# Compactness of Hashing Modes and Efficiency Beyond Merkle Tree

Elena Andreeva[1](✉), Rishiraj Bhattacharyya[2], and Arnab Roy[3]

[1] Technical University of Vienna, Vienna, Austria
elena.andreeva@tuwien.ac.at
[2] NISER, HBNI, Jatani, India
rishirajbhattacharyya@protonmail.com
[3] University of Klagenfurt, Klagenfurt, Austria
arnab.roy@aau.at

**Abstract.** We revisit the classical problem of designing optimally efficient cryptographically secure hash functions. Hash functions are traditionally designed via applying modes of operation on primitives with smaller domains. The results of Shrimpton and Stam (ICALP 2008), Rogaway and Steinberger (CRYPTO 2008), and Mennink and Preneel (CRYPTO 2012) show how to achieve optimally efficient designs of $2n$-to-$n$-bit compression functions from non-compressing primitives with asymptotically optimal $2^{n/2-\epsilon}$-query collision resistance. Designing optimally efficient and secure hash functions for larger domains ($>2n$ bits) is still an open problem.

To enable efficiency analysis and comparison across hash functions built from primitives of different domain sizes, in this work we propose the new *compactness* efficiency notion. It allows us to focus on asymptotically optimally collision resistant hash function and normalize their parameters based on Stam's bound from CRYPTO 2008 to obtain maximal efficiency.

We then present two tree-based modes of operation as a design principle for compact, large domain, fixed-input-length hash functions.

1. Our first construction is an Augmented Binary Tree (ABR) mode. The design is a $(2^\ell + 2^{\ell-1} - 1)n$-to-$n$-bit hash function making a total of $(2^\ell - 1)$ calls to $2n$-to-$n$-bit compression functions for any $\ell \geq 2$. Our construction is optimally compact with asymptotically (optimal) $2^{n/2-\epsilon}$-query collision resistance in the ideal model. For a tree of height $\ell$, in comparison with Merkle tree, the ABR mode processes additional $(2^{\ell-1} - 1)$ data blocks making the same number of internal compression function calls.
2. With our second design we focus our attention on the indifferentiability security notion. While the ABR mode achieves collision resistance, it fails to achieve indifferentiability from a random oracle within $2^{n/3}$ queries. ABR$^+$ compresses only 1 less data block than ABR with the same number of compression calls and achieves in addition indifferentiability up to $2^{n/2-\epsilon}$ queries.

Both of our designs are closely related to the ubiquitous Merkle Trees and have the potential for real-world applicability where the speed of hashing is of primary interest.

A. Canteaut and F.-X. Standaert (Eds.): EUROCRYPT 2021, LNCS 12697, pp. 92–123, 2021.
https://doi.org/10.1007/978-3-030-77886-6_4

## 1 Introduction

Hash functions are fundamental cryptographic building blocks. The art of designing a secure and efficient hash function is a classical problem in cryptography. Traditionally, one designs a hash function in two steps. In the first, one constructs a *compression function* that maps fixed length inputs to fixed and usually smaller length outputs. In the second step, a *domain extending* algorithm is designed that allows longer messages to be mapped to a fixed-length output via a sequence of calls to the underlying compression functions.

Most commonly compression functions are designed based on block ciphers and permutations [10,13–15,32,34]. For a long time block ciphers were the most popular primitives to build a compression function and the classical constructions of MD5 and SHA1, SHA2 hash functions are prominent examples of that approach. In the light of the SHA3 competition, the focus has shifted to permutation [12] or fixed-key blockcipher-based [3,4] compression functions. Classical examples of domain extending algorithms are the Merkle–Damgård [21,28] (MD) domain extender and the Merkle tree [27] which underpins numerous cryptographic applications. Most recently, the Sponge construction [11] that is used in SHA-3 has come forward as a domain extender [5,13,16,33] method for designs which directly call a permutation.

Efficiency of Hash Design: Lower Bounds. Like in all cryptographic primitives, the design of a hash function is a trade-off between efficiency and security. Black, Cochran, and Shrimpton [14] were the first to formally analyze the security-efficiency trade-off of compression functions, showing that a $2n$-to-$n$-bit compression function making a single call to a fixed-key $n$-bit block cipher can not achieve collision resistance. Rogaway and Steinberger [35] generalized the result to show that any $mn$-to-$ln$ bit compression function making $r$ calls to $n$-bit permutations is susceptible to a collision attack in $(2^n)^{1-\frac{m-l/2}{r}}$ queries, provided the constructed compression function satisfies a "collision-uniformity" condition. Stam [37] refined this result to general hash function constructions and conjectured: if any $m+s$-to-$s$-bit hash function is designed using $r$ many $n+c$-to-$n$-bit compression functions, a collision on the hash function can be found in $2^{\frac{nr+cr-m}{r+1}}$ queries. This bound is known as the Stam's bound and it was later proven in two works by Steinberger [38] and by Steinberger, Sun and Yang [39].

Efficiency of Hash Design: Upper Bounds. The upper bound results matching Stam's bound focused on $2n$-to-$n$-bit constructions from $n$-bit non-compressing primitives. In [36], Shrimpton and Stam showed a (Shrimpton-Stam) construction based on three $n$-to-$n$-bit functions achieving asymptotically birthday bound collision resistance in the random oracle model. Rogaway and Steinberger [34] showed hash constructions using three $n$-bit permutations matching the bound of [35] and assuming the "uniformity condition" on the resulting hash construction. In [25], Mennink and Preneel generalized these results and identified four equivalence classes of $2n$-to-$n$-bit compression functions from $n$-bit permutations and XOR operations, achieving collision security of the birthday bound asymptotically in the random permutation model.

In comparison, upper bound results for larger domain compressing functions have been scarce. The only positive result we are aware of is by Mennink and Preneel [26]. In [26], the authors considered generalizing the Shrimpton-Stam construction to get $m+n$-to-$n$-bit hash function from $n$-bit primitives for $m > n$, and showed $n/3$-bit collision security in the random oracle model. For all practical purposes the following question remains open.

*If an $m$+$n$-to-$n$-bit hash function is designed using $r$ many $n$+$c$-to-$n$-bit compression functions, is there a construction with collision security matching Stam's bound when $m > n$?*

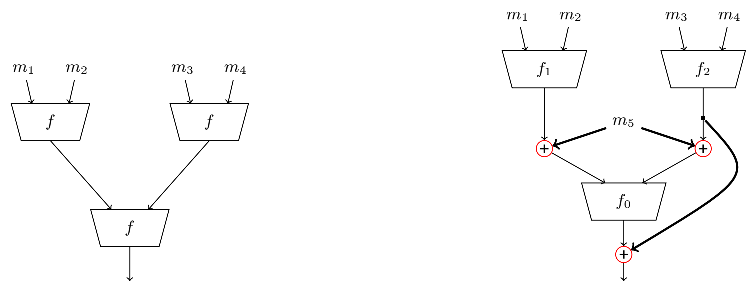
<u>Beyond Collision Resistance: Indifferentiability.</u> *Collision resistance* is undoubtedly the most commonly mandated security property for a cryptographic hash function. Naturally, all the hash function design principles and respective efficiencies are primarily targeting to achieve collision resistance. More recently, for applications of hash functions as replacement of random oracles in higher-level cryptographic schemes or protocols, the notion of indifferentiability has also gained considerable traction. The strong notion of indifferentiability from a random oracle (RO) by Maurer, Renner and Holenstein [24] has been adopted to prove the security of hash functions when the internal primitives (compression functions, permutations etc.) are assumed to be ideal (random oracle, random permutation, etc.). An important advantage of the indifferentiability from a random oracle notion is that it implies multiple security notions (in fact, all the notions satisfied by a random oracle in a single stage game) simultaneously up to the proven indifferentiability bound. The question of designing an optimally efficient hash function naturally gets extended also to the indifferentiability setting.
*If an $m$+$n$-to-$n$-bit hash function is designed using $r$ many $n$+$c$-to-$n$-bit compression functions, is there a construction with indifferentiability security matching Stam's bound when $m > n$?* Note that, a collision secure hash function matching Stam's bound may not imply the indifferentiability notion up to the same bound.

## 1.1 Our Results

<u>New measure of efficiency.</u> Comparing efficiency of hash functions built from primitives of different domain sizes is a tricky task. In addition to the message size and the number of calls to underlying primitives, one needs to take into account the domain and co-domain/range sizes of the underlying primitives. It is not obvious how to scale the notion of rate up to capture these additional parameters.

We approach the efficiency measure question from Stam's bound perspective. We say an $m+s$-to-$s$-bit hash function construction designed using $r$ many $n+c$-to-$n$-bit compression functions is optimally efficient if Stam's bound is tight, that is one can prove that asymptotically at least $2^{\frac{nr+cr-m}{r+1}}$ queries are required to find a collision. Notice that the value in itself can be low (say $2^{s/4}$), but given the proof, we can argue that the parameters are *optimal* for that security level.

Given that the collision-resistance requirement for a hash function is given by the birthday bound ($2^{s/2}$ queries), we can say that a hash function construction achieves optimal security-efficiency trade-off if $\frac{nr+cr-m}{r+1} = \frac{s}{2}$ and Stam's bound is asymptotically tight. Then one can focus on schemes which achieve the asymptotically optimal collision security, and normalize the efficiency of the construction. We hence propose the notion of *compactness* as the ratio of the parameter $m$ and its optimal value $(\frac{2nr+2cr-sr-s}{2})$ as an efficiency measure of a hash function construction $C$. In Sect. 3 we formally define the notion and derive compactness of some popular modes.



(a) Merkle tree hashing 4 input messages

(b) ABR mode hashing 5 input messages

**Fig. 1.** Merkle Tree and ABR mode for height $\ell = 2$

<u>Optimally Compact ABR Mode.</u> We present a new tree-based mode ABR. ABR of height $\ell$ implements a $(2^\ell + 2^{\ell-1} - 1)n$-to-$n$-bit function making only $(2^\ell - 1)$ calls to the underlying $2n$-to-$n$-bit compressing primitives. Assuming the underlying primitives to be independent random oracles, we show that the ABR mode is collision resistant up to the birthday bound asymptotically. The parameters of ABR mode achieve maximum compactness. In Sect. 4 we formally present the ABR mode and prove its collision resistance.

A natural comparison with Merkle tree is in order. We show that Merkle Tree can achieve only 2/3 of the optimal compactness and thus our mode is significantly more efficient. For a tree of height $\ell$, in comparison to the Merkle tree, the ABR mode can process an additional $(2^{\ell-1} - 1)$ message blocks with the same number of calls to the underlying compression functions.

<u>ABR does not satisfy Indifferentiability.</u> Our next target is to consider the notion of indifferentiability. Specifically, how does the ABR compression score in the indifferentiability setting? The primary objective of this question is twofold. If we can prove the ABR construction with height $\ell = 2$ to be indifferentiable from a random oracle up to the birthday bound, then we could use the indifferentiability composition theorem and replace the leaf level compression function of ABR by $5n$-to-$n$-bit ideal compression function. Then by recursively applying the proof of collision resistency of ABR with height $\ell = 2$, we could

extend the collision resistance proof to arbitrary large levels. Secondly, the proof of indifferentiability implies simultaneously all the security notions satisfied by a random oracle in single stage games. Unfortunately, we show that the ABR mode with height $\ell = 2$ does not preserve indifferentiability. We show an indifferentiability attack of order $2^{\frac{n}{3}}$ in Sect. 5. The attack can easily be generalized to ABR of arbitrary levels.

SALVAGING INDIFFERENTIABILITY. Next, in Sect. 5.2 we propose an almost optimally compact $\mathsf{ABR}^+$ mode design which salvages the indifferentiability security (up to birthday bound) of the original ABR mode. In principle, our second construction $\mathsf{ABR}^+$ (see Fig. 4a) tree merges two left and right ABR mode (of possibly different heights) calls by an independent post-precessor. Using the H-coefficient technique, we prove the indifferentiability of the $\mathsf{ABR}^+$ construction up to the birthday bound.

Compared to ABR mode, $\mathsf{ABR}^+$ compresses 1 less message block for the same number of calls. For large size messages, this gap is extremely small. In comparison to the Merkle Tree, the $\mathsf{ABR}^+$ mode, improves the efficiency significantly and still maintains the indifferentiability property.

