Revisiting Collision and Local Opening Analysis of ABR Hash

Chandranan Dhar ✉
Indian Statistical Institute, Kolkata, India

Yevgeniy Dodis ✉
New York University, NY, USA

Mridul Nandi ✉
Indian Statistical Institute, Kolkata, India

Abstract

The question of building the most efficient tn-to-n-bit collision-resistant hash function H from a smaller (say, $2n$-to-n-bit) compression function f is one of the fundamental questions in symmetric key cryptography. This question has a rich history, and was open for general t, until a recent breakthrough paper by Andreeva, Bhattacharyya and Roy at Eurocrypt'21, who designed an elegant mode (which we call ABR) achieving roughly $2t/3$ calls to f, which matches the famous Stam's bound from CRYPTO'08. Unfortunately, we have found serious issues in the claims made by the authors. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proofs (e.g., omissions of important cases and unjustified bounds).

We were unable to patch up the current proof provided by the authors. Instead, we prove from scratch the security of the ABR construction for the first non-trivial case $t = 11$ (ABR mode of height 3), which was incorrectly handled by the authors. In particular, our result matches Stam's bound for $t = 11$. While the general case is still open, we hope our techniques will prove useful to finally settle the question of the optimal efficiency of hash functions.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases ABR hash, collision resistance, local opening

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.11

Funding *Yevgeniy Dodis*: Partially supported by gifts from VMware Labs and Google, and NSF grants 1815546 and 2055578.
Mridul Nandi: Partially supported by the project "Study and Analysis of IoT Security" under Government of India at R.C.Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata.

1 Introduction

The *Merkle-Damgård construction* [3, 7] is a sequential construction which is used in MD5, SHA-1 and SHA-2 and many other hash functions. On the other hand, the *Merkle tree* [6] is a parallel construction that is used in hash-based signatures (of interest due to their post-quantum security), version control systems such as git, and cryptocurrencies such as Ethereum. It is well known that the Merkle-Damgård construction and the Merkle tree are collision-resistant provided so are the compression functions. The number of compression function calls is (essentially) the same for both constructions. When we use $2n$-to-n-bit compression functions, we can process t blocks of messages by making t or $(t-1)$ calls to the compression function.

Although both of these widely used constructions are rather efficient, and only rely on the collision-resistance of the compression function, practical compression functions are believed to have more properties than mere collision resistance. As such, it is interesting to study the question of designing the most efficient way to build a t-to-1 collision-resistant hash function, even if modeling the compression function as ideal (i.e. a random oracle). In particular, to see whether the classical Merkle-Damgård and Merkle tree constructions can be improved under such idealized modeling. This question has received a lot of attention from the cryptography community, which we survey below.

Lower Bound on the Number of Calls. We start with lower bounds (i.e., attacks). In [2], Black et al. 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. Later Rogaway and Steinberger [9] generalized the result for permutation-based hash. For a general hash function based on a compression function, Stam [11] conjectures a lower bound on the number of compression function calls. In particular, a collision with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function can be found after making r calls to λ-to-1 block compression functions. Equivalently, for optimal birthday security, the number of hash calls must be at least $r \geq (2t-1)/(2\lambda-1)$. This bound is popularly known as the Stam's bound. Stam has shown the bound for some cases under a uniformity assumption. Later by Steinberger [12] and by Steinberger, Sun and Yang [13], a formal proof of the Stam's bound is shown.

Hence, for the most widely studied case of $\lambda = 2$, we have a lower bound $r \geq (2t-1)/3$, leaving a factor 1.5 efficiency gap when compared to the Merkle-Damgård and Merkle trees.

Upper Bound on the Number of Calls. For the upper bounds, much of earlier work concentrated on the setting of the "non-compressing" case of $\lambda = 1$, and often focused on the case of small t (e.g., $t = 2$), implicitly suggesting that – once the 2-to-1 function is built, – one should do further extensions with either Merkle-Damgård and Merkle trees. For example, Shrimpton and Stam [10] proposed a 2-to-1 compression function based on three calls of non-compressing function, which matches Stam's bound for $\lambda = 1$ and $t = 2$. Rogaway and Steinberger [8] designed similar results when the non-compressing primitive is an invertible permutation, which they also showed is optimal for this setting [9].

For general (large) t, Mennink and Preneel [5] also considered the non-compressing case $\lambda = 1$ and proposed an elegant tree-based mode of operation making $(2t-1)$ calls to the non-compressing round function, which matches Stam's bound. Unfortunately, they could only prove below-birthday security of $2^{n/3}$ queries for this construction. They also conjectured that the construction achieves optimal birthday security $2^{n/2}$, but could only prove it for a very restricted special-case attacker. These attacks make all their random oracle calls "layer-by-layer" (as opposed to in any order). As acknowledged by the authors, the simplifying assumption significantly helps with the proof of this special case and appears to be with a great loss of generality. In fact, they presented evidence that their existing analysis is unlikely to work for proving optimal security against unrestricted attackers.

Recently, two papers have appeared to tackle the compressing case $\lambda = 2$. In [4], Dodis et al. optimally settled the case $t = 5$, by introducing the $T5$ construction that processes five n-bit message blocks using three $2n$-to-n-bit compression function calls, which matches Stam's bound for $t = 5$ and $\lambda = 2$. Further, they suggested extending the $T5$ construction to a larger value of t using either Merkle-Damgård or Merkle trees. In both cases, they already achieve non-trivial saving compared to the earlier efficiency of these modes (equal to t compression calls): both variants now make roughly $3t/4$ calls to the compression functions. Still, once $t > 5$, this does not match the current lower bound of $2t/3$ calls. [4] also mentioned a natural, but more aggressive, variant of this extended construction for the case of Merkle trees. However, they remark that this construction – even if proven collision-resistant (which is open), – would lose the efficient "local opening" properties of their simpler tree construction with $3t/4$ compression calls. Namely, one can no longer open one message block by only opening $O(\log t)$ internal values in the tree (as any such opening cannot have birthday security, despite satisfying correctness).

Finally, a breakthrough result of Andreeva, Bhattacharyya and Roy at Eurocrypt'21 [1] have claimed to settle the general case in the affirmative. They proposed a hash function ABR_l based on a perfect binary tree of height l. The hash ABR_l can process $t = (2^l + 2^{l-1} - 1)$ blocks with $r = (2^l - 1)$ calls of compression functions. This matches Stam's bound $r \geq (2t-1)/3$. Somewhat interestingly, the ABR construction looks very similar to the tree construction of Mennink and Preneel [5] from non-compressing primitives, except all the compression calls at the leaf level now have an extra input (due to $\lambda = 2$ instead of $\lambda = 1$), while the internal calls to the compression function can also process an extra input, but using a slightly trickier rule involving two simple XOR operations. So, at least in the intuitive sense, the authors must have resolved the difficulty of [5] of dealing with general adversaries, for a construction very similar to the one of [5].

As an additional bonus feature, the work of [1] even claimed that the ABR_l mode also has attractive local opening properties, at the expense of slightly longer proof length ($2l$ instead of l of Merkle trees), but still having only l compression calls to verify such local opening.

Are We Done? Unfortunately, we have found serious issues in many claims made by the authors of [1], whom we call ABR hereafter. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proof (e.g., omissions of important cases and unjustified bounds). Unfortunately, at this stage, we are unable to fix these issues in any simple way.

1.1 Our Results

Our results can be roughly divided into 3 categories:

(1) explicit refutation of some claims made by [1];

(2) serious technical issues in the proof provided by [1];

(3) a correct (but very different from [1]) proof for the for the ABR_3 construction (i.e. $t = 11$ and $r = 7$), which is incorrectly handled by ABR.

We detail these below.

Local Opening Insecurity of ABR. As we mentioned, ABR proposed a very efficient local opening for ABR_l. It opens about $2l$ blocks and makes l calls to verify. However, we have shown that a collision pair of the verification function can be found in $O(2^{n/2l})$ queries, which is significantly below birthday security already for $l = 2$. Hence, the suggested local opening can be broken in the above complexity. Moreover, we have shown that *any* non-trivial local opening of ABR_l satisfying a "by-pass verification" property (which is a natural class of openings that seems to include any natural opening one can think of) is broken below the birthday bound. For example, even opening $(t-1)$ out of t inputs cannot be birthday-secure, where $t = 2^l + 2^{l-1} - 1 = 2^{\Omega(l)}$. In contrast, previous tree-like constructions (e.g., [4]) achieve birthday security with logarithmic opening length $O(l)$. This is discussed in Section 4.

There are two surprising aspects to this mistake. First, our attack is completely standard (using standard generalized birthday attack [14]). Second, the local opening subsection in the ABR-paper does not even mention anything about security, only focusing on the correctness of the opening. We found this quite surprising.

Mistakes with the Main Proof. While the local opening mistake above is indisputable, the technical mistakes in the main collision resistance proof of ABR are harder to explain in detail (at least in the Introduction, before the technical notation is developed). They are also harder to state with conviction, since they often do a combination of the following pitfalls:

(a) involve imprecise statements,
(b) state a bound which might be true, but which appears completely non-obvious to us (to the extent of being the most difficult part of the proof);
(c) point to an "analogous" earlier case, but we fail to see why the previous argument generalizes;
(d) state some bound which appears to be correct only if one makes some restricting assumption on the attacker (but no such assumptions are made by the authors, who claim a fully general result!);
(e) silently omitting an important special case of the proof (i.e., the proof is non-exhaustive).

The totality of these issues make the proof presented by [1] at best unverifiable, and at worst incorrect. In particular, we still believe that the end result is correct, but fixing it would require a substantially harder proof.

At a very high level, the correct collision analysis for a tree-based function like ABR_l is complex mostly due to the *adaptive nature* of queries, and the queries made to different layers in the tree might not come in monotone order (i.e., may not be in order of the level of the nodes). Indeed, this is precisely the reason why the earlier birthday security result of Mennink and Preneel [5] only held for "in order" adversaries. Fortunately, the outputs of the leaf nodes can be given beforehand, as the input of those has no role in finding a collision. More formally, we can make a simple argument to force the attacker "evaluate" the first layer compression calls before any of the subsequent calls as follows. We give the attacker q random outputs (where q is the total number of queries made by the attacker) at the very beginning, but allow the adversary to *arbitrarily label* the corresponding input values at any point in the game. This is fine, since those input values do not participate in any other computation, but now all the outputs in the first layer are known before a single compression call is made to the lower layers. This allows for relatively simple analysis for the special case $l = 2$, and the authors of [1] indeed start with the correct analysis of this special case.[1]

Unfortunately, this argument completely fails after the first layer. (Indeed, handling this case will be one of the most difficult parts of our analysis, when we provide a correct proof for $l = 3$ in this paper.) In particular, we see the following high-level issues with the proof presented by [1] for $l \geq 3$. (More lower-level issues are discussed in Section 5.3 in the paper.)

