <https://doi.org/10.17487/RFC7693>, <https://www.rfc-editor.org/rfc/rfc7693>, proposed Standard
 48. Unruh, D.: Random oracles and auxiliary input. In: Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. pp. 205–223 (2007), https://doi.org/10.1007/978-3-540-74143-5_12