## 1.2 Impact of Our Result

Merkle trees were first published in 1980 by Ralph Merkle [27] as a way to authenticate large public files. Nowadays, Merkle trees find ubiquitous applications in cryptography, from parallel hashing, integrity checks of large files, long-term storage, signature schemes [8,9,18,19], time-stamping [23], zero-knowledge proof based protocols [7,22], to anonymous cryptocurrencies [6], among many others. Despite their indisputable practical relevance, for 40 years we have seen little research go into the rigorous investigation of how to optimize their efficiency, and hence we still rely on design principles that may in fact have some room for efficiency optimizations.

In view of the wide spread use of Merkle trees, we consider one of the main advantage of our construction as being in: *increased number of message inputs (compared to the classical Merkle tree) while maintaining the same tree height and computational cost (for both root computation and node authentication).* Our trees then offer more efficient alternatives to Merkle trees in scenarios where the performance criteria is *the number of messages hashed* for: 1. a fixed computational cost – compression function calls to compute the root, or/and 2. fixed authentication cost – compression function calls to authenticate a node.

Regular hashing is naturally one of the first candidates for such an applications. Other potential use cases are hashing on parallel processors or multicore machines, such as authenticating software updates, image files or videos; integrity checks of large files systems, long term archiving [17], memory authentication, content distribution, torrent systems [1], etc. A recent application that can benefit from our ABR or $\mathsf{ABR}^+$ mode designs are (anonymous) cryptocurrency applications. We elaborate more on these in Sect. 6.

## 2 Notation and Preliminaries

Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers and $\{0, 1\}^*$ be the set of all bit strings. If $k \in \mathbb{N}$, then $\{0, 1\}^k$ denotes the set of all $k$-bit strings. The empty string is denoted by $\varepsilon$. $[n]$ denotes the set $\{0, 1, \cdots, n-1\}$. $f : [r] \times \mathsf{Dom} \to \mathsf{Rng}$ denotes a family of $r$ many functions from $\mathsf{Dom}$ to $\mathsf{Rng}$. **We often use the shorthand $f$ to denote the family $\{f_0, \cdots, f_{r-1}\}$ when the function family is given as oracles**.

If $S$ is a set, then $x \xleftarrow{\$} S$ denotes the uniformly random selection of an element from $S$. We let $y \leftarrow \mathsf{A}(x)$ and $y \xleftarrow{\$} \mathsf{A}(x)$ be the assignment to $y$ of the output of a deterministic and randomized algorithm $\mathsf{A}$, respectively, when run on input $x$.

An *adversary* $\mathsf{A}$ is an algorithm possibly with access to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_\ell$ denoted by $\mathsf{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_\ell}$. The adversaries considered in this paper are computationally unbounded. The complexities of these algorithms are measured solely on the number of queries they make. Adversarial queries and the corresponding responses are stored in a transcript $\tau$.

**Hash Functions and Domain Extensions.** In this paper, we consider Fixed-Input-Length (FIL) hash functions. We denote these by the hash function $H : \mathcal{M} \to \mathcal{Y}$ where $\mathcal{Y}$ and $\mathcal{M}$ are finite sets of bit strings. For a FIL $H$ the domain $\mathcal{M} = \{0, 1\}^N$ is a finite set of $N$-bit strings.

Note that, modelling the real-world functions such as SHA-2 and SHA-3, we consider the hash function to be unkeyed. Typically, a hash function is designed in two steps. First a compression function $f : \mathcal{M}_f \to \mathcal{Y}$ with small domain is designed. Then one uses a domain extension algorithm $C$, which has a blackbox access to $f$ and implements the hash function $H$ for larger domain.

**Definition 1.** *A domain extender $C$ with oracle access to a family of compression functions $f : [r] \times \mathcal{M}_f \to \mathcal{Y}$ is an algorithm which implements the function $H = C^f : \mathcal{M} \to \mathcal{Y}$.*

**Collision Resistance.** Our definitions of collision (Coll) security is given for any general FIL hash function $H$ built upon the compression functions $f_i$ for $i \in [r]$ where $f_i$s are modeled as ideal random functions. Let $\mathrm{Func}(2n, n)$ denote the set of all functions mapping $2n$ bits to $n$ bits. Then, for a fixed adversary $\mathsf{A}$ and for all $i \in [r]$ where $f_i \xleftarrow{\$} \mathrm{Func}(2n, n)$, we consider the following definition of collision resistance.

**Definition 2.** *Let $\mathsf{A}$ be an adversary against $H = C^f$. $H$ is said to be $(q, \varepsilon)$ collision resistant if for all algorithm $\mathsf{A}$ making $q$ queries it holds that*

$$\mathbf{Adv}_H^{\mathrm{Coll}}(\mathsf{A}) = \Pr\left[ M', M \xleftarrow{\$} \mathsf{A}^f(\varepsilon) \ : \ M \neq M' \text{ and } H(M) = H(M') \right] \leq \varepsilon.$$

**Indifferentiability**

In the game of indifferentiability, the distinguisher is aiming to distinguish between two worlds, the *real* world and the *ideal* world. In the real world, the distinguisher has oracle access to $(C^{\mathcal{F}}, \mathcal{F})$ where $C^{\mathcal{F}}$ is a construction based on an ideal primitive $\mathcal{F}$. In the ideal world the distinguisher has oracle access to $(\mathcal{G}, S^{\mathcal{G}})$ where $\mathcal{G}$ is an ideal functionality and $S$ is a simulator.

**Definition 3 (Indifferentiability** [24]**).** *A Turing machine $C$ with oracle access to an ideal primitive $\mathcal{F}$ is said to be $(t_A, t_S, q_S, q, \varepsilon)$ indifferentiable (Fig. 2) from an ideal primitive $\mathcal{G}$ if there exists a simulator $S$ with an oracle access to $\mathcal{G}$ having running time at most $t_S$, making at most $q_S$ many calls to $\mathcal{G}$* per invocation*, such that for any adversary $\mathsf{A}$, with running time $t_A$ making at most $q$ queries, it holds that*

$$\mathbf{Adv}^{\mathrm{Indiff}}_{(C^{\mathcal{F}},\mathcal{F}),(\mathcal{G},S^{\mathcal{G}})}(\mathsf{A}) \overset{def}{=} \left|\Pr[\mathsf{A}^{(C^{\mathcal{F}},\mathcal{F})} = 1] - \Pr[\mathsf{A}^{(\mathcal{G},S^{\mathcal{G}})} = 1]\right| \leq \varepsilon$$

*$C^{\mathcal{F}}$ is computationally indifferentiable from $\mathcal{G}$ if $t_A$ is bounded above by some polynomial in the security parameter $k$ and $\varepsilon$ is a negligible function of $k$.*

In this paper, we consider an information-theoretic adversary implying $t_A$ is unbounded. We derive the advantage in terms of the query complexity of the distinguisher. The composition theorem of indifferentiability [24] states that if a construction $C^{\mathcal{F}}$ based on an ideal primitive $\mathcal{F}$ is indifferentiable from $\mathcal{G}$, then $C^{\mathcal{F}}$ can be used to instantiate $\mathcal{G}$ in any protocol with single-stage game. We note, however, the composition theorem does not extend to the multi-stage games, or when the adversary is resource-restricted. We refer the reader to [31] for details. We refer to the queries made to $C^{\mathcal{F}}/\mathcal{G}$ as construction queries and to the queries made to $\mathcal{F}/S$ as the primitive queries.

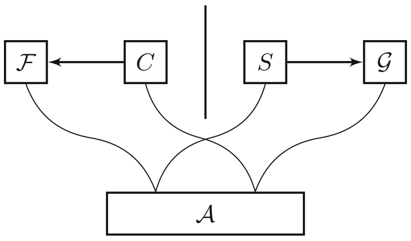


**Fig. 2.** The indifferentiability notion

**Coefficient-H Technique.** We shall prove indifferentiability using Patarin's coefficient-H technique [30]. Fix any distinguisher $\mathcal{D}$ making $q$ queries. As the distinguisher is computationally unbounded, without loss of generality we can assume it to be deterministic [20,29]. The interaction of $\mathcal{D}$ with its oracles is described by a transcript $\tau$. $\tau$ contains all the queries and the corresponding responses $\mathcal{D}$ makes during its execution. Let $\Theta$ denote the set of all possible transcripts. Let $X_{\texttt{real}}$ and $X_{\texttt{ideal}}$ denote the probability distribution of the transcript in the real and the ideal worlds, respectively.

**Lemma 1.** *[30] Consider a fixed deterministic distinguisher $\mathcal{D}$. Let $\Theta$ can be partitioned into sets $\Theta_{good}$ and $\Theta_{bad}$. Suppose $\varepsilon \geq 0$ be such that for all $\tau \in \Theta_{good}$,*

$$\Pr\left[\, X_{\texttt{real}} = \tau \,\right] \geq (1-\varepsilon)\Pr\left[\, X_{\texttt{ideal}} = \tau \,\right]$$

*Then* $\mathbf{Adv}^{\mathit{Indiff}}_{(C^{\mathcal{F}},\mathcal{F}),(\mathcal{G},S^{\mathcal{G}})} \leq \varepsilon + \Pr\left[\, X_{\texttt{ideal}} \in \Theta_{bad} \,\right]$

**Markov Inequality.** We recall the well known Markov inequality.

**Lemma 2.** *Let $X$ be a non-negative random variable and $a > 0$ be a real number. Then it holds that*

$$\Pr[X \geq a] \leq \frac{\mathbf{E}\left[\, X \,\right]}{a}$$

## 3 Compactness: Normalizing Efficiency for Optimally Secure Constructions

In Crypto 2008, Stam made the following conjecture (Conjecture 9 in [37]): If $C^f : \{0,1\}^{m+s} \to \{0,1\}^s$ is a compression function making $r$ calls to primitive $f : \{0,1\}^{n+c} \to \{0,1\}^n$, a collision can be found in the output of $C$ by making $q \leq 2^{\frac{nr+cr-m}{r+1}}$ queries. The conjecture was proved in two papers, the case $r = 1$ was proved by Steinberger in [38], whereas the general case was proved by Steinberger, Sun and Yang in [39]. The result, in our notation, is stated below.

**Theorem 1** ([39])**.** *Let $f_1, f_2, \ldots, f_r : \{0,1\}^{n+c} \to \{0,1\}^n$ be potentially distinct $r$ many compression functions. Let $C : \{0,1\}^{m+s} \to \{0,1\}^s$ be a domain extension algorithm making queries to $f_1, f_2, \ldots, f_r$ in the fixed order. Suppose it holds that $1 \leq m \leq (n+c)r$ and $\frac{s}{2} \geq \frac{nr+cr-m}{r+1}$. There exists an adversary making at most $q = \mathcal{O}\left(r2^{\frac{nr+cr-m}{r+1}}\right)$ queries finds a collision with probability at least $\frac{1}{2}$.*

In other words, if one wants to construct a hash function that achieves birthday bound collision security asymptotically, the query complexity of the attacker must be at least $2^{s/2}$. Then the parameters must satisfy the following equation:

$$\frac{nr+cr-m}{r+1} \geq \frac{s}{2}$$

Next, we rearrange the equation and get

$$m \leq \frac{2nr+2cr-sr-s}{2}$$

Thus we can analyze the security-efficiency trade-off across different constructions by considering only the schemes secure (asymptotically) up to the birthday bound and describe the efficiency by the ratio $\frac{2m}{2nr+2cr-sr-s}$. Then we argue that the optimal efficiency is reached when the parameters satisfy

$$m = \frac{2nr + 2cr - sr - s}{2}$$

Now we are ready to define compactness of hash functions based on compressing primitives.