1. ABR claimed a relation between collision and the number of computable hash outputs (termed as load). We will show in Section 5.4 that the relation is not true in general by giving a counterexample. This seems to hold for ABR if queries to the root node are performed at the end (which is the case for ABR_2). However, it seems non-obvious to us why a similar relation holds when the adversary makes out-of-order queries.
2. We have also found issues while bounding load. ABR consider "input multi-collision" for every node up to $O(n)$. However, due to the multiplicative nature of the number of multi-collisions as one goes down in the tree, we find that $O(n^i)$ multi-collision must be considered for the nodes at the i-th level. This would degrade the bound for load

[1] Another correct proof for $t = 5$ (corresponding to tree depth $l = 2$) was made for the $T5$ compression function by [4]. Interestingly, the authors did not notice the simplifying non-adaptivity argument above, and had to work relatively hard to handle out-of-order queries (e.g., it involved a careful expectation analysis and applying Markov's inequality; see proof of Proposition 5 in [4], which is over a page). This shows that handling out-of-order attackers is indeed highly non-trivial.

claimed by ABR, and invalidate the claimed birthday security at the end (unless the number of levels i is constant, in which case one can hide the extra n^i bound in the "O-tilde"-notation). This will be discussed in Step 1 of Section 5.3.

3. In fact, even if the load analysis is somehow fixed, ABR seem to consider the last query happens in the final node (or at the node where the load is considered). This is effectively equivalent to in order adversaries, but does not seem to be the case for general attackers. See Step 2 of Section 5.3.
4. Moreover, both messages of a collision pair can be generated due to a single query response (termed as *twin collision pair*). ABR completely ignore this case. This is discussed in detail in the last paragraph of Section 5.3.

We leave a more detailed explanation of these (and other issues (a)-(e)) later in the paper.

Collision Analysis of ABR_3. On a positive, our main technical result shows that the ABR_3 construction for $t = 11$ indeed achieves birthday security (roughly $n^5q^2/2^n$, where q is the number of compression function queries) with an optimally small number of $r = 7$ compression calls (see Section 6). While forming only the first step in recouping the incorrect results of [1], we are optimistic that our approach could be extended to finally settle the general case correctly. For example, compared to best known correct proofs for $t = 5$ (e.g., ABR_2 from [1], or the $T5$ compression function from [4]), we can no longer assume that the second layer calls to the compression function are made before all the third-layer calls, which is the main (unresolved) difficulty in the work of [5], and one of the key mistakes in the analysis of [1] (as we explained above). Thus, our proof is the first which handles non-trivial "out-of-order" adversaries correctly.

We also hope our proof of ABR_3 provides a sharp contrast to the flawed proof of [1], even for this special case. For example, we already mentioned handling general "out-of-order" adversaries. In a different vein, we also consider the twin-collision analysis for ABR_3 which is completely missing from [1]. This analysis requires a non-trivial multi-collision analysis on a sum of our compression functions, and we also need to bound some other failure events to analyze the non-twin collision security of ABR_3. None of these arguments appeared in [1].

2 Security Definitions

2.1 Notations

We call elements of $\{0,1\}^n$ blocks. A k-to-r (block) function or random oracle has domain $\{0,1\}^{kn}$ and range $\{0,1\}^{rn}$. We write the set $[k] = \{1, 2, \dots, k\}$. A partial function τ from D to R is a subset $\tau \subseteq D \times R$ such that for every $x \in D$, there are at most one y with $(x, y) \in \tau$. We define domain $\mathrm{dom}(\tau) := \{x : \exists y, (x, y) \in \tau\}$ and range $\mathrm{ran}(\tau) = \{y : \exists x, (x, y) \in \tau\}$ of a partial function τ. We use the shorthand notation $A \cup x$ and $A \setminus x$ to denote $A \cup \{x\}$ and $A \setminus \{x\}$ respectively. For any q-tuple x^q, we define $\mathrm{mc}(x^q) = \mathsf{max}_a |\{i : x_i = a\}|$. For two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, we define $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) = \mathsf{max}_a |\{(i, i') : L_i \oplus L_{i'} = a, L_i \in \mathcal{L}_1, L_{i'} \in \mathcal{L}_2\}|$. It can be similarly extended for xor of more than two lists.

2.2 Generic Hash Mode

Let $\mathtt{H}^f$ be a t-to-1 hash function which uses an n-bit compression function (i.e. λ-to-1 compression function f for some $\lambda > 1$) as an oracle. Note that a mode can use more than one compression functions $f_1, \dots, f_r$. However, as we analyze in the random oracle model, independent random oracles can be obtained from a single random oracle with a little bit larger domain by using the standard domain separation method. In this paper, we only consider fixed-length input and also assume r is the same for all messages. Moreover, the hash function calls f_i on i-th call and so the domains of every call are separated by domain separation. We also denote the family $f := (f_i : i \in [r])$ by f and we call λ-to-1 r r.o. (random oracle). We denote $\tau_{\mathtt{H}}(M \mid f) := \{((1, x_1), y_1), \ldots, ((r, x_r), y_r)\}$ where x_i denotes the input of i-th call of its oracle tuple while computing $\mathtt{H}^f(M)$ and $y_i = f_i(x_i) := f(i, x_i)$. A λ-to-1 **transcript** τ is a partial function from $[r] \times \{0,1\}^{\lambda n}$ to $\{0,1\}^n$. For a λ-to-1 r r.o. f, we have

$$\forall (i, x) \notin \mathrm{dom}(\tau), y \in \{0,1\}^n, \quad \mathrm{Prob}(f(i, x) = y \mid \tau \subseteq f) = 2^{-n}.$$

▶ **Definition 1** (transcript-based hash computation). *Given a partial function $\tau \subseteq f$, let $\mathtt{H}^\tau = \{(M, \mathtt{H}^f(M)) : \tau_{\mathtt{H}}(M \mid f) \subseteq \tau\}$ be a partial hash function. In other words, $\mathtt{H}^\tau$ consists of all pairs (M, z) such that $\mathtt{H}^f(M)$ can be computed by simply using the transcript τ and z is the final value. The elements of the set $\mathrm{dom}(\mathtt{H}^\tau)$ are called τ-computable messages. As $\tau \subseteq f$, we must have $\mathtt{H}^\tau \subseteq \mathtt{H}^f$.*

2.3 Collision Game

Let $\mathcal{A}$ be an adversary having oracle access of f which makes q queries to each f_i adaptively. As we assume an unbounded time adversary, there is no loss in assuming that $\mathcal{A}$ is deterministic. Thus, the i-th query (x_i, v_i) of $\mathcal{A}$ depends on τ^{i-1} (the transcript of query-responses after $(i-1)$ queries). After the query-response phase, $\mathcal{A}$ returns a pair of distinct messages (M, M') such that both M, M' are transcript-computable. We say $\mathsf{coll}_{\mathtt{H}}$ holds if $\mathtt{H}^\tau(M) = \mathtt{H}^\tau(M')$, called a *computable collision pair.* We define $\mathbf{Adv}^{\mathsf{coll}}_{\mathtt{H}^f}(\mathcal{A}) := \Pr(\mathsf{coll}_{\mathtt{H}})$.

▶ **Definition 2** (cross collision). *Let $\mathtt{H}$ and $\mathtt{H}'$ be two hash functions. A* cross-collision τ-computable pair *is a pair (M, M') (not necessarily distinct) such that $\mathtt{H}^\tau(M) = \mathtt{H}'^\tau(M')$. We denote $\mathsf{coll}^\tau_{\mathtt{H},\mathtt{H}'} := \{M \in \mathrm{dom}(\mathtt{H}^\tau) : \exists M',\ \mathtt{H}^\tau(M) = \mathtt{H}'^\tau(M')\}$.*

2.4 Local Opening

We now define the local opening security of a hash function output (viewed as a commitment of a message). Given a hash function mode $\mathtt{H}^f$, a local opening Open^f for $\mathtt{H}$ maps a pair (M, i) to $\pi = (m_i, i, \pi')$ (called proof) where $M = (m_1, m_2, \ldots, m_c)$ is a message (a tuple of blocks) and $i \in [c]$ is an index.

Correctness of Local Opening. There is an efficient function Ver^f such that for all message M, all index i, $\mathsf{Ver}^f(\mathsf{Open}^f(M, i), \mathtt{H}^f(M)) = 1$.

Security of Local Opening. In the local opening security, the adversary wins if it produces an output h corresponding to two contradicting local openings for some position i.

▶ **Definition 3** (local opening advantage). *Let $\mathtt{H}$ be a hash function and Open be a correct local opening for $\mathtt{H}$ with verification function Ver. For any adversary $\mathcal{A}$, we define the local opening advantage as*

$$\mathbf{Adv}^{local}_{H}(\mathcal{A}) = \Pr\Big[\mathsf{Ver}(i, m, \pi, h) = \mathsf{Ver}(i, m', \pi', h) = 1, m \neq m' \\ \mid (i, m, m', \pi, \pi', h) \leftarrow \mathcal{A}^f\Big]$$

By-Pass Hash Computation. We say that $\mathtt{H}$ has a *by-pass computation* $(\mathtt{H}_i : i \in [c])$ corresponding to a local opening Open if for all M, $i \in [c]$,

$$\mathtt{H}_i^f(\mathsf{Open}^f(M,i)) = \mathtt{H}^f(M).$$

In other words, given a proof (output of the Open) and the message block for the index (for which the proof is produced), we can compute the hash output of the message (without knowing the other blocks of the message). The verification algorithm simply checks whether the hash value computed through the by-pass hash is the same as what was committed before. As f is treated as an oracle, it is natural to assume that for all M and for all i,

$$\tau_{\mathsf{Open}}(M, i \mid f) \cup \tau_{\mathtt{H}_i}(\mathsf{Open}^f(M,i) \mid f) = \tau_{\mathtt{H}}(M \mid f).$$

We now define the *inter-collision* advantage for by-pass computation $\mathtt{H}_i$ as

$$\mathbf{Adv}_{\mathtt{H}_i}^{\mathsf{coll}*}(\mathcal{A}) = \Pr\left[\mathtt{H}_i(m,\pi) = \mathtt{H}_i(m',\pi') \text{ and } m \neq m' \mid (m,\pi,m',\pi) \leftarrow \mathcal{A}^f\right].$$

Thus, it is the same as the collision game, except that the adversary needs to find a collision pair for which $m \neq m'$. Suppose $\mathcal{A}$ finds a collision pair $((m,\pi),(m',\pi'))$ for $\mathtt{H}_i$, and let $h = \mathtt{H}_i(m,\pi)$. Then $\mathcal{A}$ can commit h and later on, it can successfully open for either of two messages m and m' as required. Now we make the following simple observation

$$\mathbf{Adv}_{\mathtt{H}}^{local}(q') = \max_{\mathcal{A}} \max_{i} \mathbf{Adv}_{\mathtt{H}_i}^{\mathsf{coll}*}(\mathcal{A}). \tag{1}$$

The above observation (see [4] for details) helps us to reduce the local opening security to inter-collision security problem for the by-pass hash family.