**Definition 4 Compactness.** *Let $f_1, f_2, \ldots, f_r : \{0,1\}^{n+c} \to \{0,1\}^n$ be potentially distinct $r$ many compression functions. Let $C : \{0,1\}^{m+s} \to \{0,1\}^s$ be a domain extension algorithm making queries to $f_1, f_2, \ldots, f_r$ in the fixed order. We say $C$ is $\alpha$-compact if*

– *for all adversary $\mathsf{A}$ making $q$ queries, for some constant $c_1, c_2$, it satisfies that*

$$\mathbf{Adv}_C^{\mathrm{Coll}}(\mathsf{A}) \le \mathcal{O}\left(\frac{s^{c_1} r^{c_2} q^2}{2^s}\right),$$

–

$$\alpha = \frac{2m}{2nr + 2cr - sr - s}$$

Clearly for any construction, $\alpha \le 1$. For the rest of the paper, we consider constructions where $s = n$. Thus, we derive the value of $\alpha$ as

$$\alpha = \frac{2m}{2cr + nr - n}$$

In Sect. 3.1, in Examples 1 and 3 we estimate that both Merkle–Damgård and Merkle tree domain extenders with $2n$-to-$n$-bit compression function primitives have a compactness of $\approx 2/3$.

## 3.1 Compactness of Existing Constructions

*Example 1.* We consider the textbook **Merkle–Damgård** (MD) domain extension with length padding and fixed IV. Let the underlying function be a $2n$-to $n$-bit compression function $f$. Let the total number of calls to $f$ be $r$. At every call $n$-bits of message is processed. Assuming the length-block is of one block, the total number of message bits hashed using $r$ calls is $(r-1)c$. Hence, we get $m = (r-1)c - n$. Putting $c = n$ we compute

$$\alpha = \frac{2n(r-1) - 2n}{2nr + nr - n} = \frac{2nr - 4n}{3nr - n} < \frac{2}{3}$$

*Example 2.* For binary **Merkle tree** with $c = n$, let the number of $f$ calls at the leaf level is $z$. Then the total number of message bit is $2nz$. Let the total number of calls to the compression function $f$ is $r = z + z - 1 = 2z - 1$. Comparing with the number of message bits we get $m + n = (r+1)n$ which implies $m = rn$. So we calculate the compactness of Merkle tree as

$$\alpha = \frac{2rn}{3nr - n} = \frac{2r}{3r - 1} < \frac{2}{3}$$

*Example 3.* Next we consider **Shrimpton-Stam $2n$-to-$n$** compression function using three calls to $n$-to-$n$-bit function $f$. Here $m = n$ and $c = 0$. Then $\alpha = \frac{2n}{3n-n} = 1$. The **Mennink-Preneel** generalization [25] of this construction gives $2n$-to-$n$-bit compression function making three calls to $n$-bit permutations. Thus in that case $\alpha = \frac{2n}{3n-n} = 1$ as well.

*Example 4.* Consider again MD domain extension with length padding and fixed IV but let the underlying function be a $5n$-to $n$-bit compression function $f$. At every (out of $r$) $f$ call $4n$-bits are processed (the rest $n$-bits are the chaining value). As we have one length-block, the total number of message bits hashed is $(r-1)4n$. Hence, we get $m = (r-1)4n - n$ and compute:

$$\alpha = \frac{2 \times 4n(r-1) - 2n}{2 \times 4r + nr - n} = \frac{8nr - 6n}{9nr - n} \approx \frac{8}{9}$$

*Example 5.* The 5-ary Merkle tree with $5z$ leaf messages has $5nz$ bit input in total. Thus $r = \frac{3(5z-1)}{4}$ and $m = n(5z-1)$. The compactness is given by

$$\alpha = \frac{2n(5z-1)}{2nr + nr - n} = \frac{5z-1}{3r-1} = \frac{8(5z-1)}{9(5z-1)-4} \approx \frac{8}{9}$$

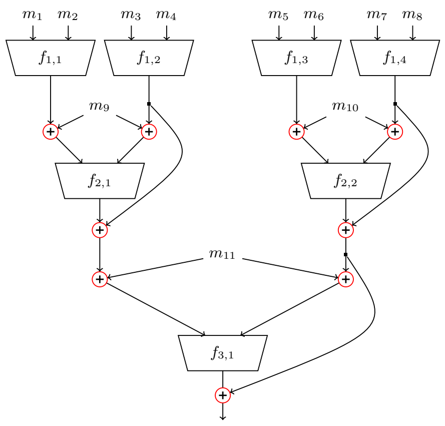
## 4 ABR Mode with Compactness $\alpha = 1$

In this section we present the ABR domain extender. We prove its collision resistance in the random oracle model and show that it is optimally ($\alpha = 1$)-compact. Our ABR mode collision-resistance-proof is valid for FIL trees. That means that our result is valid for trees of arbitrary height but once the height is fixed, all the messages queried by the adversary must correspond to a tree of that height. We remind the reader that the majority of Merkle tree applications rely *exactly* on FIL Merkle trees.[1] The parameter of our construction is $\ell$ which denotes the height of the tree. The construction makes $r = 2^\ell - 1$ many independent $2n$-to-$n$-bit functions and takes input messages from the set $\{0,1\}^{\mu n}$, where $\mu = 2^\ell + 2^{\ell-1} - 1$. $f_{(j,b)}$ denotes the $b^{th}$ node at $j^{th}$ level. The parents of $f_{(j,b)}$ are denoted by $f_{(j-1,2b-1)}$ and $f_{(j-1,2b)}$. We use the following notations for the messages. Let $M$ be the input messages with $\mu$ many blocks of $n$-bits. The corresponding input to a leaf node $f_{(1,b)}$ is denoted by $m_{(1,2b-1)}$ and $m_{(1,2b)}$. For the internal function $f_{(j,b)}$, $m_{(j,b)}$ denotes the message that is xored with the previous chaining values to produce the input. We refer the reader to Fig. 3b for a pictorial view. Note, the leaves are at level 1 and the root of the tree is at level $\ell$. The message is broken in $n$-bit blocks. $2^\ell$ many message blocks are processed at level 1. For level $j(>1)$, $2^{\ell-j}$ many blocks are processed. The adversary $\mathsf{A}$ has query access to all functions, and it makes $q$ queries in total.

[1] Although VIL Merkle tree exists with collision preservation proof, that is done at the cost of an extra block of Merkle-Damgård-type strengthening and padding schemes. As Stam's bound is derived for FIL constructions, we restrict our focus on FIL constructions only.

```
$y \leftarrow$ ABR mode$(m_1, \ldots, m_{2^\ell+2^{\ell-1}-1})$
$i \leftarrow 1, j \leftarrow 1$
 do
   $y_{1,j} = f_{1,j}(m_i, m_{i+1})$
   $i \leftarrow i+2, j \leftarrow j+1$
while $i < 2^\ell$
$count \leftarrow 2^\ell$
for $j$ in$\{2, \ldots, \ell\}$
$i \leftarrow 1, s \leftarrow count$
   do
      $y_{j,i} = f_{j,i}(m_{s+i} \oplus y_{j-1,2i-1},$
            $m_{s+i} \oplus y_{j-1,2i}) \oplus y_{j-1,2i}$
   while $i < 2^{\ell-j}$
$count \leftarrow count + 2^{\ell-j}$
endfor
return $y_{\ell,1}$
```

(a) Algorithm for computing ABR mode hash value with height $\ell$


$m_1$ $m_2$ $m_3$ $m_4$ $m_5$ $m_6$ $m_7$ $m_8$
$f_{1,1}$ $f_{1,2}$ $f_{1,3}$ $f_{1,4}$
$m_9$ $m_{10}$
$f_{2,1}$ $f_{2,2}$
$m_{11}$
$f_{3,1}$

(b) ABR mode of height $\ell = 3$ with $2^3$ leaf message inputs (valid for Merkle tree), $r = 7$ compression function calls, and total of $2^\ell + 2^{\ell-1} - 1 = 11$ input blocks.

**Fig. 3.** ABR mode algorithm and instantiation

**Theorem 2.** *Let $\ell \geq 2$ be a natural number and $r = 2^\ell$. Let $f : [r] \times \{0,1\}^{2n} \rightarrow \{0,1\}^n$ be a family of functions. Let $\mathsf{A}$ be an adversary against the collision resistance of ABR mode. If the elements of $f$ are modeled as independent random oracles, then*

$$\mathbf{Adv}^{\mathrm{Coll}}_{ABR}(\mathsf{A}^f) = \mathcal{O}\left(\frac{rn^2q^2}{2^n}\right).$$

*where $q$ is the number of queries $\mathsf{A}$ makes to $f$ satisfying $q^2 < \frac{2^n}{2e(n+1)}$.*

## 4.1 Warmup: ABR Mode with Height 2

First, we prove the security of the case $\ell = 2$. In this case ABR mode implements a $5n$-to-$n$-bit compression function with 3 calls to $2n$-to-$n$-bit compression functions. For convenience of explanation, we refer the three functions as $f_0, f_1, f_2$ (see Fig. 1b).

**Construction 3.** *Let $f_0, f_1, f_2 : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ be three compression functions. We define ABR mode for $\ell = 2$ as $ABR^f : \{0,1\}^{5n} \rightarrow \{0,1\}^n$ where*

$$ABR(m_1, m_2, m_3, m_4, m_5) = f_2(x_3, x_4) \oplus f_0(m_5 \oplus f_1(x_1, x_2), m_5 \oplus f_2(x_3, x_4))$$

Theorem 2 can be restated for this case as the following proposition.

**Proposition 4.** *Let $f_0, f_1, f_2 : \times\{0,1\}^{2n} \to \{0,1\}^n$. Let $\mathsf{A}$ be an adversary against the collision resistance of ABR. If $f_i$s are modeled as independent random oracles, then*

$$\mathbf{Adv}^{\mathrm{Coll}}_{\mathit{ABR}}(\mathsf{A}^f) = \mathcal{O}\left(\frac{n^2q^2}{2^n}\right)$$

*where $q$ is the maximum number of queries $\mathsf{A}$ makes to the oracles $f_0, f_1, f_2$s.*

**Proof of Proposition 4.** The proof strategy closely follows [36].
MOVING TO LEVEL-WISE SETTING. In general, one needs to consider the adversary making queries in some adaptive (and possibly probabilistic) manner. But for the case of $5n$-bit to $n$-bit ABR, as in [36], we can avoid the adaptivity as $f_1$ and $f_2$ are independent random oracles.

**Lemma 3.** *For every adaptive adversary $\hat{\mathsf{A}}$, there exists an adversary $\mathsf{A}$ who makes level-wise queries and succeeds with same probability;*

$$\mathbf{Adv}^{\mathrm{Coll}}_{\mathit{ABR}}(\hat{\mathsf{A}}) = \mathbf{Adv}^{\mathrm{Coll}}_{\mathit{ABR}}(\mathsf{A}).$$

COLLISION PROBABILITY IN THE LEVEL-WISE QUERY SETTING. From this point on, we assume that the adversary is provided with two lists $L_1$ and $L_2$ at the start of the game. $L_1$ and $L_2$ have $q$ uniformly sampled points and they should be considered as the responses of the queries made by the adversary to $f_1$ and $f_2$, respectively. The adversary only needs to query $f_0$.

Let $\mathsf{A}$ be an adversary that can find a collision in ABR. Two cases may arise. In the first case, $\mathsf{A}$ can find collision in the leaf nodes ($f_1$ or $f_2$). In that case, there is a collision in either $L_1$ and $L_2$. In the other case, there is no collision among the outputs of $f_1$ or $f_2$, and the collision is generated at the final output. Let $\mathsf{Coll}_i$ denote the event that $\mathsf{A}$ finds a collision in $L_i$. Let $\mathsf{Coll}$ denote the event that $\mathsf{A}$ finds a collision in ABR.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{Coll}}_{\mathsf{ABR}}(\mathsf{A}^f) \le \Pr[\mathsf{Coll}] &= \Pr[\mathsf{Coll} \wedge (\mathsf{Coll}_1 \vee \mathsf{Coll}_2)] + \Pr[\mathsf{Coll} \wedge \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)] \\ &\le \Pr[\mathsf{Coll}_1 \vee \mathsf{Coll}_2] + \Pr[\mathsf{Coll} \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)] \\ &\le \Pr[\mathsf{Coll}_1] + \Pr[\mathsf{Coll}_2] + \Pr[\mathsf{Coll} \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)].\end{aligned}$$

As the functions are independent random oracles, $\Pr[\mathsf{Coll}_1]$ and $\Pr[\mathsf{Coll}_2]$ are bounded above by $\frac{q^2}{2^n}$. In the remaining, we bound the probability of the third term.
DEFINING THE RANGE. For every query $(u_i, v_i)$ made by the adversary to $f_0$, we define the following quantity

$$Y_i \overset{def}{=} |\{(h_1, h_2) \mid h_1 \in L_1, h_2 \in L_2, h_1 \oplus u_i = h_2 \oplus v_i\}|.$$

where $f_0(u_i, v_i)$ is the $i^{th}$ query of the adversary. While $Y_i$ counts the number of valid or more precisely consistent with the ABR structure pairs $(h_1, h_2)$ that were already queried to $f_1$ and $f_2$, $Y_i$ also denotes the number of possible $\mathsf{ABR}$ hash outputs produced by the adversary by making $f_0(u_i, v_i)$ query. Notice, that $Y_i$ inputs to $f_0$ generate $Y_i$ outputs. Each of these outputs are XORed each with only one corresponding consistent $h_2$ value determined by the equation $h_1 \oplus u_i = h_2 \oplus v_i$, hence producing $Y_i$ $\mathsf{ABR}$ outputs on $Y_i$ consistent number inputs to $f_0$. Let $Y = \max_i Y_i$.

Bounding Collision by range. Now, we show how bounding the range will help us bounding the collision probability. Let $E_i$ denotes the probability that after making the $i^{th}$ query $f_0(u_i, v_i)$ produces a collision in the output of $\mathsf{ABR}$. Suppose after making $i-1$ queries, adversary is not able to produce a collision for $\mathsf{ABR}$. Hence, the adversary has produced $\sum_{j=1}^{i-1} Y_j$ many hash outputs. We bound the probability that $i^{th}$ query response produces a collision.

$$\Pr\left[E_i \mid \wedge_{j=1}^{i-1} \neg E_j\right] \leq \frac{Y_i \sum_{j=1}^{i-1} Y_j}{2^n}$$

Now we can bound the collision probability as

$$\Pr\left[\,\mathsf{Coll} \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] \leq \sum_{i=1}^{q} \frac{Y_i \sum_{j=1}^{i-1} Y_j}{2^n} \leq \sum_{i=1}^{q} \sum_{j=1}^{i-1} \frac{Y^2}{2^n} \leq \frac{q^2 Y^2}{2^{n+1}}$$

We shall use the following lemma, which we prove later.

**Lemma 4.**

$$\Pr\left[\,Y > k \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] \leq \frac{q^{2k}(2^n - k)!}{k!\,(2^n - 1)!}$$

Using Lemma 4, we get

$$\begin{aligned}\Pr\left[\,\mathsf{Coll} \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] &\leq \Pr\left[\,\mathsf{Coll} \wedge Y \leq k \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] \\ &\quad + \Pr\left[\,Y > k \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] \\ &\leq \frac{k^2 q^2}{2^{n+1}} + \frac{q^{2k}(2^n - k)!}{k!\,(2^n - 1)!}\end{aligned}$$

Putting $k = n$ we get the probability as

$$\begin{aligned}\Pr\left[\,\mathsf{Coll} \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)\,\right] \leq \frac{n^2 q^2}{2^{n+1}} + \frac{q^{2n}}{n!\,(2^n - 1)\cdots(2^n - n + 1)} &\approx \frac{n^2 q^2}{2^{n+1}} + \frac{q^{2n}}{2^{n^2}} \\ &= \mathcal{O}\left(\frac{n^2 q^2}{2^n}\right)\end{aligned}$$

Hence, we get the theorem. □

**Proof of Lemma 4.** Let $(h_{i_1}, h'_{j_1}), (h_{i_2}, h'_{j_2}), \cdots, (h_{i_k}, h'_{j_k})$ be the set of $k$ pairs such that each $h_{i_l} \in L_1$ and $h'_{j_l} \in L_2$, and

$$h_{i_1} \oplus h'_{j_1} = h_{i_2} \oplus h'_{j_2} = \cdots = h_{i_k} \oplus h'_{j_k} = a \text{ (say)}$$

The condition $\neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2)$ implies that there is no collision in $L_1$ and $L_2$. The total number of ways to choose each of $L_1$ and $L_2$ such that there is no collision is $q!\binom{2^n}{q}$.

Next we count the number of ways of choosing $L_1$ and $L_2$ such that the $k$ equalities get satisfied. The number of ways we can choose $i_1, i_2, \cdots, i_k$ is $\binom{q}{k}$. Fixing the order of $i_1, i_2, \cdots, i_k$, the number of ways to pair $j_1, j_2, \cdots, j_k$ is $k!\binom{q}{k}$. Observe that there can be $2^n$ many possible values of $a$. Fix a value of $a$. Thus for each value of $h_{i_l}$, there is a single value of $h'_{j_l}$. Hence the total number of ways we can select $L_1, L_2$ such that the equalities get satisfied is $q!\binom{2^n}{q} \times q!\binom{2^n-k}{q}$. Hence the probability that for independently sampled $L_1$ and $L_2$,

$$\Pr\left[\, Y > k \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2) \,\right] = \frac{k!\left(\binom{q}{k}\right)^2 2^n q!\binom{2^n}{q} \times q!\binom{2^n-k}{q}}{\left(q!\binom{2^n}{q}\right)^2}$$

After simplification, we get the probability as

$$\Pr\left[\, Y > k \mid \neg(\mathsf{Coll}_1 \vee \mathsf{Coll}_2) \,\right] = \frac{(q!)^2 2^n (2^n-k)!}{\left((q-k)!\right)^2 k!\,(2^n)!} \leq \frac{q^{2k} 2^n (2^n-k)!}{k!\,(2^n)!}$$

At the last step, we upper bound $\frac{(q!)^2}{((q-k)!)^2}$ by $q^{2k}$. The lemma follows. $\square$

## 4.2 Proof of Theorem 2

**Proof Overview.** Now we prove the general case. We start with an overview of the proof. Unlike the case for $\ell = 2$, we have to consider adaptive adversaries. Specifically, we can no longer assume that the adversary makes the queries level wise. Indeed, a query at a non-leaf level is derived from the previous chaining values (part of which is fed-forward to be xored with the output) and the messages. We can no longer "replace" the query without changing the chaining values. To the best of our knowledge, no proof technique achieving $2^{n/2}$ security bound asymptotically, exists in the literature for this case.

The intuition of our proof follows. Like in the previous case, our analysis focuses on the yield of a function. Informally, the yield of a query $(u, v)$ to a function $f$ is the number of chaining values created by the query. For example, consider a query $(u, v)$ made to function $f_{j,z}$, $z^{th}$ function of level $j$, and let $y$ be the output of the query. How many chaining values does this query create? A cursory inspection reveals that the number of created chaining values are the number of "legal" feedforward (chaining value from the previous level function $f_{j-1,2z}$) values $h$. Indeed a feedforward value $h$ can extend the chain, if there exists a chaining value $h'$ from the set of chaining values created from $f_{j-1,2z-1}$ (the other parent of $(j, z)$) such that $h' \oplus u = h \oplus v$.

Naturally, if we can bound the total yield of a function (denoted as load), we can bound the probability of collision among the chaining values generated by the function. The load of a function $f_{j,z}$ gets increased in two ways. The first one is by a query made to $f_{j,z}$, as encountered in the previous section. The other one is by a query made to $f_{j',z'}$ where $j' < j$ and $(j', z')$ is in the subtree of $(j, z)$. To see why the second case holds, observe that the query to $f_{j',z'}$ increases the yield of the function, and thus creating new chaining values. Some of those newly created chaining values can be "legal" feedforward values for some queries already made to the next level, and thus increasing the yield of that query as well. Moreover, this in turn again creates new chaining value at the level $j' + 1$. The effect continues to all the next levels and eventually affects the load of all the functions in the path to the root, including $(j, z)$.

We bound the load of functions at each level starting from the leaves. At each level, we bound the probability of having a transcript which creates the load on a function (of that level) over a threshold amount, conditioned on the event that in none of the previous level the load exceeded the threshold.

**Formal Analysis.** Our formal analysis involves the transcript of the queries and the corresponding responses. Each entry of the transcript contains a query response pair, denoted by $(u, v, y)_{(j,b)}$ which indicates that $y$ is the response of the query $f_{j,b}(u, v)$. $\tau$ denotes the (partial) transcript generated after the $q$ many queries. $Q_{(j,b)}$ denotes the set of queries made to the function $f_{(j,b)}$. $\mathcal{L}_{(j,b)}$ holds the responses.

Yield Set. For each function $f_{(j,b)}$, we define a set $\Gamma_{(j,b)}$ holding the possible chaining values. Note, a chaining value $h \in \Gamma_{(j-1,2b)}$ can be a valid feedforward value for entry $(u, v, y)_{(j,b)}$ if there exists a matching $h' \in \Gamma_{(j-1,2b-1)}$ such that for some $m'$, it holds that $m' \oplus h' = u$ and $m' \oplus h = v$. Such a $m'$ can exist only if $h' \oplus u = h \oplus v$.

$$\Gamma_{(1,b)} \overset{def}{=} \{y \mid (u, v, y)_{(1,b)} \in \tau\}$$
$$\Gamma_{(j>1,b)} \overset{def}{=} \{y \oplus h \mid (u, v, y)_{(j,b)} \in \tau, h \in \Gamma_{(j-1,2b)}, \exists h' \in \Gamma_{(j-1,2b-1)}, h' \oplus u = h \oplus v\}.$$

Feedforward set. For each function $f_{(j,b)}$, we define a set $F_{(j,b)}$ containing the possible elements that can be used as feedforward and xored with the output of $f_{(j,b)}$ to generate valid chaining values. It is easy to verify that $F_{(j,b)} = \Gamma_{(j-1,2b)}$, where $\Gamma_{(0,b)} = \emptyset$.

Let $\mathsf{Coll}$ denotes the event that the adversary finds collision in $\mathsf{ABR}$ mode. Let $M = (m_{1,1}, m_{1,2} \cdots, m_{1,2^\ell}, \cdots, m_{\ell,1})$ and $M' = (m'_{1,1}, m'_{1,2} \cdots, m'_{1,2^\ell}, \cdots, m'_{\ell,1})$ be the two distinct messages that produce the collision. We use $(u, v, y)_{(j,b)}$ and $(u', v', y')_{(j,b)}$ to be the corresponding queries made to function $f_{(j,b)}$ in the evaluation respectively.[2]

**Proper Internal Collision.** The transcript is said to contain a *proper internal collision* at $(j, b)$, if the transcript contains two distinct queries $(u, v, y)_{(j,b)}$ and $(u', v', y')_{(j,b)}$ and there exists $h, h' \in \Gamma_{(j-1,2b)}$ such that $y \oplus h = y' \oplus h'$.

[2] We assume the adversary makes all the internal queries before producing a collision. Indeed we can always add the missing queries in the transcript without significantly changing the query complexity.

**Lemma 5.** *Collision in tree implies a proper internal collision.*

*Proof.* The proof follows the Merkle tree collision resistance proof. Without loss of generality, we assume that there is no collision at the leaf. Now, consider a collision in the tree. This implies that there exist $(u, v, y)_{(\ell,1)}, (u', v', y')_{(\ell,1)} \in \tau$ and $h, h' \in \Gamma_{(\ell-1,2)}$ such that

$$y \oplus h = y' \oplus h'$$

If $(u, v)_{(\ell,1)} \neq (u', v')_{(\ell,1)}$, then we get our proper internal collision at $(\ell, 1)$, and we are done. Otherwise $(u, v)_{(\ell,1)} = (u', v')_{(\ell,1)}$, which in turn implies $y = y'$. This implies $h = h'$. Moreover, we get $h \oplus u \oplus v = h' \oplus u' \oplus v'$ . The above two equalities give us collision in the both left and the right subtree. As $M \neq M'$, the messages differ in one of the subtrees. Repeating the above argument in the appropriate tree, we indeed find a $(j, b)$ with distinct inputs $(u, v)_{(j,b)} \neq (u', v')_{(j,b)}$. □

## Bounding Probabilities of a Proper Internal Collision

Yield of a query. Consider an element $(u, v, y)_{(j,b)} \in \tau$. We define the following quantity as the yield of the query $f_{(j,b)}(u, v)$.

$$Y_{u,v,j,b} \overset{def}{=} \begin{cases} \mid \{(h_1, h_2) \mid h_1 \in \Gamma_{(j-1,2b-1)}, h_2 \in \Gamma_{j-1,2b}, h_1 \oplus u = h_2 \oplus v\} \mid & \text{if } j > 1 \\ 1 & \text{if } j = 1 \end{cases}$$

Load on a function. The load on a function $f_{(j,b)}$ is defined by the total yield of the queries made to that function.

$$L_{(j,b)} \overset{def}{=} \sum_{(u_i,v_i) \in Q_{j,b}} Y_{u_i,v_i,j,b}.$$

Observe that if no internal collision happens at a function, the size of the yield set is the load on that function; $L_{(j,b)} = \mid \Gamma_{j,b} \mid$

For the rest of the analysis we use the variable $k$ which is equal to $(n+1)^{\frac{1}{\ell}}$.
Bad Events. In this section we define the notion of bad event. We observe that with every query, the load on the functions in the tree change. Two types of contributions to load happen with each query.