2.5 Stam's Tradeoff between Security and Performance

Stam's bound states that there always exists a collision attack with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function making r calls to λ-to-1 block compression functions.

3 Re-introduction of the ABR Hash due to [1]

We first start by defining a generalized tree hash structure, and then re-introduce the ABR Hash as a special tree hash, as opposed to introducing as it is in [1]. This is because we feel some things have not been properly defined by the authors there, and these issues need to be addressed properly.

A *full binary tree* (FBT) is a binary tree in which every node v other than the leaves has two children, denoted as v_{L} (left child) and v_{R} (right child). A *perfect binary tree* (PBT) is a full binary tree in which all the leaf nodes are at the same level (called height of the tree).

▶ **Example 4** (perfect binary tree of height l)**.** Let l be a fixed positive integer and $\mathcal{T}$ be a perfect binary tree of height l over all vertices (j,b), $j \in [l], b \in [2^{l-j}]$ with $(l,1)$ being the root. For every two vertices (j,b) and $(j+1, \lceil b/2 \rceil)$, we associate an edge. We call $(j-1, 2b-1)$ and $(j-1, 2b)$ the left and right child of (j,b) respectively. Note that $\mathcal{T} = \mathcal{T}_{(l,1)}$.

3.1 Some Notations and Definitions on Binary Trees

For a binary tree $\mathcal{F}$, let $\mathcal{L}_{\mathcal{F}}$ and $V(\mathcal{F})$ denote the set of leaf nodes and all nodes of $\mathcal{F}$ respectively. Any non-leaf node is called an intermediate node. For a non-root intermediate node v of $\mathcal{F}$, we consider the following two full binary trees:

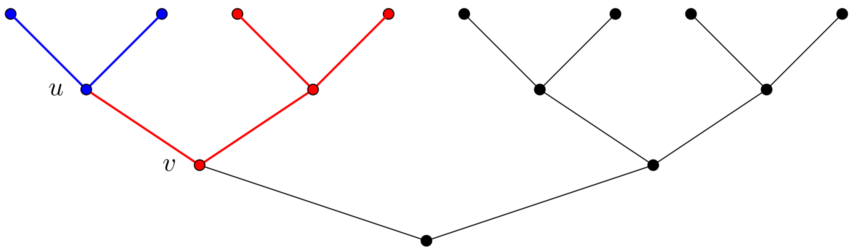

Figure 1 In this figure, $\mathcal{F}_v$ is the sub-tree rooted at v, i.e. the union of the red and blue sub-trees, $\mathcal{F}_{-v}$ is the black sub-tree, and $\mathcal{F}_{v-u}$ is the red sub-tree.

1. $\mathcal{F}_v$: the full binary sub-tree rooted at v.
2. $\mathcal{F}_{-v}$: the sub-tree $(\mathcal{F} \setminus \mathcal{F}_v) \cup v$.

For a tree $\mathcal{F}$, and a vertex v of $\mathcal{F}$, we write V_v, $\mathcal{L}_v$ and V_v^* to denote the set of all nodes, leaf nodes and intermediate (non-leaf) nodes respectively for the tree $\mathcal{F}_v$. For any $u \in V_v^* \setminus v$, we write $\mathcal{F}_{v-u} = (\mathcal{F}_v \setminus \mathcal{F}_u) \cup u$. We write V_{v-u} to denote the set of vertices of $\mathcal{F}_{v-u}$. For the sake of notational simplicity we ignore the suffix v when v is the root. In this section we only consider trees of the form $\mathcal{F}_v$ and $\mathcal{F}_{v-u}$. Refer to Figure 1 for a pictorial representation.

To each node $v \in V$ of a perfect binary tree $\mathcal{T}$, an independent 2-to-1 block compression function (modeled as a random oracle) f_v is assigned. We use the notation f to denote the collection of random oracles $\{f_v : v \in \mathcal{T}\}$.

▶ **Definition 5** (message for tree hash). *A message m for any full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ having the same root is a function $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $u \in \mathcal{L}_{\mathcal{F}} \cap \mathcal{L}_{\mathcal{T}}$, $m(u) \in \{0,1\}^{2n}$, otherwise, $m(u) \in \{0,1\}^n$. A complete message m is a message at the root of $\mathcal{T}$.*

Thus, for every leaf node of $\mathcal{F}$ (which is also a leaf node of the perfect binary tree), we associate $2n$ bit messages. For all other vertices, we associate an n bit message. We write $\mathbb{M}_{\mathcal{F}}$ to denote the set of all messages for $\mathcal{F}$. We simply write $\mathbb{M}_v$ and $\mathbb{M}_{v-u}$ instead of $\mathbb{M}_{\mathcal{T}_v}$ and $\mathbb{M}_{\mathcal{T}_{v-u}}$ respectively.

For a message m for $\mathcal{T}_v$ (also called m at the node v), and $u \in V_v$, we write $m|_u = m|_{\mathcal{T}_u}$, the message restricted to $\mathcal{T}_u$. Similarly, we write $m_{\mathsf{L}} := m|_{v_{\mathsf{L}}}$ and $m_{\mathsf{R}} := m|_{v_{\mathsf{R}}}$. We also write $m|_{v-u\to h}$ to denote a message for $\mathcal{T}_{v-u}$ which is same as the restricted function $m|_{\mathcal{T}_{v-u}}$, except at u, where it assigns h (instead of $m(u)$). In the context of our work, this basically means we replace the message $m(u)$ at node u by the intermediate hash output of $\mathcal{T}_u$, the tree rooted at u, and consider the message for the remaining tree, $\mathcal{T}_{v-u}$.

▶ **Definition 6** (Generalized Tree Hash). *Let $\mathcal{F}$ be a full binary sub-tree of a perfect binary tree $\mathcal{T}$ and let $m \in \mathbb{M}_{\mathcal{F}}$. For every $v \in \mathcal{F}$, we associate an intermediate hash output O_v and an intermediate input I_v recursively as follows:*

1. *$v \in \mathcal{L}_{\mathcal{F}} \setminus \mathcal{L}_{\mathcal{T}}$, $|m(v)| = n$: $O_v = m(v)$ and there is no input,*
2. *$v \in \mathcal{L}_{\mathcal{F}} \cap \mathcal{L}_{\mathcal{T}}$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,*
3. *otherwise: $|m(v)| = n$ and we define*

$$I_v = \big(O_{v_{\mathsf{L}}} \oplus m(v),\ O_{v_{\mathsf{R}}} \oplus m(v)\big),\ \text{and } O_v = f_v(O_{v_{\mathsf{L}}} \oplus m(v),\ O_{v_{\mathsf{R}}} \oplus m(v)) \ \oplus\ O_{v_{\mathsf{R}}}.$$

O_ω is the final hash output corresponding to $\mathcal{F}$ where ω is the root of $\mathcal{F}$. We also call I_ω final input.

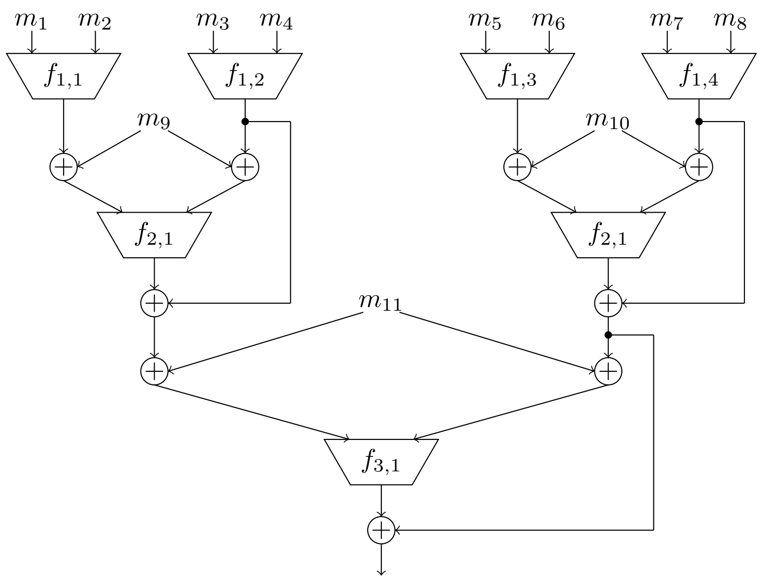

Figure 2 ABR of height 3.

Let us see what this means. If $\mathcal{F} = \mathcal{T}$, the above definition implies that for a leaf node v, the message at v, which itself is the input, is $2n$ bits long, and the output is just $f_v(m(v))$, where f_v is the 2-to-1 block compression function attached to it, and for an intermediate node, the message is n bits long, and the input and output are as defined above. If $\mathcal{F}$ is a proper sub-tree of $\mathcal{T}$, then there might exist vertices, which are leaves of $\mathcal{F}$, but not of $\mathcal{T}$. For such a vertex v, the message is n bits long, and the message itself is considered the output of the vertex. This vertex doesn't have any input.

The ABR Hash Function. The ABR hash is the hash output based on a perfect binary tree $\mathcal{T}$ of height l. In terms of Definition 6, the case $\mathcal{F} = \mathcal{T}$ corresponds to a ABR tree, and the final hash output is the ABR hash. Thus, ABR_l hash is a $(2^l + 2^{l-1} - 1)$-to-1 block hash function, $l > 1$. We refer to Figure 2 for a pictorial view of ABR with $l = 3$. For a trivial tree $\mathcal{F} = \{w\}$, with a message $m(\omega) \in \{0,1\}^{2n}$, $\mathcal{F}(m) = f_\omega(m)$.