1. **Type I** A new $(u, v)_{(j,b)}$ query contributes to $L_{(j,b)}$. The contribution amount is $Y_{(u,v,j,b)}$.
2. **Type II** A new $(u, v)_{j',b'}$ query increases the load of $(j, b)$ where $j > j'$ and $(j', b')$ is in the sub-tree rooted at $(j, b)$.

$\delta^1_{(j,b)}$ and $\delta^2_{(j,b)}$ denotes the total type-I and type-II contributions to $L_{(j,b)}$ respectively. We consider the following two helping Bad events.

1. Bad1 happens at function $(j, b)$ such that for some $(u, v, y)_{(j,b)} \in \tau$, such that $Y_{(u,v,j,b)} > k^\ell$. This event corresponds to the Type I queries.

2. $\mathsf{Bad2}$ happens at function $(j, b)$, if $\delta^2_{(j,b)} > k^\ell q$.

$\mathsf{Bad1}_j$ and $\mathsf{Bad2}_j$ denotes the event that $\mathsf{Bad1}$ or $\mathsf{Bad2}$ respectively happens at some node at level $j$. We define $\mathsf{Bad}_j$ as $\mathsf{Bad1}_j \cup \mathsf{Bad2}_j$. Let $\mathsf{Bad}$ denote the event that for the generated transcript $\mathsf{Bad}_j$ holds for some level $j$.

$$\mathsf{Bad} \overset{def}{=} \bigcup_j \mathsf{Bad}_j$$

The following proposition holds from the definitions.

**Lemma 6.**

$$\neg\mathsf{Bad}_j \implies \forall b \in [2^{\ell-j}] \ \ \textit{it holds that } L_{(j,b)} \leq 2k^\ell q$$

Deriving Collision Probability. Let $\mathsf{Coll}_j$ denote the event of a proper internal collision at $(j, b)$ for some $b \in [2^{\ell-j}]$.

$$\begin{aligned}
\Pr[\mathsf{Coll}] &\leq \Pr[\mathsf{Coll} \cup \mathsf{Bad}] \\
&\leq \Pr[\mathsf{Coll}_1 \cup \mathsf{Bad}_1] + \sum_{j>1} \Pr\left[(\mathsf{Coll}_j \cup \mathsf{Bad}_j) \cap \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'<j} \neg\mathsf{Bad}_{j'}\right] \\
&\leq \Pr[\mathsf{Coll}_1 \cup \mathsf{Bad}_1] + \sum_{j>1} \Pr\left[\mathsf{Bad}_j \cap \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'<j} \neg\mathsf{Bad}_{j'}\right] + \\
&\qquad \sum_{j>1} \Pr\left[\mathsf{Coll}_j \cap \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j} \neg\mathsf{Bad}_{j'}\right]
\end{aligned}$$

Using the fact that $\Pr[A \cap B] = \Pr[A \mid B] \Pr[B] \leq \Pr[A \mid B]$,

$$\begin{aligned}
\Pr[\mathsf{Coll}] \leq &\Pr[\mathsf{Coll}_1 \cup \mathsf{Bad}_1] + \sum_{j>1} \Pr[\mathsf{Bad}_j \mid \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'<j} \neg\mathsf{Bad}_{j'}] \\
&+ \sum_{j>1} \Pr[\mathsf{Coll}_j \mid \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j} \neg\mathsf{Bad}_{j'}] \qquad (1)
\end{aligned}$$

**Bounding.** $\Pr[\mathsf{Coll}_1 \cup \mathsf{Bad}_1]$. As all the functions are modeled as a random function, for all $b \in [2^{\ell-1}]$, we have $\Pr[\mathsf{Coll}_{1,b}] \leq \frac{q^2}{2^n}$. Hence,

$$\Pr[\mathsf{Coll}_1] \leq \frac{2^{\ell-1} q^2}{2^n}$$

In order to find $\Pr[\mathsf{Bad}_1]$, we recall that $F_{1,b} = \emptyset$. In other words the nothing is xored with the output of the functions at the leaf level. Hence, $Y_{(u,v,1,b)} = 1$ for all $b \in [2^{\ell-1}]$ and $(u, v, y)_{1,b} \in \tau$. Hence $\Pr[\mathsf{Bad}_1] = 0$. Hence we get,

$$\Pr[\mathsf{Coll}_1 \cup \mathsf{Bad}_1] \leq \frac{2^{\ell-1} q^2}{2^n} \qquad (2)$$

**Bounding.** $\sum_{j>1} \Pr[\mathsf{Coll}_j \mid \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j} \neg\mathsf{Bad}_{j'}]$. Fix $b \in [2^{\ell-j}]$ and thus fix a function at the $j$þ level. As analyzed in the previous section, given

$\cap_{j'\leq j}\neg\mathsf{Bad}_{j'}$, the proper internal collision probability for $(j,b)$ is $\frac{L^2_{(j,b)}}{2^n}$. From Lemma 6, it holds that for each $b \in [2^{\ell-j}]$, $L_{(j,b)} \leq 2k^\ell q$. Hence for each $j > 1, b \in [2^{\ell-j}]$,

$$\Pr\left[\,\mathsf{Coll}_{(j,b)} \mid \cap_{j'<j}\neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j}\neg\mathsf{Bad}_{j'}\,\right] \leq \frac{4k^{2\ell}q^2}{2^n}.$$

Taking sum over all $j > 1, b \in [2^{\ell-j}]$,

$$\begin{aligned}\sum_{j>1,b}\Pr\left[\,\mathsf{Coll}_{(j,b)} \mid \cap_{j'<j}\neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j}\neg\mathsf{Bad}_{j'}\,\right] &\leq \sum_{j=2}^{\ell}\sum_{b=1}^{2^{\ell-j}}\frac{4k^{2\ell}q^2}{2^n}\\ &= \sum_{j=2}^{\ell}2^{\ell-j}\times\frac{4k^{2\ell}q^2}{2^n}\\ &= \frac{2^{\ell+2}k^{2\ell}q^2}{2^n}\times\left(\sum_{j=2}^{\ell}\frac{1}{2^j}\right)\end{aligned}$$

In the next step we shall use the fact that $\sum_{j=2}^{\ell}\frac{1}{2^j} < \frac{1}{2}$. Finally we get,

$$\sum_{j>1,b}\Pr\left[\,\mathsf{Coll}_{(j,b)} \mid \cap_{j'<j}\neg\mathsf{Coll}_{j'} \cap \cap_{j'\leq j}\neg\mathsf{Bad}_{j'}\,\right] \leq \frac{2^{\ell+2}k^{2\ell}q^2}{2^{n+1}} \tag{3}$$

**Bounding $\mathbf{Pr}\,[\mathsf{Bad}]$.** Now we bound the probabilities of the two bad events. We bound the probabilities level-wise. Let $\mathsf{Bad1}_{j,b}$ denote that $\mathsf{Bad1}$ happens at node $b$ of level $j$. Similarly, let $\mathsf{Bad2}_{j,b}$ denote that $\mathsf{Bad2}$ happens at node $b$ of level $j$. Clearly, $\mathsf{Bad1}_j = \cup_{b\in[2^{\ell-j}]}\mathsf{Bad1}_{j,b}$ and $\mathsf{Bad2}_j = \cup_{b\in[2^{\ell-j}]}\mathsf{Bad2}_{j,b}$

**Bounding $\mathbf{Bad1}_j$**

**Lemma 7.** *For any $(u,v,y)_{(j,b)}$ for $b \in [2^{\ell-j}]$*

$$\Pr[\mathbf{Bad1}_{j,b} \mid \cap_{j'<j}\neg\mathsf{Coll}_{j'} \cap \cap_{j'<j}\neg\mathsf{Bad}_{j'}] \leq 2^n\left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell}$$

*Proof.* We bound the probability for any possible input $(u,v)_{(j,b)}$ that $Y_{(u,v,j,b)} > k^\ell$. Fix $u \oplus v = a$. Consider any entry $(u_1,v_1,y_1)_{(j-1,2b)}$ from $\tau$. This entry contributes to $Y_{(u,v,j,b)}$ if there exists a $h \in F_{(j-1,2b)}$ and $x \in \Gamma_{(j-1,2b-1)}$ such that $y_1 \oplus h \oplus v = x \oplus u$. Rearranging, we get that $y_1 = h \oplus x \oplus a$. Probability of that event is $\frac{Y_{(u_1,v_1,j-1,2b)}|\Gamma_{(j-1,2b-1)}|}{2^n}$. As $\neg\mathsf{Bad}_{j'}$ holds for all $j' < j$, we have $|\ \Gamma_{(j-1,2b-1)}\ |\leq k^\ell q$, and $Y_{u_1,v_1,j-1,2b} \leq k^{j-1}$. Hence, the probability that

$(u_1, v_1, y_1)_{(j-1,2b)}$ contributes to $Y_{u,v,j,b}$ is at most $\frac{k^{\ell+j-1}q}{2^n}$. As there are at most $q$ choices for $(u_1, v_1, y_1)_{(j-1,2b)}$ and each choice contributes one to $Y_{u,v,j,b}$,

$$\Pr\left[Y_{u,v,j,b} > k^\ell\right] \leq \binom{q}{k^\ell}\left(\frac{k^{\ell+j-1}q}{2^n}\right)^{k^\ell}$$

Next, we use the inequality $\binom{a}{b} \leq \left(\frac{ea}{b}\right)^b$, where $e$ is the base of natural logarithm.

$$\Pr\left[Y_{u,v,j,b} > k^\ell\right] \leq \left(\frac{ek^{j-1}q^2}{2^n}\right)^{k^\ell} \leq \left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell}$$

Now, taking union bound over all possible choice of $a$, we get that for any possible input $(u, v)$ to $f_{(j,b)}$,

$$\Pr\left[Y_{u,v,j,b} > k^\ell\right] \leq 2^n\left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell}$$

□

**Bounding $\mathbf{Bad2}_j$**

**Lemma 8.** *Fix $b \in [2^{\ell-j}]$ and thus fix a function at the $j$þ level.*

$$\Pr[\mathbf{Bad2}_{j,b} \mid \cap_{j'<j}\neg\mathsf{Coll}_{j'} \cap \cap_{j'<j}\neg\mathsf{Bad}_{j'} \cap \neg\mathbf{Bad1}_{j,b}] \leq \frac{2^\ell k^\ell q^2}{2^n}$$

*Proof.* Consider a query $(u, v, y)_{j',b'}$ where $(j', b')$ is in the sub-tree of $(j, b)$. As $\cap_{j'<j}\neg\mathsf{Bad}_{j'}$ holds, we argue $\neg\mathsf{Bad1}_{j'}$ holds. Thus the number of chaining value created by $(u, v, y)_{j',b'}$ query at the output of $j', b'$ is at most $k^\ell$, we have $Y_{u,v,j',b'} \leq k^\ell$.

Next we calculate the increase in the load of the next node $f_{(j'+1,\lceil\frac{b'}{2}\rceil)}$ due to query $(u, v, y)_{j',b'}$. Consider any chaining value $h$ created due to the query $(u, v, y)_{j',b'}$. $h$ increases the load of $(j' + 1, \lceil\frac{b'}{2}\rceil)$ if there exists $h_1 \in \Gamma_{j',b'-1}$ and $(u_1, v_1, y_1)_{j'+1,\lceil\frac{b'}{2}\rceil} \in \tau$ such that $h = h_1 \oplus u_1 \oplus v_1$. For a fixed $h_1$ and query $(u_1, v_1, y_1)_{j'+1,\lceil\frac{b'}{2}\rceil}$, probability the equation gets satisfied is $\frac{1}{2^n}$. There can be at most $|Q_{j'+1,\lceil\frac{b'}{2}\rceil}|$ many queries made to the function $j'+1, \lceil\frac{b'}{2}\rceil$ in the transcript, implying at most $q$ many choices for candidate $(u_1, v_1, y_1)_{j'+1,\lceil\frac{b'}{2}\rceil}$.

$$\mathbf{E}\left[\delta^2_{(j'+1,\lceil\frac{b'}{2}\rceil)}\right] \leq \frac{Y_{u,v,j',b'}\left|\Gamma_{(j',b'-1)}\right|\left|Q_{j'+1,\lceil\frac{b'}{2}\rceil}\right|}{2^n}$$

As $\neg\mathsf{Bad}_{j'}$ holds in the given condition, $\left|\Gamma_{(j',b'-1)}\right| = L_{(j',b'-1)} < 2k^\ell q$. Moreover, $Y_{u,v,j',b'} \leq k^\ell$; thus the expected increase in the load of $f_{(j'+1,\lceil\frac{b'}{2}\rceil)}$ is at most $\frac{2k^{2\ell}q^2}{2^n}$.

We extend this argument to the next levels. For a random element from $Q_{j'+1,\lceil \frac{b'}{2} \rceil} \times \Gamma_{(j',b'-1)}$ the expected number of matched elements in $Q_{j'+2,\lceil \frac{b'}{4} \rceil} \times \Gamma_{(j'+1,\lceil \frac{b'}{2} \rceil -1)}$ is $\frac{\left|\Gamma_{(j'+1,\lceil \frac{b'}{2} \rceil -1)}\right|\left|Q_{j'+2,\lceil \frac{b'}{4} \rceil}\right|}{\left|\Gamma_{(j',b'-1)}\right|\left|Q_{j'+1,\lceil \frac{b'}{2} \rceil}\right|}$. Using $\neg\mathsf{Bad}_{j'}$ for all $j' < j$, we bound the expected increase of load for $f_{(j'+2,\lceil \frac{b'}{4} \rceil)}$ as

$$\begin{aligned}
&\mathbf{E}\left[\delta^2_{(j'+2,\lceil \frac{b'}{4} \rceil)}\right] \\
&\le \frac{Y_{u,v,j',b'} \mid \Gamma_{(j',b'-1)} \mid\mid Q_{j'+1,\lceil \frac{b'}{2} \rceil} \mid}{2^n} \times \frac{\mid \Gamma_{(j'+2,\lceil \frac{b'}{2} \rceil -1)} \mid\mid Q_{j'+1,\lceil \frac{b'}{4} \rceil} \mid}{\mid \Gamma_{(j',b'-1)} \mid\mid Q_{j'+1,\lceil \frac{b'}{2} \rceil} \mid} \\
&\le \frac{Y_{u,v,j',b'} \mid \Gamma_{(j'+1,\lceil \frac{b'}{2} \rceil -1)} \mid\mid Q_{j'+1,\lceil \frac{b'}{4} \rceil} \mid}{2^n} \\
&\le \frac{2k^{2\ell}q^2}{2^n}
\end{aligned}$$

Inductively extending the argument

$$\mathbf{E}\left[\delta^2_{(j,b)}\right] \le \frac{2k^{2\ell}q^2}{2^n}.$$

As there $q$ many queries in the transcript, the expected total type II contribution for a function $(j,b)$ is $\frac{2^\ell k^{2\ell} q^3}{2^n}$. By using Markov inequality we get that

$$\Pr\left[\delta^2_{(j,b)} > k^\ell q\right] \le \frac{\mathbf{E}\left[\delta^2_{(j,b)}\right]}{k^\ell q} \le \frac{2k^\ell q^2}{2^n}$$

□

**Finishing the Proof.** From Lemma 6, Lemma 7, and Lemma 8, we bound the probability of bad as

$$\sum_{j>1} \Pr\left[\mathsf{Bad}_j \mid \cap_{j'<j} \neg\mathsf{Coll}_{j'} \cap \cap_{j'<j} \neg\mathsf{Bad}_{j'}\right] = \sum_{j>1, b\in[2^{\ell-j}]} \left(2^n \left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell} + \frac{2k^\ell q^2}{2^n}\right) \tag{4}$$

$$= \frac{2^{\ell+1}k^\ell q^2}{2^n} + 2^{\ell+n}\left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell} \tag{5}$$

From Eq. 1, Eq. 2, Eq. 3, and Eq. 5, we get,

$$\Pr[\mathsf{Coll}] \le \frac{2^{\ell-1}q^2}{2^n} + \frac{2^{\ell+1}k^{2\ell}q^2}{2^n} + \frac{2^{\ell+1}k^\ell q^2}{2^n} + 2^{\ell+n}\left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell} \tag{6}$$

$$\le \frac{2^{\ell+1}q^2(1+k^\ell+k^{2\ell})}{2^n} + 2^{\ell+n}\left(\frac{ek^\ell q^2}{2^n}\right)^{k^\ell} \tag{7}$$

Finally, putting $k = (n+1)^{\frac{1}{\ell}}$, and assuming $q^2 < \frac{2^n}{2e(n+1)}$, we get

$$2^{\ell+n}\left(\frac{ek^{\ell}q^2}{2^n}\right)^{k^{\ell}} < \frac{2^{\ell}e(n+1)q^2}{2^n}$$

Putting $k^{\ell} = (n+1)$ in Eq. 7,

$$\Pr[\mathsf{Coll}] = \mathcal{O}\left(\frac{2^{\ell}(1+n+n^2)q^2}{2^n}\right) = \mathcal{O}\left(\frac{rn^2q^2}{2^n}\right).$$

This finishes the proof of Theorem 2. □

**Corollary 1.** *The compactness of* $\mathsf{ABR}$ *is* 1.

## 5 Achieving Indifferentiability Efficiently

Below we first consider the basic ABR compression function and analyze its security with respect to the indifferentiability notion. We show that while ABR fails to achieve indifferentiability, a simple modification can restore the indifferentiability. We call that modified tree $\mathsf{ABR}^+$ mode construction. $\mathsf{ABR}^+$ mode is the merge of two ABR modes (trees), not necessarily of the same height $\ell \geq 2$ each, and feeding their inputs to a final compression function (omitting the final message injection and feedforward).

### 5.1 Indifferentiability Attack Against ABR Mode

Our main result of this section is the following.

**Theorem 5.** *Consider the* $\mathsf{ABR}$ *mode with* $\ell = 2$. *There exists an indifferentiability adversary* $\mathsf{A}$ *making* $\mathcal{O}(2^{\frac{n}{3}})$ *many calls such that for any simulator* $S$ *it holds that*

$$\mathbf{Adv}^{\text{Indiff}}_{(\mathsf{ABR},f),(\mathcal{G},S^{\mathcal{G}})}(\mathsf{A}) \geq 1 - \epsilon$$

*where* $\epsilon$ *is a negligible function of* $n$.

Theorem 5 can be extended for $\ell > 2$ as well.

**Principle Behind the Attack.** Recall the ABR with $\ell = 2$ from Fig. 1b. The idea is to find collision on the input of $f_0$ for two distinct messages $m, m'$. If the adversary finds such a collision, then the output of the simulator on this input needs to be consistent with the random oracle ($\mathcal{F}$) responses on two distinct messages. That is impossible unless there is a certain relation at the output of $\mathcal{F}$, making that probability negligible.

**The Attack.** The adversary $\mathsf{A}$ maintains three (initially empty) query-response lists $L_0, L_1, L_2$ for the three functions $f_0, f_1, f_2$, respectively. $\mathsf{A}$ chooses $2^{n/3}$ messages $(x_1^{(1)}, x_2^{(1)}) \in \{0,1\}^{2n}$, queries to $f_1$, and adds the query-response tuple to $L_1$. Similarly, $\mathsf{A}$ chooses $2^{n/3}$ messages $(x_1^{(2)}, x_2^{(2)}) \in \{0,1\}^{2n}$, queries to $f_2$, and adds the query-response tuple to $L_2$. $\mathsf{A}$ checks whether there exists $(x_1^{(1)}, x_2^{(1)}, h_1^{(1)}) \in L_1$, and $(x_1^{(2)}, x_2^{(2)}, h_1^{(2)}) \in L_1$, and $(x_3^{(1)}, x_4^{(1)}, h_2^{(1)}) \in L_2$, and $(x_3^{(2)}, x_4^{(2)}, h_2^{(2)}) \in L_2$ such that

$$h_1^{(1)} \oplus h_1^{(2)} \oplus h_2^{(1)} \oplus h_2^{(2)} = 0 \tag{8}$$

If such tuples do not exist, $\mathsf{A}$ outputs 1 and aborts. If there is collision in the lists, $\mathsf{A}$ outputs 1 and aborts. Otherwise, it chooses a random $\hat{m} \in \{0,1\}^n$. The adversary sets $m = h_1^{(1)} \oplus h_1^{(2)} \oplus \hat{m} = h_2^{(1)} \oplus h_2^{(2)} \oplus \hat{m}$, adversary computes $u = m \oplus h_1^{(1)} = \hat{m} \oplus h_1^{(2)}$ and $v = m \oplus h_2^{(1)} = \hat{m} \oplus h_2^{(2)}$. Finally, adversary queries $z = f_0(u, v)$ and outputs 1 if $z \neq \mathcal{F}(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}, m) \oplus h_2^{(1)}$ or $z \neq \mathcal{F}(x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}, \hat{m}) \oplus h_2^{(2)}$. Else adversary outputs 0.
The full probability analysis is straightforward and skipped in this version.

### 5.2 Almost Fully Compact and Indifferentiable $\mathsf{ABR}^+$ Mode

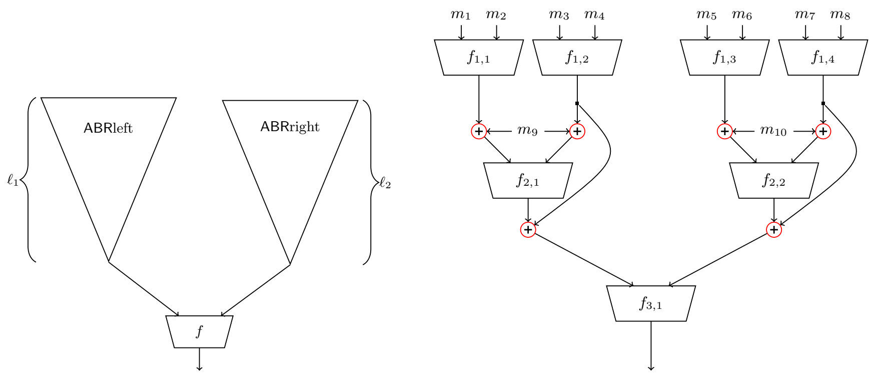
In this section, we show that the generalized $\mathsf{ABR}^+$ mode without the additional message block at the last level is indifferentiable (up to the birthday bound) from a random oracle. For ease of explanation, we prove the result for three-level (see Fig. 4b) balanced tree. The proof for the general case follows exactly the same idea. The generalized $\mathsf{ABR}^+$ mode can be viewed as the merge of two $\mathsf{ABR}$ mode instances, one being the left $\mathsf{ABR}^+$ branch and the other being the right branch. Both their root values are input to a final $2n$-to-$n$-bit compression function to compute the final value of the $\mathsf{ABR}^+$ tree. The $\mathsf{ABR}^+$ tree can be either balanced or unbalanced depending on whether it uses two $\mathsf{ABR}$ modes of identical or distinct heights (see Fig. 4a), respectively.

Our main result here is the following theorem. The result can be generalized to $\mathsf{ABR}^+$ with arbitrary height. However, the simulator description will be more detailed. For ease of explanation we consider the mode with $\ell = 3$.