We write $\mathtt{H}^\tau(m)$ and $\mathsf{in}^\tau(m)$ to denote the transcript based hash and the final input respectively, whenever defined for the message m for a tree $\mathcal{F}$. If $\mathtt{H}^\tau(m)$ is defined we call m τ-computable or simply computable message. We write $\perp$ to mean that it is undefined. Note that a tree is uniquely determined from the message. We write dom_v^τ and dom_{v-u}^τ to denote the set of all computable messages at v and for $\mathcal{T}_{v-u}$ respectively. Similarly, we write ran_v^τ and ran_{v-u}^τ to denote the set of all computable hashes at v and for $\mathcal{T}_{v-u}$ respectively. The size of the set ran_v^τ, called *load at* v, is denoted as $L_{\tau,v}$.

4 Local Opening Analysis of ABR Hash Function

In section 3, we have defined hash function based on a tree $\mathcal{F}$ for a message over the tree $\mathcal{F}$. In this section, we consider a variant of the message function and a hash function for the variant message. This is required to properly define the local opening of the ABR tree.

Message for a Full Binary Tree. Let $\mathcal{F}$ be a full binary tree and $L \subseteq \mathcal{L}_{\mathcal{F}}$. Let $\mathcal{M}_{\mathcal{F},L}$ be the set of all functions $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $v \in \mathcal{L}_{\mathcal{F}} \setminus L$, $m(v) \in \{0,1\}^{2n}$ and for all other vertices v, $m(v) \in \{0,1\}^n$. We call m a message (or a message function) for $\mathcal{F}$.

▶ **Definition 7** (Generalized Tree Hash, a variant). *Let $m \in \mathcal{M}_{L,\mathcal{F}}$ be a message function for $\mathcal{F}$. For every $v \in \mathcal{F}$, the intermediate hash output O_v is defined recursively as follows:*
1. $v \in L$, $|m(v)| = n$: $O_v = m(v)$,
2. $v \in \mathcal{L}_{\mathcal{F}} \setminus L$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,
3. $v \notin \mathcal{L}_{\mathcal{F}}$: *we define*

$$I_v = \big(h_1 \oplus m(v),\ h_2 \oplus m(v)\big) \text{ and } O_v = f_v(h_1 \oplus m(v),\ h_2 \oplus m(v)) \oplus h_2,$$

where $h_1 = O_{v_\mathsf{L}}$ *and* $h_2 = O_{v_\mathsf{R}}$.

The hash output corresponding to $\mathcal{F}$ is defined as $\mathcal{F}^f(m) := O_\omega$ where ω is the root of $\mathcal{F}$. We also call $I_\omega := \mathcal{F}^f_{\mathsf{in}}(m)$ final input. It is clear from the definition that for any node $v \notin L$, $\mathcal{F}^f_v(m|_v) = O_v$ and $\mathcal{F}^f_{v,\mathsf{in}}(m|_v) = I_v$.

Visualizing the tree is not difficult. As an example, when $\mathcal{F} = \mathsf{ABR}_3$, we have Figure 2, where L is a subset of the leaf nodes, say $(1,1)$ and $(1,2)$. We now define local opening of the Generalized Tree Hash.

▶ **Definition 8.** *Let m be a message for a perfect binary tree $\mathcal{T}$. For any full binary sub-tree $\mathcal{F}$ and a set $\mathcal{L}_{\mathcal{F}} \setminus \mathcal{L}_{\mathcal{T}} \subseteq L \subseteq \mathcal{L}_{\mathcal{F}}$, we define a message $m' := \mathsf{Open}^f_{\mathcal{F},L}(m) \in \mathcal{M}_{\mathcal{F},L}$ for $\mathcal{F}$ as follows.*
1. $v \in L$: $m'(v) = \mathcal{T}^f_v(m_v)$.
2. *Otherwise:* $m'(v) = m(v)$.

Now, we first analyze the local opening security of ABR_l proposed by [1] and then show that no non-trivial opening of ABR can achieve birthday bound security.

4.1 Local Opening Analysis of ABR Hash due to [1]

We describe the by-pass hash corresponding to the message block m_1 for ABR_l. It is based on the full sub-tree $\mathcal{F}$ consisting of nodes $\{(i,1) : i \in [l]\} \cup \{(i,2) : i \in [l-1]\}$ and $L = \{(1,2),(2,2),\ldots,(l-1,2)\}$. Refer to Figure 3. Note that the number of blocks in $\mathsf{Open}_{\mathcal{F},L}(m)$ is $2l$, and in the sub-tree $\mathcal{F}$ corresponding to $\mathsf{Open}_{\mathcal{F},L}(m)$, the number of calls to underlying compression function f is l. According to Stam's bound, there exists a collision attack with at most $2^{n/2l}$ queries. We give an attack that matches this bound.

Let $I(i,1) = (u_{i,1}, v_{i,1})$ be the input of $f_{i,1}$ and let $y_{i,1}$ be the output. Let $h_{i,2}$ be the message for node $(i,2)$. Let $h_{i,1} = y_{i,1} \oplus h_{i-1,2}$ for $i > 1$ and $h_{1,1} = y_{1,1}$. Then, $h_{i,1}$ is the output at node $(i,1)$. Also, let $m_{i,1}$ be the message associated with a non-leaf node $(i,1)$. We wish to find a collision at the output of node $(l,1)$, i.e. we need to find two messages m' and m'' for $\mathcal{F}$ such that $\mathcal{F}_{(l,1)}(m') = \mathcal{F}_{(l,1)}(m'')$. Given any message for $\mathcal{F}$, the output at node $(l,1)$ is given by $h_{l,1}$.

Note that $h_{1,1} = f_{1,1}(u_{1,1}, v_{1,1})$. After computing $h_{i-1,1}$, we proceed to compute $h_{i,1}$. We note that $h_{i-1,2}$ is a message block for $\mathcal{F}$. The input at node $(i,1)$, $I(i,1) = (h_{i-1,1} \oplus m_{i,1}, h_{i-1,2} \oplus m_{i,1}) = (u_{i,1}, v_{i,1})$ and the output at node $(i,1)$ is:

$$\begin{aligned} h_{i,1} = f_{i,1}(I(i,1)) \oplus h_{i-1,2} &= f_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,2} \\ &= f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1} \oplus h_{i-1,1} \\ &= g_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,1} \end{aligned}$$

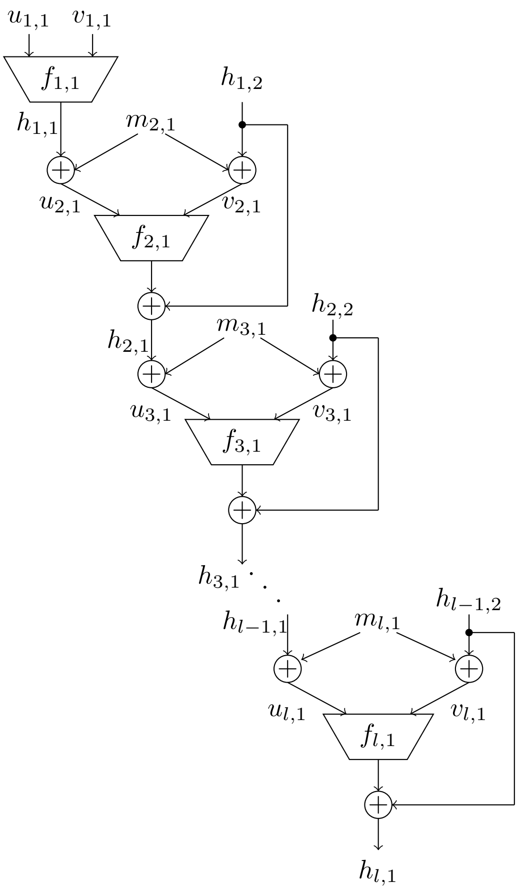

Figure 3 A specific local opening of ABR_3.

where $g_{i,1}(u_{i,1}, v_{i,1}) = f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1}$. By induction, the final hash computation is

$$h_{l,1} = g_{l,1}(u_{l,1}, v_{l,1}) \oplus g_{l-1,1}(u_{l-1,1}, v_{l-1,1}) \oplus \ldots \oplus g_{1,1}(u_{1,1}, v_{1,1}).$$

Since the functions $f_{i,1}$ are random and independent so are $g_{i,1}$'s. Thus $h_{l,1}$ is the XOR of l random functions. Thus, a collision is expected at node $(l, 1)$ with $2^{n/2l}$ queries. One can also apply a generalized birthday attack with complexity $2^{n/(1+\lceil \log 2l \rceil)}$.

Now, let us look at the target collision resistance of the above local opening of ABR_l. Target Collision Resistance describes the ability of an adversary to find a second pre-image for a fixed message. Target collision resistance has many practical applications. For example, if a client sends a file F to the server and then wants the server to send part of the file F_i along with a proof of correctness then, as long as the server does not control the choice of the file F, the server would need to find a targeted collision to break security and reveal an incorrect value F_i'.

Here, for a fixed message m, the final hash computation $h_{l,1}$ is fixed. Hence, for target collision resistance we wish the XOR of l random functions to collide with this value of $h_{l,1}$. This collision is expected with $2^{n/l}$ queries.

4.2 Decomposition of ABR Hash

Now we decompose ABR hash computation on $\mathcal{T}$ through a full binary proper sub-tree $\mathcal{F}$ sharing the same root and a set L.

▶ **Lemma 9** (decomposition lemma for any full binary tree). *For all full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ and a set of nodes $\mathcal{L}_\mathcal{F} \setminus \mathcal{L}_\mathcal{T} \subseteq L \subseteq \mathcal{L}_\mathcal{F}$, we have*

$$\mathcal{T}^f = \mathcal{F}^f \circ \mathsf{Open}^f_{\mathcal{F},L}.$$

Proof. Let m be a message for $\mathcal{T}$. $\mathcal{T}^f(m)$ represents the hash output based on the perfect binary tree $\mathcal{T}$. For any node v of $\mathcal{T}$, the restricted message over $\mathcal{T}_v$ is m_v. Hence, $\mathcal{T}^f(m)$ computes $\mathcal{T}^f_v(m_v)$ for all nodes $v \in \mathcal{T}$.