**Theorem 6.** *Let $f : [7] \times \{0,1\}^{2n} \to \{0,1\}^n$ be a family of random functions. Let $C^f : \{0,1\}^{10n} \to \{0,1\}^n$ be the $\mathsf{ABR}^+$ mode as in Fig. 4b. $(C^f, f)$ is $(t_S, q_S, q, \epsilon)$ indifferentiable from a random oracle $\mathcal{F} : \{0,1\}^{10n} \to \{0,1\}^n$ where*

$$\epsilon \leq \mathcal{O}\left(\frac{n^2 q^2}{2^n}\right).$$

*where $q$ is the total number of queries made by the adversary. Moreover $t_S = \mathcal{O}(q^2)$ and $q_S = 1$*



(a) General $\mathsf{ABR}^+$ mode (b) $\mathsf{ABR}^+$ mode with 10 input messages

**Fig. 4.** $\mathsf{ABR}^+$ mode examples

## 5.3 Proof of Theorem 6

We assume that the distinguisher $\mathcal{D}$ makes all the primitive queries corresponding the construction queries. This is without loss of generality as we can construct a distinguisher $\mathcal{D}'$ for every distinguisher $\mathcal{D}$ such that $\mathcal{D}'$ satisfies the condition. $\mathcal{D}'$ emulates $\mathcal{D}$ completely, and in particular, makes the same queries. However, at the end, for each construction queries made by $\mathcal{D}$, $\mathcal{D}'$ makes *all* the (non-repeating) primitive queries required to compute the construction queries. At the end, $\mathcal{D}'$ outputs the same decision as $\mathcal{D}$. As a result, in the transcript of $\mathcal{D}'$, all the construction query-responses, can be reconstructed from the primitive queries. Hence, it is sufficient to focus our attention on only the primitive queries and compare the distribution of outputs. If $\mathcal{D}$ makes $q_1$ many construction queries and $q_2$ many primitive queries, then $\mathcal{D}$ makes $q_1$ many construction queries and $q_2 + q_1 l$ many primitive queries in total where $l$ is the maximum number of primitive queries to compute $C$.

**The Simulator.** We start with the high-level overview of how the simulator $S$ works. For each $j \in [3]$, $b \in [2^{3-j}]$ the simulator maintains a list $L_{(j,b)}$. The list $L_{(j,b)}$ contains the query-response tuples for the function $f_{(j,b)}$.
Message Reconstruction. The main component of the simulator is the message reconstruction algorithm `FindM`. In the case of traditional Merkle tree, the messages are only injected in the leaf level. We have, in addition, the message injection at each (non-root) internal node. The message reconstruction in our case is slightly more involved.

The algorithm for message reconstruction is the subroutine `FindM`. It takes $(u_0, v_0)$, the input to $f_{(3,1)}$, as input. Let $M = m_1||m_2|| \cdots ||m_{10}$ be the message for which $f_{(3,1)}(u_0, v_0)$ is the hash value. Also, suppose all the intermediate queries to $f_{(j,b)}(j < 3)$ has been made. In the following, we describe how the (partial) messages corresponding to chaining value $u_0$ is recovered. The other half of the message, corresponding to $v_0$, is recovered in analogous way.

Recall that there is no message injection at the final node. Hence, if all the intermediate queries related to $M$ is made by the adversary, then $m_9$ must satisfy all the following relations, $\exists(u, v, y)_{(2,1)} \in L_{(2,1)}$, such that

$$y = u_0 \oplus v \oplus m_9 \qquad (m_1, m_2, u \oplus m_9) \in L_{(1,1)} \qquad (m_3, m_4, v \oplus m_9) \in L_{(1,2)}$$

We find a candidate $m_9$ by xoring $u_0$ with $y \oplus v$ for all the (so far) recorded entries $(u, v, y)_{(2,1)} \in L_{(2,1)}$. To check the validity of the candidate, we check the other two relations. If indeed such query tuples exist, we can recover the message.

Simulation of the functions. For every non-root function $f_{(j,b)}$, $j < 3$, the simulator simulates the function perfectly. Every query response is recorded in the corresponding list $L_{(j,b)}$. The simulation of $f_{(3,1)}$ is a little more involved, albeit standard in indifferentiability proof. Upon receiving a query $(u_0, v_0)$ for $f_{(3,1)}$, the simulator needs to find out whether it is the final query corresponding to the evaluation for a message $M$. Suppose, all other queries corresponding to $M$ has been made. The simulator finds $M$ using the message reconstruction algorithm. If only one candidate message $M$ is found, the simulator programs the output to be $\mathcal{F}(M)$. If the list returned by `FindM` is empty, then the simulator chooses a uniform random string and returns that as output. The first problem, however, arises when there are multiple candidate messages, returned by `FindM`. This implies, there are two distinct messages $M, M'$ for both of which $f_{(3,1)}(u_0, v_0)$ is the final query. The simulator can not program its output to both $\mathcal{F}(M)$ and $\mathcal{F}(M')$. Hence, it aborts. In that case, there is a collision at either $u$ or $v$, implying that the adversary is successful in finding a collision in ABR mode. The probability of that event can indeed be bounded by the results from the previous section. The second problem occurs in the output of non-root functions. Suppose for a $f_{(3,1)}(u_0, v_0)$ query the `FindM` algorithms returns an empty set. Intuitively, the simulator assumes here the adversary can not find a message $M$, for which the final query will be $f_{(3,1)}(u_0, v_0)$. Hence, the simulator does not need to maintain consistency with the Random Oracle. Now the second problem occurs, if later in the interaction, the output of some $f_{(j,b)}$ query forces a completion in the chaining value and a message $M$ can now be recovered for which the final query will be $f_{(3,1)}(u_0, v_0)$. This will create an inconsistency of the simulator's output and the response of the Random Oracle. In the following, we bound the probability of these two events.

The description of the simulator is given in Fig. 5. The message reconstruction algorithm finds a candidate $m_9$ (and resp. $m_{10}$) for each entry in $L_{(2,1)}$ (and resp. $L_{(2,2)}$), and checks the validity against every entry of $L_{(1,1)}$ along with $L_{(1,2)}$ (resp. $L_{(1,3)}$ along with $L_{(1,4)}$). Thus the time complexity of message reconstruction algorithm is $\mathcal{O}(q^2)$. As the simulator invokes the message reconstruction algorithm at most once for each query, we bound $t_s = \mathcal{O}(q^2)$. Similarly, we find $q_s = 1$ as the simulator has to query $\mathcal{F}$ only once per *invocation*.

```
Procedure $S(3,1,u,v)$
1:  if $(u,v,z) \in L_{(3,1)}$  return $z$
2:  $\mathcal{M} = \texttt{FindM}(u,v)$
3:  if $|\mathcal{M}| > 1$return $\perp$
4:  if $|\mathcal{M}| = 0$
5:    $z \xleftarrow{\$} \{0,1\}^n$
6:    $L_{(3,1)} = L_{(3,1)} \cup (u,v,z)$
7:    return $z$
8:  endif
9:  $M \leftarrow \mathcal{M}$
10:  $z = \mathcal{F}(M)$
11:  $L_{(3,1)} = L_{(3,1)} \cup (u,v,z)$
12:  return $z$
```

```
Procedure $S(j,b,u,v)$ where $j < 3$
1:  if $\exists(u,v,z) \in L_{(j,b)}$
2:    return $z$
3:  else
4:    $z \xleftarrow{\$} \{0,1\}^n$
5:    $L_{(j,b)} = L_{(j,b)} \cup (u,v,z)$
6:    return $z$
7:  endif
```

```
Procedure $\texttt{FindM}(u,v)$
// Recovering message from $u$ part
1:  $\mathcal{M}_1 = \emptyset$
2:  for each $(u',v',h') \in L_{(2,1)}$
3:    $m_9 = h' \oplus u \oplus v'$
4:  endfor
5:  if $\exists(m_1,m_2)$ such that $(m_1,m_2,u' \oplus m_9) \in L_{(1,1)}$
    $\wedge\, \exists(m_3,m_4)$ such that $(m_3,m_4,v' \oplus m_9) \in L_{(1,2)}$
6:      $\mathcal{M}_1 = \mathcal{M}_1 \cup (m_1,m_2,m_3,m_4,m_9)$
7:  endif
// Recovering message from $v$ part
8:  $\mathcal{M}_2 = \emptyset$
9:  for each $(u',v',h') \in L_{2,2}$
10:    $m_{10} = h' \oplus v \oplus v'$
11:  endfor
12:  if $\exists(m_5,m_6)$ such that $(m_5,m_6,u' \oplus m_{10}) \in L_{(1,3)}$
    $\wedge\, \exists(m_7,m_8)$ such that $(m_7,m_8,v' \oplus m_{10}) \in L_{(1,4)}$
13:      $\mathcal{M}_2 = \mathcal{M}_2 \cup (m_5,m_6,m_7,m_8,m_{10})$
14:  endif
// Combining the messages
15:  for each$(m_1,m_2,m_3,m_4,m_9) \leftarrow \mathcal{M}_1$
    $\wedge$ each$(m_5,m_6,m_7,m_8,m_{10}) \leftarrow \mathcal{M}_2$
16:  $\mathcal{M} = \mathcal{M} \cup (m_1,m_2,\cdots,m_{10})$
17:  endif
18:  return $\mathcal{M}$
```

**Fig. 5.** Description of the simulator

**The Bad Events.** We shall prove the theorem using the H-coefficient technique. We consider the following Bad events.

BAD0: The set $\mathcal{M}$, returned by the message reconstruction algorithm has cardinality more that one. This implies, one can extract two message $M_1, M_2$ from the transcript such that the computation of $\mathsf{ABR}^+(M_1)$ and $\mathsf{ABR}^+(M_2)$ makes the same query to $f_{(3,1)}$.

BAD1: There exists an $i$, such that for the $i^{th}$ entry in the transcript $h_i = f_{(j,b)}(x_i, y_i)$ with $j < 3$, there exists a message $M$ such that $C^f(M)$ can be computed from the first $i$ entries of the transcript, but can not be computed from the first $i-1$ entries. This in particular implies that there exists a $i'$ with $i' < i$, such that:

- $i'^{th}$ query is a query to $f_{(3,1)}$. $h = f_{(3,1)}(u_{i'}, v_{i'})$

– By setting $h_i = f_{(j,b)}(x_i, y_i)$ with $\ell > 0$, we create a message $M$ such that all the other chaining values of $C^f(M)$ are present in the first $i-1$ queries with $f_{(3,1)}(u_{i'}, v_{i'})$ as the final query.

**Lemma 9.** *For adversary $\mathcal{A}$ making $q$ many queries,*

$$\Pr[\,\text{BAD}\,] \leq \mathcal{O}\left(\frac{n^2q^2}{2^n}\right).$$

**Bounding Pr [ BAD ].** We bound the probabilities of the BAD events.

– **Case** BAD0**:** If there is a collision in the final query of the computations for two different messages, then there is a collision in the $u$ part or $v$ part of the chain. This implies a collision in one of the ABR mode output. Hence, by Proposition 4

$$\Pr[\text{BAD0}] \leq \mathcal{O}\left(\frac{n^2q^2}{2^n}\right)$$

– **Case** BAD1**:** We first consider a query $f_{(j,b)}(u,v)$ with $j = 2$. Let $Y_{(u,v,j,b)}$ denote the yield of this query (recall that yield of a query denotes the number of new chaining values a query creates, see page 17). As there can be at most $q$ many queries to $f_{(3,1)}$ done before this, probability that such a query raises the BAD1 is bounded by $\frac{Y_{(u,v,j,b)}q}{2^n}$. Taking union bound over all the queries at $f_{(j,b)}$, the probability gets upper bounded by $\frac{q\sum Y_{(u,v,j,b)}}{2^n}$. As we showed in the previous section this probability can be bounded by $\mathcal{O}\left(\frac{n^2q^2}{2^n}\right)$. Finally, we consider the case of BAD1 raised by some queries at the leaf level. As in the proof of collision resistance, the expected number of new chaining values created at the output by the leaf level queries is $\frac{nq^3}{2^n}$. Hence, by Markov inequality, the probability that the total number of new chaining values created is more that $q$ is at most $\frac{nq^2}{2^n}$. Finally, conditioned on the number of new chaining values be at most $q$, the probability that it matches with one of the $f_{(3,1)}$ queries is at most $\frac{q^2}{2^n}$. Hence, we get

$$\Pr[\text{BAD1}] \leq \mathcal{O}\left(\frac{nq^2}{2^n}\right)$$

**Good Transcripts Are Identically Distributed.** We show that the good views are identically distributed in the real and ideal worlds. Note that the simulator perfectly simulates $f$ for the internal node. The only difference is the simulation of the final query. In case of good views, the queries to $f_0$ are of two types:

1. The query corresponds to the final query of a distinct message $M$, such that all the internal queries of $C^f(M)$ have occurred before. In this case, the simulator response is $\mathcal{F}(M)$. Conditioned on the rest of the transcript the output distribution remains same in both the worlds.