For any full binary sub-tree $\mathcal{F}$ of $\mathcal{T}$, $m' = \mathsf{Open}^f_{\mathcal{F},L}$ is defined as above. For any $v \in L$: $m'(v) = \mathcal{T}^f_v(m_v)$. We calculate the hash outputs for the restricted messages on these nodes first. Since for all other $v \in \mathcal{F}$, $m'(v) = m(v)$, and $\mathcal{F}$ is a sub-tree of $\mathcal{T}$, $\mathcal{F}^f(m')$ actually computes $\mathcal{T}^f_v(m_v)$ for all $v \in \mathcal{F} \setminus \mathcal{L}_\mathcal{F}$. Thus, $\mathcal{F}^f \circ \mathsf{Open}^f_\mathcal{F}(m)$ also computes $\mathcal{T}^f_v(m_v)$ for all nodes $v \in \mathcal{T}$ and produces the same output $\mathcal{T}^f(m)$. ◀

If $\mathcal{F} = \mathcal{T}$ and $L = \emptyset$ then $\mathsf{Open}^f_{\mathcal{F},L}(m) = m$. For any other proper local opening we cannot ensure birthday bound security. We prove the following theorem:

▶ **Theorem 10.** *No non-trivial opening of* ABR *can achieve birthday bound security.*

Proof. Stam's bound states that there exists a collision attack with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function making r calls to λ-to-1 block compression functions. We have $\lambda = 2$. If we want to achieve $2^{n/2}$ collision security, $t \leq 1.5r + 0.5$. In other words, if $t > 1.5r + 0.5$, then we have a collision attack with query complexity $2^{\frac{n}{2}(1-\delta/r)}$, $\delta := t - 1.5r - 0.5$.

For ABR of height l, we have $t = 2^l + 2^{l-1} - 1$ and $r = 2^l - 1$. This satisfies $t = 1.5r + 0.5$, and it is optimal. We show that for any non-trivial opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR, $\mathcal{F}$ satisfies $t > 1.5r + 0.5$. Let us consider the simplest non-trivial opening, corresponding to $L = \{(1,1)\}$. Then, for $m = (m_1, m_2, m')$, where m_1, m_2 are the first two message blocks and m' is the remaining part, $\mathsf{Open}_{\mathcal{F},L}(m) = (f_{1,1}(m_1, m_2), m')$. Then, $t = 2^l + 2^{l-1} - 2$, and $r = 2^l - 2$ ($f_{1,1}$ is not called). This satisfies $t > 1.5r + 0.5$. If $\mathsf{Open}_{\mathcal{F},L}$ consists of only one sub-tree computation of height h, then for $\mathcal{F}$, we have $t = (2^l + 2^{l-1} - 1) - (2^h + 2^{h-1} - 1) + 1$ and $r = 2^l - 2^h$, which satisfies $t > 1.5r + 0.5$.

A general opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR may consist of more than one complete sub-tree computation. Let the number of complete sub-tree computations in $\mathsf{Open}_{\mathcal{F},L}$ be k, and for each $1 \leq i \leq k$, let h_i be the height of the i-th sub-tree. Then, for $\mathcal{F}$, we have

$$t = (2^l + 2^{l-1} - 1) - \sum_{i=1}^{k}(2^{h_i} + 2^{h_i-1} - 1) + k, \qquad r = (2^l - 1) - \sum_{i=1}^{k}(2^{h_i} - 1).$$

It can be easily seen that $t > 1.5r + 0.5$. Thus, no non-trivial opening of ABR can achieve birthday bound security. ◀

5 Collision Analysis of ABR hash

In this section, we first define certain items which will be required to analyze the collision.

▶ **Definition 11** (input multi-collision). *For any $x \in \{0,1\}^n$, let $\mathrm{MC}_v^\tau(x)$, called input multi-collision set at v (with x as input multi-collision value), denote the set of all messages m at v with $\mathsf{in}^\tau(m) = x$. also, let*

$$\mathrm{mc}_v^\tau(x) = |\mathrm{MC}_v^\tau(x)|, \quad \mathrm{mc}_v^\tau = \max_{x \in \{0,1\}^n} \mathrm{mc}_v^\tau(x).$$

When v is the root node, we skip the notation v.

We define the newly generated messages and the hashes at a node v due to addition of the query-response (x, y) to the transcript τ as

$$\mathsf{New}_v^\tau(x,y) := \mathrm{dom}_v^{\tau\cup(x,y)} \setminus \mathrm{dom}_v^\tau, \quad \mathsf{NewH}_v^\tau(x,y) := \mathrm{ran}_v^{\tau\cup(x,y)} \setminus \mathrm{ran}_v^\tau.$$

Clearly, $\mathsf{NewH}_v^\tau(x,y) = \mathtt{H}^{\tau\cup(x,y)}(\mathsf{New}_v^\tau(x,y))$ (image set of $\mathtt{H}^{\tau\cup(x,y)}$ for the domain $\mathsf{New}_v^\tau(x,y)$). Note that x need not be queried at v. However, to have a new computable message, x should be queried at some node, say u, in $\mathcal{T}_v$. Analyzing the behavior of the set $\mathsf{New}_v^\tau(x,y)$ (or its size) is easy when $u = v$ or when u is one of the children of v. However, it becomes more complex when u is far away from v.

- Case $u = v$: $\mathsf{New}_v^\tau(x,y) = \mathrm{MC}_v^\tau(x)$ (and does not depend on y) and we call these messages freshly generated **immediate** messages.
- Case $u \in \mathcal{T}_v \setminus v$: The newly generated messages at v is

$$\mathsf{New}_v^\tau(x,y) = \{m|_v : \mathsf{in}^\tau(m|_u) = x, m|_{v-u\to h} \in \mathrm{dom}_{v-u}^\tau, h = y \oplus \mathtt{H}^\tau(m|_{u\mathsf{R}})\}.$$

So, we have $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) = \dfrac{\mathrm{mc}_u^\tau(x) \times |\mathrm{dom}_{v-u}^\tau|}{2^n}$.

Now we discuss how the size of the computable message space $|\mathrm{dom}_{v-u}^\tau|$ can be written when u is one of the children or grandchildren of v.

▶ **Example 12.** Suppose $u = v_{\mathsf{R}}$. In this case,

$$\begin{aligned}\mathsf{New}_v^\tau(x,y) =& \{m|_v : \mathsf{in}^\tau(m_{\mathsf{R}}) = x, y = \mathtt{H}^\tau(m_{\mathsf{RR}}) \oplus \mathtt{H}^\tau(m_{\mathsf{L}}) \oplus x_1 \oplus x_2, \\ & \mid (v,(x_1,x_2)) \in \mathrm{dom}(\tau), m(v) = x_1 \oplus \mathtt{H}^\tau(m_{\mathsf{L}})\}.\end{aligned}$$

So, $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) \le \dfrac{\mathrm{mc}_{v_{\mathsf{R}}}^\tau(x) \times |\mathrm{ran}_{v_{\mathsf{L}}}^\tau| \times |\tau_v|}{2^n}$, where τ_v denotes the set of elements in the transcript of the form $((v,x),y)$.

▶ **Example 13.** In the previous case, we could write the expectation of number of newly generated messages in terms of input multi-collision and range size of tree hash. Now, we consider $u = v_{\mathsf{RR}}$, i.e. u is a grandchild of v. Refer to Figure 4. Let $h = y \oplus \mathtt{H}^\tau(m_{\mathsf{RRR}})$. First, let us look at $|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau|$.

$$\begin{aligned}|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau| =& \{m|_v : H_1 \oplus h = x_1' \oplus x_2', H_2 \oplus y' \oplus h = x_1'' \oplus x_2'', \\ & \mid H_1 = \mathtt{H}^\tau(m_{\mathsf{RL}}), H_2 = \mathtt{H}^\tau(m_{\mathsf{L}}), (v_{\mathsf{R}},(x_1',x_2'),y'), (v,(x_1'',x_2''),*) \in \tau\}.\end{aligned}$$

Note that this implies $H_1 \oplus H_2 \oplus \bar{y}' = x_1'' \oplus x_2''$, where $\bar{y}' = x_1' \oplus x_2' \oplus y'$. Thus,

$$|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau| = \mathrm{mc}(\mathrm{ran}_{v_{\mathsf{L}}}^\tau \oplus \mathrm{ran}_{v_{\mathsf{RL}}}^\tau \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|,$$

where $\bar{f}_{v_{\mathsf{R}}}(u_1,u_2) = u_1 \oplus u_2 \oplus f_{v_{\mathsf{R}}}(u_1,u_2)$. Hence,

$$\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) \le \frac{\mathrm{mc}_{v_{\mathsf{RR}}}^\tau(x) \times \mathrm{mc}(\mathrm{ran}_{v_{\mathsf{L}}}^\tau \oplus \mathrm{ran}_{v_{\mathsf{RL}}}^\tau \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|}{2^n}.$$

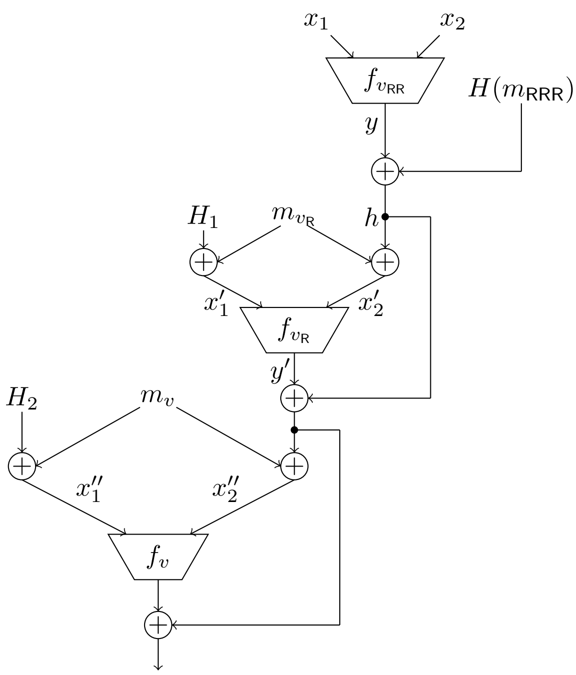

Figure 4 The graph of $\mathcal{T}_{v-u}$ when u is the rightmost grandchild of v.

Adversary and Its Queries. Let $\mathcal{L}_v$ denote the lists of all responses of f_v, for all leaf node v. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to f_v's have no role in the collision event. However, this is not true for all intermediate nodes (the non-leaf nodes) and so adaptivity of intermediate nodes must be considered. We assume that an adversary makes exactly q queries to each node. Let $q' := qr$ denote the total number of queries where $r = |V^*|$ and V^* is the set of non-leaf or intermediate nodes. Let Q_v denote the set of query numbers for the node v, $v \in V^*$. So for all non-leaf node v, $|Q_v| = q$. Let (x_i, y_i) denote the i-th query-response pair made to the node v_i. So given transcript τ^{i-1} (transcript after $(i-1)$ queries), the distribution of y is uniform over $\{0,1\}^n$. For notational simplicity, we use simply i in a superscript instead of τ^i (the transcript after i-th query) in all above notations defined so far. For example, $\mathtt{H}^i(m)$ denotes the transcript based hash of m where the transcript is τ^i. We write New_v^i instead of $\mathsf{New}_v^{\tau^{i-1}}(x_i, y_i)$, which represents the set of all newly generated computable messages at node v immediately after obtaining i-th query-response. We also ignore the superscript τ^i completely when we all the queries have been made, i.e. $i = q'$. For example, we write $\mathrm{mc}_v(x)$ instead of $\mathrm{mc}_v^{\tau}(x)$, when τ is the final transcript, obtained at the end of all the queries.

- For any computable message m at v, we write $\mathrm{Fin}(m) := i$ to encode the final query index after which m is computable.

- For all m for which $m_{\mathsf{L}}, m_{\mathsf{R}}$ are τ-computable, we define $\mathrm{Fin}^*(m) = i$ such that $\max\{\mathrm{Fin}(m_{\mathsf{L}}), \mathrm{Fin}(m_{\mathsf{R}})\} = i$, (i.e. immediately after i-th query the final-input for the message m is computable).

5.1 Steps of Collision Analysis

Proper Internal Collision. We say that a **proper internal collision** happens at $v = (j, b)$ for a transcript τ if for some distinct messages m, m' at v, (i) $\mathtt{H}^\tau(m) = \mathtt{H}^\tau(m')$, (ii) $\mathsf{in}^\tau(m) \neq \mathsf{in}^\tau(m')$, and (iii) no collision happens for $\mathtt{H}^\tau_u$ for all $u \in V(\mathcal{T}_v)$, $u \neq v$. By using standard reduction, a collision of ABR must have proper internal collision at some node. So it is sufficient to bound the probability of a proper internal collision at the root node of ABR as $\mathtt{H}_v$ is identical to ABR_s where s denotes the level of the node v. We write $\mathsf{coll} := \mathsf{coll}_l$ to denote the proper internal collision at the root node of $\mathcal{T}$ of height l. The probability of collision of ABR_l can be then bounded as $\sum_{i \le l} 2^{l-i} \Pr(\mathsf{coll}_i)$.

Now, there are two types of collision which can happen for any proper collision at the root. Let us consider the i-th query. This query itself can generate two new computable messages for which the collision occurs. This is the first type of collision. Also, the hash output of one among the new computable messages generated by the i-th query can match with one of the hash outputs generated by the previous queries. We formalize them here:

▶ **Definition 14** (types of collision)**.**

- *We call a collision pair (M, M') **twin** at the i-th query, $i \in [q']$ if $M, M' \in \mathsf{New}^i$. In this case $\mathsf{in}^i_{v_i}(M) = \mathsf{in}^i_{v_i}(M') = x_i$, where v_i is the node where the i-th query is made.*
- *The collision pair is called **non-twin** at the i-th query if exactly one of M and M' is a member of New^i, and the other message is τ^{i-1}-computable.*

We write coll^i to denote that the proper internal collision happens at the i-th query. Moreover, if it is a twin-collision (or non-twin collision) we denote the event as $\mathsf{coll}^{i,\mathrm{tw}}$ (or $\mathsf{coll}^{i,\mathrm{ntw}}$ respectively). Thus,

$$\mathsf{coll} = \bigcup_{i \in [q']} \left(\mathsf{coll}^{i,\mathrm{ntw}} \cup \mathsf{coll}^{i,\mathrm{tw}}\right).$$

It is easy to see that twin-collision at the root node is not possible as a collision at the right child of the root node is necessary. In notation, $\mathsf{coll}^{i,\mathrm{tw}} = \emptyset$, whenever $v_i = \omega$.

5.1.1 Non-Twin Collision Analysis

For any non-root, non-leaf node v, we consider cross-collision between $\mathtt{H}_{-v}$ and $\mathtt{H}_\omega$. Let CC^i_v denote the set of all pairs (m, m') such that (i) m is a complete message, m' is a message for $\mathcal{T}_{-v}$ and (ii) $\mathtt{H}^i(m) = \mathtt{H}^i_{-v}(m')$. Now, a *non-twin collision* can happen at the i-th query (to the node v_i) if freshly generated hash of a message at v_i matches with the v_i-th message block of m' for a cross-collision pair (m, m') of CC^{i-1}_v. Thus,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \le \frac{\mathrm{mc}^{i-1}_v(x_i) \times |\mathsf{CC}^{i-1}_v|}{2^n}. \tag{2}$$

Now, if $v = \omega$ then the freshly generated hash at the root node is a hash. So, we have,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \le \frac{\mathrm{mc}^{i-1}_\omega(x_i) \times L}{2^n}. \tag{3}$$

5.1.2 Twin Collision Analysis

For any non-root, non-leaf node v and $\delta \in \{0,1\}^n \setminus \{0^n\}$, let $\mathsf{C}_{\delta,v}$, called δ-collision, denote the set of all pairs (m, m') such that $\mathtt{H}^\tau_{-v}(m) = \mathtt{H}^\tau_{-v}(m')$ and $m(v) \oplus m'(v) = \delta$. We have seen that no twin collision possible at the root node. We define a set

$$\Delta^i = \{\mathtt{H}^{i-1}(m_\mathsf{R}) \oplus \mathtt{H}^{i-1}(m'_\mathsf{R}) : m, m' \in \mathrm{MC}^{i-1}_{v_i}(x_i)\}.$$

Now,

$$\Pr(\mathsf{coll}^{i,\mathrm{tw}}) \leq \frac{\sum_{\delta \in \Delta} \mathrm{mc}_v^{i-1}(x_i) \times |\mathsf{C}_{\delta,v}^{i-1}|}{2^n}. \tag{4}$$

Note that the size of Δ can be at most $(\mathrm{mc}_v^{i-1}(x_i))^2$.

Thus, we have seen a collision analysis requires to bound the following random variables.

1. $\mathrm{mc}_v^{i-1}(x_i)$ for all i (and so for all nodes v),
2. L: load of the hash,
3. $|\mathrm{dom}_{-v}^{i-1}|$: load for $\mathcal{T}_{-v}$ which is required to bound the load L,
4. $|\mathsf{C}_{\delta,v}^{i-1}|$: size of δ-collision, and
5. $|\mathsf{CC}_v^{i-1}|$: size of cross-collision.

In the following subsection, we present the collision analysis of ABR_2 in which we only need the input multi-collision and load (which is also bounded in terms of input multi-collision). We also present a collision analysis of ABR_3 for which the above terms are present.

5.2 Collision Analysis of ABR_2 by ABR

As discussed above, we can assume that all queries to the compression functions at the leaf node have been made beforehand and let q denote the number of queries to each oracle. Let $\mathcal{L}_1, \mathcal{L}_2$ be two lists of outputs of the leaf node functions and let $\omega := (2,1)$ denote the root (the only non-leaf node for $\mathcal{T}$ of height 2). Note that the proper collision at height 1 is the same as the collision of the lists $\mathcal{L}_1, \mathcal{L}_2$. The proper collision at a leaf node can happen with probability at most $q^2/2^n$.

So, we now consider collision at the root $(2,1)$. For this, we now define a bad event mc_ω that $\mathrm{mc}_\omega^q > n$. Equivalently, the event can be expressed as $\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$. Note that we do not have any non-leaf node other than root node. So, the load for hash values L can be upper bounded as nq, given that mc_ω^q does not hold. Moreover, cross-collision and δ-collision is also not possible as we do not have any non-leaf, non-root node. Now, it is well known that

$$\Pr(\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n) \leq \frac{q^2}{2^n}$$

(see [1] for details). Thus, the collision probability is bounded by $\frac{(n^2+2)q^2}{2^n}$.

5.3 Collision Analysis of $\mathsf{ABR}_h, h \geq 3$ by [1]

The proof of [1] is divided into two main parts: (i) bounding the load and (ii) bounding proper collision probability in terms of the load. ABR fix a parameter ρ (which is chosen to be $n+1$, however, the exact value is not relevant to our discussion). Let $L_{i,v} = \sum_{j \leq i, j \in Q_v} |\mathsf{NewH}_v^i|$ represent the total number of generated hash values at v after all i queries. If there is no collision (which is true while we consider proper internal collision), $L_{i,v}$ is same as the size of the set $|\mathrm{dom}_v^i|$. To bound load, ABR considered the following bad events (in our notations):

1. $\mathsf{bad}_{1,v}$: $\mathrm{mc}_v^{q'} > \rho$ at v. Let $\mathsf{bad}_1 := \cup_v \mathsf{bad}_{1,v}$.
2. $\mathsf{bad}_{2,v}$: $L_{q',v} \geq \rho q$. Let $\mathsf{bad}_2 := \cup_v \mathsf{bad}_{2,v}$.

Given $\mathsf{bad}_1, \mathsf{bad}_2$ do not hold, clearly $L \leq 2\rho q$.

5.3.1 Step-1: Bounding $\Pr(\mathsf{bad}_1)$

Let $\mathsf{bad}_{1,\leq i} = \cup_{(j,b):j\leq i}\mathsf{bad}_{1,v}$. So it is sufficient to bound $\Pr(\mathsf{bad}_{1,(j,b)} \wedge \neg\mathsf{bad}_{1,<j})$. Let us fix a query x at $v = (j,b)$. Now, ABR implicitly claimed the following:

▷ Claim ([1]). If $\mathrm{MC}^{q'}_{(j,b)}(x) \supseteq \{m_1,\ldots,m_\rho\}$ then $\mathsf{in}_{(j-1,2b)}(m_{i,\mathsf{R}})$'s are distinct.