2. There is no message $M$ in the transcript so far for which this is the final query. In this case, the response of the simulator is a uniformly chosen sample. As Bad1 does not occur, the property remains true. In that case as well, the output remains same, conditioned on the rest of the transcript.

Hence, for all $\tau \in \Theta_{good}$

$$\Pr\left[ X_{\mathtt{real}} = \tau \right] = \Pr\left[ X_{\mathtt{ideal}} = \tau \right]$$

This finishes the proof of Theorem 6.

**Corollary 2.** *The compactness of $\mathsf{ABR}^{+}$ making $r$ calls to underlying $2n$-to-$n$-bit function is* $1 - \frac{2}{3r-1}$.

## 6 Efficiency and Applications

In this section, we discuss the compactness of our proposed designs, possible applications and use cases.

### 6.1 Efficiency and Proof Size

Below we discuss and compare our designs with the Merkle tree regarding efficiency of compression and authentication and proof size: the number of openings to prove a membership of a node in a tree.

Efficiency of compression and authentication. To measure efficiency of compression we consider the amount of message (in bits) processed for a fixed tree height or a fixed number of compression function calls. As mentioned earlier, compared to a Merkle tree of height $\ell$ which absorbs $n2^{\ell}$ message bits, the $\mathsf{ABR}$ or $\mathsf{ABR}^{+}$ modes process an additional $n(2^{\ell-1}-1)$ message bits. Thus, asymptotically the number of messages inserted in our $\mathsf{ABR}$ (or $\mathsf{ABR}^{+}$) mode increases by 50% compared to Merkle tree. Additionally, the cost of authentication (number of compression function calls to authenticate a node) in a Merkle tree is $\log N$ where $N = 2^{\ell}$. Here as well the $\mathsf{ABR}$ or $\mathsf{ABR}^{+}$ modes compress 50% more message bits compared to Merkle tree keeping the same cost of authentication as in Merkle tree as shown in Lemma 10.

Proof Size. We refer to the tree chaining and internal message nodes as the tree *openings*. The proof size in a tree is determined by the number of openings. In a Merkle tree, the proof of membership of *all* (leaf) inputs requires $\log N$ compression function evaluations and openings each. More precisely, to prove the *membership of an arbitrary leaf input*, $\log N - 1$ chaining values and one leaf input are required. Note that while counting the number of openings, we exclude the input for which the membership is being proved.

**Lemma 10.** *In $\mathsf{ABR}$ mode, to prove the membership of any node (message block): leaf or internal, we require* $2 \log N - 1$ *($n$-bit) openings and* $\log N$ *compression function computations.*

*Proof.* To prove the membership of a leaf input in the $\mathsf{ABR}$ mode $2(\log N - 1)$ openings are required together with one leaf input. This makes a total of $2\log N - 1$ openings. To obtain the root hash $\log N$ computation must be computed. To prove the membership of an internal node, we need $2(\log N - 1)$ openings, excluding any openings from the level at which the internal node resides. Additionally, one more opening is required from the level of the node. Thus, in total we need again $2\log N - 1$ openings. The number of compression calls remains $\log N$.

Compared to Merkle tree, in $\mathsf{ABR}^+$ the proof size increases by $\log N - 1$. Admittedly, for Merkle tree applications where the proof size is the imperative performance factor, the $\mathsf{ABR}^+$ modes do not provide an advantage.

## 6.2 Applications and Variants

**ZK-SNARKs.** We briefly point out here the potential advantages of using the $\mathsf{ABR}$ mode in zk-SNARKS based applications, such as Zcash. In a zk-SNARK [22] based application, increasing the number of inputs or transactions in a block means that we need to increase the size of the corresponding Merkle tree. The complexity of the proof generation process in zk-SNARK is $C \log C$ where $C$ is the circuit size of the underlying function. In $\mathsf{ABR}^+$ modes the additional messages are inserted without increasing the tree height or introducing additional compression function calls. Since the messages are only injected with xor/addition operation, this does not deteriorate the complexity of the proof generation. Zcash uses a Merkle tree with height $\approx$29 and $2^{34}$ byte inputs. By using either one, $\mathsf{ABR}$ or $\mathsf{ABR}^+$ modes, an additional of $\approx 2^{33}$ byte inputs can be compressed *without* making any extra calls to the underlying compression function. Asymptotically, $\mathsf{ABR}$ or $\mathsf{ABR}^+$ provides 50% improvement in the number of maintained (in the tree structure) messages compared to a Merkle tree.

Further Applications. Our modes can be useful in applications, such as hashing on parallel processors or multicore machines: authenticating software updates, image files or videos; integrity checks of large files systems, long term archiving [17], content distribution, torrent systems [1], etc.

Variants. We continue with possible variants of utilizing the $\mathsf{ABR}$ compression function in existing constructions, such as the Merkle–Damgård domain extender and a 5-ary Merkle tree, and discuss their compactness and efficiency.

**Merkle–Damgård (MD) Domain Extender with $\mathsf{ABR}$.** When the compression function in MD is substituted by $\mathsf{ABR}$ ($\ell = 2$) compression function, the collision resistance preservation of the original domain extender is maintained. We obtain compactness of $\approx 8/9$ of such an MD variant (see Sect. 3.1).

For all our modes, the high compactness allows us to absorb more messages at a fixed cost or viewed otherwise, to compress the same amount of data (e.g. as MD or Merkle tree) much cheaper. We elaborate on the latter trade-off here. To compress 1 MB message with classical MD that produces a 256-bit hash value and uses a 512-to-256-bit compression function, around 31250 calls to the underlying (512-to-256-bit) compression function are made. In contrast, ABR in MD requires just ≈7812 calls to the (512-to-256) compression function, that is an impressive 4-fold cost reduction.

**5-ary Merkle Tree with ABR.** One can naturally further construct a 5-ary Merkle tree using ABR with compactness <8/9 (see Sect. 3.1). That means to compress 1MB data with a 5-ary ABR mode with $5n$-to-$n$-bit ($n = 256$) compression functions will require ≈23437 calls to the 512-to-256-bit compression functions. Using the Merkle tree the number is 31250 compression function calls. On the other hand, the ABR and $\mathsf{ABR}^+$ modes require *only* ≈20832 calls.

We have also considered simpler versions of ABR (e.g. when the feed forward from $f_2$ is omitted) to show how they fail to achieve collision security (in the extended version [2]).

## 7 Discussion and Conclusions

The ABR mode is the first collision secure, large domain, hash function that matches Stam's bound for its parameters. The ABR+ is also close to optimally efficient and achieves the stronger indifferentiablity notion, both completed in the ideal model. Based on our security results we can conclude that the $\mathsf{ABR}^+$ mode is indeed the stronger proposal that achieves all the 'good' function properties up to the birthday bound. Driven by practical considerations for suitable replacements of Merkle tree, the ABR mode appears to be the more natural choice. This is motivated by the fact that the majority of Merkle tree uses are indeed FIL, namely they work for messages of fixed length.

Indeed, for such FIL Merkle trees collision preservation in the standard model holds but it fails once message length variability is allowed (for that one needs to add MD strengthening and extra compression function call). The ABR mode is proven collision secure in the ideal model. Our result confirms the structural soundness of our domain extenders in the same fashion as the Sponge domain extender does it for the SHA-3 hash function.

We clarify that simple modifications of ABR lead to the same security results. These variants are when one uses for feed-forward the left chaining value (instead of the right as in the ABR mode) *or* when the internal message itself (instead of the right chaining value) are fed-forward into the output of $f_0$. The collision security proofs for these two variants follow exactly the same arguments and are identical up to replacement for the mentioned values. Similarly, an extended tree version of the latter constructions can be shown collision or indifferentiability secure when it is generalized in the same fashion as the $\mathsf{ABR}^+$ mode.

An interesting practical problem is to find and benchmark concrete mode instantiations. From a theory perspective, finding compact double length constructions is an interesting research direction.

**Acknowledgements.** We thank Martijn Stam for reading an earlier version of the draft and providing valuable comments. We would like to also thank Markulf Kohlweiss for discussion on the zksnarks and other applications of this work. We sincerely thank the reviewers of this and the earlier version of this paper for their insightful comments. Rishiraj is supported by *SERB ECR/2017/001974*.

## References

1. http://bittorrent.org/beps/bep_0030.html
2. Andreeva, E., Bhattacharyya, R., Roy, A.: Compactness of hashing modes and efficiency beyond Merkle tree. IACR Cryptol. ePrint Arch. (2021)
3. Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the Grøstl hash function. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_7
4. Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 39–53. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18178-8_5
5. Andreeva, E., Mennink, B., Preneel, B.: The parazoa family: generalizing the sponge hash functions. Int. J. Inf. Sec. **11**(3), 149–165 (2012)
6. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. IACR Cryptol. ePrint Arch. **2014**, 349 (2014)
7. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von Neumann architecture. In: Fu, K., Jung, J. (eds.) USENIX Security 2014, pp. 781–796. USENIX Association (August 2014)
8. Benjamin, D.: Batch signing for TLS (2019). https://tools.ietf.org/html/draft-davidben-tls-batch-signing-02
9. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS$^+$ signature framework. In: ACM CCS 2019, pp. 2129–2146. ACM Press (November 2019)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. Cryptology ePrint Archive, Report 2011/499 (2011). http://eprint.iacr.org/2011/499
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_19
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_11
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_19
14. Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient Blockcipher-based hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_31
15. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_21

16. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: A Lightweight Hash Function. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_21
17. BSI. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03125/TR-03125_M3_v1_2_2.pdf
18. Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., Vuillaume, C.: Merkle signatures with virtually unlimited signature capacity. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 31–45. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_3
19. Buchmann, J., García, L.C.C., Dahmen, E., Döring, M., Klintsevich, E.: CMSS – an improved Merkle signature scheme. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 349–363. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_25
20. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_19
21. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
22. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
23. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. J. Cryptol. **3**(2), 99–111 (1991). https://doi.org/10.1007/BF00196791
24. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_2
25. Mennink, B., Preneel, B.: Hash functions based on three permutations: a generic security analysis. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 330–347. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_20
26. Mennink, B., Preneel, B.: Efficient parallelizable hashing using small non-compressing primitives. Int. J. Inf. Secur. **15**(3), 285–300 (2015). https://doi.org/10.1007/s10207-015-0288-7
27. Merkle, R.C.: Protocols for public key cryptosystems. In: Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, 14–16 April 1980, pp. 122–134. IEEE Computer Society (1980)
28. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
29. Nandi, M.: A simple and unified method of proving indistinguishability. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 317–334. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_23
30. Patarin, J.: The "Coefficients H" technique. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_21

31. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_27
32. Ristenpart, T., Shrimpton, T.: How to build a hash function from any collision-resistant function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 147–163. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_9
33. Rivest, R.L., Schuldt, J.C.N.: Spritz - a spongy RC4-like stream cipher and hash function. IACR Cryptol. ePrint Arch. **2016**, 856 (2016)
34. Rogaway, P., Steinberger, J.: Constructing cryptographic hash functions from fixed-key blockciphers. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_24
35. Rogaway, P., Steinberger, J.: Security/efficiency tradeoffs for permutation-based hashing. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_13
36. Shrimpton, T., Stam, M.: Building a collision-resistant compression function from non-compressing primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_52
37. Stam, M.: Beyond uniformity: better security/efficiency tradeoffs for compression functions. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_22
38. Steinberger, J.: Stam's collision resistance conjecture. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 597–615. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_30
39. Steinberger, J., Sun, X., Yang, Z.: Stam's conjecture and threshold phenomena in collision resistance. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 384–405. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_23
40. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–304. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_19