We note that this claim is not correct. As there can be ρ multi-collision at node $(j-1,2b)$, each query can potentially give at most ρ multi-collision at node (j,b). Hence we can have ρ^2 multi-collision at node (j,b). Thus, a corrected version of the above claim requires to revise the parameter ρ depending on the level. So, we may redefine $\mathsf{bad}_{1,(j,b)}$: $\mathrm{mc}_v > \rho^j$ which could solve the issue. This is a fixable minor issue (but will have an impact on the claimed bound).

Now to continue with the bound, let us assume that $\mathrm{MC}^{q'}_v(x) \supseteq \{m_1,\ldots,m_\rho\}$ such that $\mathsf{in}(m_{i,\mathsf{R}})$'s are distinct and $x = (a,b)$. So we can choose ρ query indices out of q queries to $v_2 := v_{\mathsf{R}}$ in $\binom{q}{\rho}$ ways. For any such choices of ρ tuple $(i_1,i_2,\ldots,i_\rho)$ (all queried to v_2), we have

$$\Pr(f(x_{i_1}) \oplus \mathtt{H}(m_{1,\mathsf{RR}}) = b,\ldots,f(x_{i_\rho}) \oplus \mathtt{H}(m_{\rho,\mathsf{RR}}) = b) = \frac{2\rho q}{2^{n\rho}}$$

as there ρq many choices of $\mathtt{H}(m_{i,\mathsf{RR}})$ values (as we assume the load at v_{RR} is less than $2\rho q$). However, the above is true when we consider the cases where $\mathrm{Fin}^*(m_i) = j_i$ where $v_{j_i} = v_2$ for all i. The most important case in which the input multi-collision is contributed due to the final queries which are not on right child is not considered in the proof by [1].

5.3.2 Step-2: Bounding $\Pr(\mathsf{bad}_2)$

Let $\mathsf{bad}_{2,\leq i} = \cup_{(j,b):j\leq i}\mathsf{bad}_{2,v}$. So it is sufficient to bound

$$\Pr(\mathsf{bad}_{2,(j,b)} \wedge \neg\mathsf{bad}_{1,<j} \wedge \neg\mathsf{bad}_1 \wedge \neg\mathsf{coll}).$$

The main idea to bound the above probability is to bound the expected number of newly generated hash at $v = (j,b)$ over all queries. Then the bad event probability can be bounded by applying Markov's inequality. We have already seen that

$$\mathbb{E}_y(|\mathsf{New}^i_v| \mid \tau^{i-1}) = \frac{\mathrm{mc}^{\tau}_{v_i}(x_i) \times |\mathrm{dom}^i_{v-v_i}|}{2^n}.$$

Moreover, we have shown that bounding $|\mathrm{dom}^i_{v-v_i}|$ becomes more complex when v_i is neither v nor a child of v (see Example 13). [1] tried to argue in a different way. ABR showed a bound expectation of load due to all queries of its children (see Example 12). Then, they continued this argument for two levels up (i.e. for the queries on grandchildren as we consider in Example 13). However, they did not analyze this case properly. In particular, they did not consider to bound the $\mathrm{mc}(\mathrm{ran}^{\tau}_{v_{\mathsf{L}}} \oplus \mathrm{ran}^{\tau}_{v_{\mathsf{RL}}} \oplus \bar{f}_{v_{\mathsf{R}}})$. Finally, they claimed the general case by using induction which is clearly unverifiable.

5.3.3 Step-3: Proving Collision in terms of Load

ABR stated that as analyzed for ABR_2, given (i) no collision for all primitive, (ii) $\neg\mathsf{bad}_{1,\leq l}$ and (iii) $\neg\mathsf{bad}_{2,\leq l}$, the proper internal collision probability at the root node is $\mathbb{E}(L^2)/2^n$ where L is the total number computable hash values.

There is a fundamental gap in the high level of the proof. As ABR did not explain anything supporting his claim, we show that this statement is not true in general. In particular, we show (in the next subsection) a *hash mode based on* 2*-to-*1 *compression function whose load is at most* q^2 *(for any* q*-query adversary), however, a collision can be found in* $O(n)$ *queries.* So the above claim cannot be made in general.

5.3.4 Missing Step: Twin-Collision Analysis

We find that the twin-collision analysis of the ABR hash is missed completely. The bound for δ-collision is not obvious and it requires bounding the probability of some more bad events. In the following section, we have analyzed ABR_3 in which the twin-collision analysis requires a bad event dealing with the multi-collision of xor of random oracle compression function outputs for two distinct inputs. We do not know any method to bound the number of cross-collision pairs for a general height tree.

5.4 Relationship between Load and Collision Probability

A hash function with a high load is unlikely to be collision-resistant. For example, $\mathsf{xor}(x_1, \ldots, x_r) = f_1(x_1) \oplus \cdots \oplus f_r(x_r)$ has load 2^r after 2 queries to each oracle f_i. It is easy to see that the hash function xor is not collision-resistant. Let $r = n$. Then, after making two queries to each function, we have sufficiently many computable messages. It is then very easy to find computable collision pairs by solving a linear system of equations. In general, if the load becomes the order of $2^{n/2}$ then one may expect a collision. However, the converse need not be true. In other words, we have a hash function where load can not be high, but still, a collision pair can be generated efficiently.

Example of Collision Insecure Hash Functions with Low Load

Let MD^f be the MD hash which takes n blocks and initial value is also replaced by one message block (so exactly $n-1$ calls of f is required). We define $MD_n^f(M) = MD^{f_1}(M) \| \cdots \| MD^{f_n}(M)$ which is n^2-to-n^2 hash function. Now we define a hash function $\mathtt{H}(M_1, M_2)$ for $M_1, M_2 \in \{0,1\}^{n^2}$:

1. Let $(C_1, C_2) = (MD_n^f(M_1) \oplus M_2, MD_n^g(C_1) \oplus M_1)$ (two round LR construction which is invertible).
2. Let $h_1, \ldots, h_n$ be $2n$-to-n functions. The final hash output is defined as $h_1(x_1) \oplus \cdots \oplus h_n(x_n)$ where $C_1 \| C_2 = x_1 \| \cdots \| x_n$, $x_i \in \{0,1\}^{2n}$.

Note that we cannot compute (C_1, C_2) for more than q^2 messages assuming there is no collisions in f and g functions. So, $L(q) \leq q^2$ for any q-query adversary.

A Collision Attack. Now, we construct a collision finding algorithm for the above hash. It first finds collision pair for xor function $h_1 \oplus \cdots \oplus h_n$ (can be achieved easily by making $2n$ queries altogether). Let (C, C') be a collision pair. We can easily invert C and C' to obtain M and M' respectively. Clearly, (M, M') is a computable collision pair.

6 Analysis of ABR of height 3

In this section, we show that the ABR_3 construction achieves birthday security. In particular, we prove the following theorem:

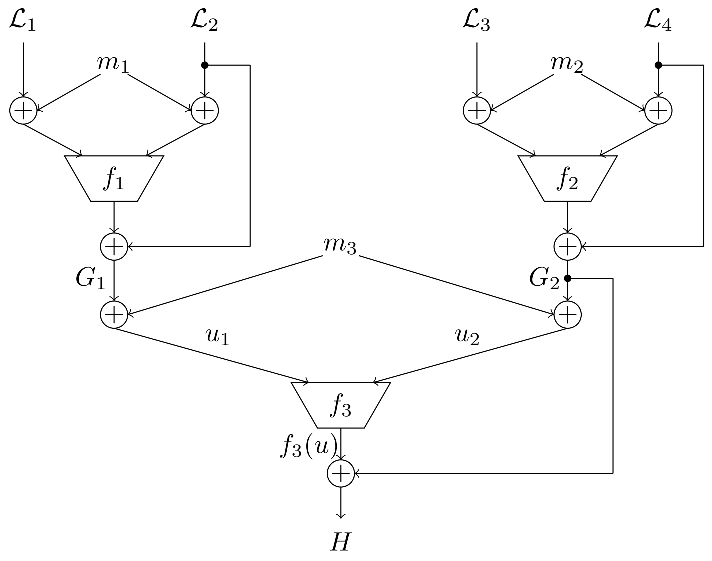

Figure 5 ABR_3 according to our new notation when the query $u = (u_1||u_2)$ is made to f_3.

▶ **Theorem 15** (collision theorem for ABR_3). *For any adversary $\mathcal{A}$ making at most q queries to each compression function modeled to be random oracle, we have*

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathtt{H}^f}(\mathcal{A}) \leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q + 2n^2q^2 + 13q^2}{2^n}. \tag{5}$$

Let $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ be the four lists of size q each corresponding to the outputs of $f_{1,1}, f_{1,2}, f_{1,3}, f_{1,4}$ respectively. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to $f_{1,i}$'s are independent from the rest of the transcripts. Also, for ease of notation, from now on we denote $f_{2,1}$ by f_1, $f_{2,2}$ by f_2 and $f_{3,1}$ by f_3. If the input to any of the functions is $u = (u_1, u_2)$, we define $u^{\oplus} = u_1 \oplus u_2$. Also, if $f_3(u) = v$, then we define $\bar{f}_3(u) = u^{\oplus} \oplus v$. As f_3 is a random oracle, the output distributions of $\bar{f}_3$ are uniform and independent. Let Q_j be the set of queries to f_j. We assume $|Q_j| = q$ for $j = 1, 2, 3$. Also, let Q_j^i denote the set of queries to Q_j up to the i-th query (including the i-th one). Let $\mathcal{G}_1 = O_{(2,1)}$ denote the set of intermediate hash outputs at node $(2, 1)$ and $\mathcal{G}_2 = O_{(2,2)}$. Let $\mathcal{H}$ denote the set of final hash outputs of ABR_3. Refer to Figure 5 for a pictorial representation. We follow the general approach as described before. We have already shown the collision bound for ABR_2 and so it is sufficient to bound proper collision at the root for ABR_3.

As we have seen above, the collision analysis requires us to bound some random variables. We first define some bad events to bound these random variables.

▶ **Definition 16** (list collision). *The first bad event we consider is:*

- $B0$*: There exists a collision in at least one of the lists* $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_4$, $\{f_1(u) : u \in Q_1\}$, $\{f_2(u) : u \in Q_2\}$, $\{f_3(u) : u \in Q_3\}$.

Since f is modeled as a random function, the collision probability in any of the lists is at most $q^2/2^n$. Hence, $\Pr(B0) \leq 7q^2/2^n$.

▶ **Definition 17** (bad event on input multi-collision). *We define the following bad events:*

- $B1$*:* $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$*, or* $\mathrm{mc}(\mathcal{L}_3 \oplus \mathcal{L}_4) > n$*,*
- $B2 : \mathrm{mc}(\mathcal{G}_1 \oplus \mathcal{G}_2) > n^2$.

We now state some simple observations related to input multi-collision:

1. Given that $B1$ does not hold, $\text{mc}_{(2,1)}, \text{mc}_{(2,2)} \leq n$ and so $|\mathcal{G}_1|, |\mathcal{G}_2| \leq nq$.
2. Given that $B2$ does not hold, $\text{mc}_{(3,1)} \leq n^2$ and so $L_{(3,1)} \leq n^2 q$.
3. Note, $|\text{dom}(\mathcal{T}_{-(2,1)})|, |\text{dom}(\mathcal{T}_{-(2,2)})| \leq nq^2$. So, $\mathbb{E}(L_{(2,1)}), \mathbb{E}(L_{(2,2)}) \leq n^2 q^3/2^n$. By Markov's inequality, $\Pr(L > 3n^2 q) \leq 2q^2/2^n$ ($3n^2q$ because we include $L_{(3,1)}$ as well).
4. By using a similar argument as we applied for multi-collision, we have $\Pr(B1) \leq 2q^2/2^n$.
5. Now, given that $B1$ does not hold and $B2$ holds, there must exist at least n distinct inputs to f_2 leading to n^2 input multi-collision. So, we can similarly prove $\Pr(B2) \leq q^2/2^n$.

We say that bad_{mc} holds if either $B1$ or $B2$ happens, or $L > 3n^2 q$. Then, from above, $\Pr(\mathsf{bad}_{mc}) \leq 3q^2/2^n$. We now define bad events which would be used to bound cross-collision.

▶ **Definition 18** (bad event on cross-collision)**.** *We define the following bad events:*

- *$B3$: $|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3, H \in \mathcal{H}\}| > 3n^4 q$.*
- *$B4$: $|\{(G_1, \bar{f}_3(u), H) : G_1 \oplus \bar{f}_3(u) \oplus H = 0; G_1 \in \mathcal{G}_1, u \in Q_3, H \in \mathcal{H}\}| > 3n^4 q$.*

We say that bad_{cc} holds if any one of the above happens.

If the i-th query is made at f_2, an intermediate hash output G_2 generated at this level due to this query can match with a query u already done to f_3 to generate a final hash output H which was already previously generated by the first $i-1$ queries. The event $B3$ implies that the number of such triplets $(G_2, f_3(u), H)$ is more than $3n^4 q$. $B4$ has a similar implication when we consider $\mathcal{G}_1$ instead of $\mathcal{G}_2$.

▶ **Lemma 19.** $\Pr(\mathsf{bad}_{cc} \wedge \neg\mathsf{bad}_{mc}) \leq 2q^2/2^n$.

Proof. $\Pr(\text{mc}(\mathcal{G}_2 \oplus \text{ran}(f_3)) > n^2) \leq q^2/2^n$. The proof is similar to that of event $B2$. Hence, for a fixed $H \in \mathcal{H}$, we have

$$\Pr[|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3\}| > n^2] \leq q^2/2^n.$$

Now, there are $3n^2 q$ choices for H. Therefore, $\Pr(B3 \wedge \neg\mathsf{bad}_{mc}) \leq q^2/2^n$. A similar argument works for $B4$. Hence,

$$\Pr(\mathsf{bad}_{cc} \wedge \neg\mathsf{bad}_{mc}) \leq 2q^2/2^n.$$ ◀

Given that bad_{cc} does not hold, $|\mathsf{CC}_{(2,1)}| \leq 3n^4 q$ (or $|\mathsf{CC}_{(2,2)}| \leq 3n^4 q$ respectively). We finally define bad events which would be used to bound δ-collision pairs.

▶ **Definition 20** (bad event on δ-collision)**.** *We define the following bad event:*

- *$B5$: $\mathsf{mc}(\bar{f}_3(u) \oplus \bar{f}_3(u')) > n$.*

We say that bad_δ holds if the above happens.

▶ **Lemma 21.** $\Pr(\mathsf{bad}_\delta) \leq \frac{q^2}{2^n}$.

Proof. Since $f_3(u)$ is random, $\bar{f}_3(u) = f_3(u) \oplus u^{\oplus}$ is also random. Therefore, bounding $B5$ is similar to bounding $B1$. ◀

Given that bad_δ does not hold, $|\mathsf{C}_\delta| \leq n$. Let $\mathsf{bad} = B0 \cup \mathsf{bad}_{mc} \cup \mathsf{bad}_{cc} \cup \mathsf{bad}_\delta$. Then, $\Pr(\mathsf{bad}) \leq \frac{13q^2}{2^n}$.

Collision Analysis

We assume that bad does not hold. Since $\mathsf{coll} = \bigcup_{i\in[q],v\in V\setminus\mathcal{L}} (\mathsf{coll}_v^{i,\mathrm{ntw}} \cup \mathsf{coll}_v^{i,\mathrm{tw}})$, we need to bound $\mathsf{coll}_v^{i,\mathrm{ntw}}$ and $\mathsf{coll}_v^{i,\mathrm{tw}}$ for $v = (2,1),(2,2),(3,1)$. In the following lemmas, we bound them, assuming bad does not occur. We already know that $\mathsf{coll}_{(3,1)}^{i,\mathrm{tw}}$ does not occur.

▶ **Lemma 22.** $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \frac{3n^4q}{2^n}$.

Proof. As seen above in equation 3, $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}) \leq \dfrac{\mathrm{mc}_{(3,1)}^{i-1}(x_i) \times L}{2^n}$.

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(3,1)} \leq n^2$ and $L \leq 3n^2q$. Hence, $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \dfrac{3n^4q}{2^n}$. ◀

▶ **Lemma 23.** $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$.

Proof. As seen above in equation 3, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}) \leq \dfrac{\mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,1)}^{i-1}|}{2^n}$.

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,1)} \leq n$ and $|\mathsf{CC}_{(2,1)}| \leq 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \dfrac{3n^5q}{2^n}$. ◀

▶ **Lemma 24.** $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$.

Proof. This proof is similar to that of the previous lemma.
$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}) \leq \dfrac{\mathrm{mc}_{(2,2)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,2)}^{i-1}|}{2^n}$.

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,2)} \leq n$ and $|\mathsf{CC}_{(2,2)}| \leq 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \leq \dfrac{3n^5q}{2^n}$. ◀

▶ **Lemma 25.** $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \leq \frac{n^4}{2^n}$.

Proof. As seen above in equation 4, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}) \leq \dfrac{\sum_{\delta\in\Delta} \mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{C}_{\delta,(2,1)}^{i-1}|}{2^n}$.
Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,1)} \leq n$, $|\Delta| \leq (\mathrm{mc}_{(2,1)}^{i-1}(x_i))^2 \leq n^2$ and $|\mathsf{C}_{\delta,(2,1)}| \leq n$. Hence,
$\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \leq \dfrac{n^4}{2^n}$. ◀

▶ **Lemma 26.** $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \leq \frac{n^4}{2^n}$.

Proof. This proof is similar to that of the previous lemma.
$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}) \leq \dfrac{\sum_{\delta\in\Delta} \mathrm{mc}_{(2,2)}^{i-1}(x_i) \times |\mathsf{C}_{\delta,(2,2)}^{i-1}|}{2^n}$.
Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,2)} \leq n$, $|\Delta| \leq (\mathrm{mc}_{(2,2)}^{i-1}(x_i))^2 \leq n^2$ and $|\mathsf{C}_{\delta,(2,2)}| \leq n$. Hence,
$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \leq \dfrac{n^4}{2^n}$. ◀

From the above lemmas, we have

$$\Pr(\mathsf{coll}|\neg\mathsf{bad}) \leq \sum_{i\in[q],v\in V\setminus\mathcal{L}} \Pr(\mathsf{coll}_v^{i,\mathrm{ntw}}|\neg\mathsf{bad}) + \Pr(\mathsf{coll}_v^{i,\mathrm{tw}}|\neg\mathsf{bad}) \leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q}{2^n}.$$

Therefore, $\Pr(\mathsf{coll}) \leq \Pr(\mathsf{coll}|\neg\mathsf{bad}) + \Pr(\mathsf{bad}) \leq \dfrac{6n^5q^2 + 3n^4q^2 + 2n^4q + 13q^2}{2^n}$.

Note that we have bound the proper collision probability at the root for ABR_3. Since $B0$ does not occur, collision does not occur at the leaf node. As seen in section 5.2, the probability that proper collision occurs at node $(2,1)$ (resp. $(2,2)$) is bounded above by $\frac{n^2q^2}{2^n}$. Hence, the theorem is proved.

7 Conclusion

In this paper, we revisit the collision security of the ABR hash. We found that there is a serious gap in the analysis of collision security. Some missing and important cases have also been identified. In this paper, we have shown collision security for level 3. Several new bad events have been identified in ABR_3 which were not considered for the general hash. We leave the collision security analysis open for general hash. Thus, the optimality of Stam's bound remains open for an arbitrary domain hash.

We have also found that the ABR hash cannot have any non-trivial local opening which can give birthday bound security. This shows a limitation in terms of applications in local opening. In particular, the efficient local opening proposed by [1] can be broken in $O(2^{n/2l})$ query complexity.

References

1 Elena Andreeva, Rishiraj Bhattacharyya, and Arnab Roy. Compactness of hashing modes and efficiency beyond merkle tree. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, pages 92–123. Springer, 2021.

2 John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541. Springer, 2005.

3 Ivan Damgård. A design principle for hash functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

4 Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi. T5: Hashing five inputs with three compression calls. In *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, pages 24:1–24:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Bart Mennink and Bart Preneel. Efficient parallelizable hashing using small non-compressing primitives. *Int. J. Inf. Sec.*, 15(3):285–300, 2016.

6 Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society, 1980.

7 Ralph C. Merkle. One way hash functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

8 Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 433–450. Springer, 2008.

9 Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2008.

10 Thomas Shrimpton and Martijn Stam. Building a collision-resistant compression function from non-compressing primitives. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654. Springer, 2008.

11 Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 397–412. Springer, 2008.

12 John P. Steinberger. Stam's collision resistance conjecture. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615. Springer, 2010.

13 John P. Steinberger, Xiaoming Sun, and Zhe Yang. Stam's conjecture and threshold phenomena in collision resistance. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405. Springer, 2012.

14 David A. Wagner. A generalized birthday problem. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.