# $T_5$: Hashing Five Inputs with Three Compression Calls

Yevgeniy Dodis[1], Dmitry Khovratovich[2], Nicky Mouha[3], and Mridul Nandi[4]

[1] New York University
dodis@cs.nyu.edu
[2] Ethereum Foundation and Dusk Network
khovratovich@gmail.com
[3] Strativia
nicky@mouha.be
[4] Indian Statistical Institute
mridul.nandi@gmail.com

**Abstract.** Given $2n$-to-$n$ compression functions $h_1, h_2, h_3$, we build a new $5n$-to-$n$ compression function $T_5$, using only 3 compression calls:

$$T_5(m_1, m_2, m_3, m_4, m_5) := h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5) \oplus m_5$$

We prove that this construction matches Stam's bound, by providing $\tilde{O}(q^2/2^n)$ collision security and $O(q^3/2^{2n} + nq/2^n)$ preimage security (the latter term dominates in the region of interest, when $q < 2^{n/2}$). In particular, it provides birthday security for hashing 5 inputs using three $2n$-to-$n$ compression calls, instead of only 4 inputs in prior constructions.

Thus, we get a sequential variant of the Merkle-Damgård (MD) hashing, where $t$ message blocks are hashed using only $3t/4$ calls to the $2n$-to-$n$ compression functions; a 25% *saving* over traditional hash function constructions. This time reduces to $t/4$ (resp. $t/2$) sequential calls using 3 (resp. 2) parallel execution units; saving a factor of 4 (resp. 2) over the traditional MD-hashing, where parallelism does not help to process one message.

We also get a novel variant of a Merkle tree, where $t$ message blocks can be processed using $0.75(t-1)$ compression function calls and depth $0.86 \log_2 t$, thereby *saving* 25% *in the number of calls and* 14% *in the update time* over Merkle trees. We provide two modes for a local opening of a particular message block: conservative and aggressive. The former retains the birthday security, but provides longer proofs and local verification time than the traditional Merkle tree.

For the aggressive variant, we reduce the proof length to a 29% overhead compared to Merkle trees ($1.29 \log_2 t$ vs $\log_2 t$), but the verification time is now 14% *faster* ($0.86 \log_2 t$ vs $\log_2 t$). However, birthday security is only shown under a plausible conjecture related to the 3-XOR problem, and only for the (common, but not universal) setting where the root of the Merkle tree is known to correspond to a valid $t$-block message.

## 1 Introduction

A fundamental problem in cryptography is the construction of a hash function using idealized building blocks. A natural way to approach this problem is to use $\lambda n$-to-$n$-bit compression functions. Two well-known and widely-deployed constructions follow this approach:

- the *Merkle-Damgård construction* [10, 20], a sequential construction that is used in hash functions such as MD5, SHA-1 and SHA-2, and
- the *Merkle tree* [19], a parallel construction used in hash-based signatures (of interest due to their post-quantum security), version control systems such as git, and cryptocurrencies such as Ethereum.

The collision resistance of the Merkle-Damgård construction and the Merkle tree can be proven, based on the collision-resistance of the compression functions. The number of compression function calls is (essentially) the same for both constructions. For example, setting $\lambda = 2$, which is the focus of this work,[5] they both process $t$ message blocks using $t$ and $(t-1)$ compression function calls, respectively.

New Compression Function $T_5$. In this paper, we introduce the $T_5$ construction (see Fig. 1) that processes five message blocks using three $2n$-to-$n$-bit compression function calls, thereby

[5] Without loss of generality, all our results easily generalize to any $\lambda \geq 2$.

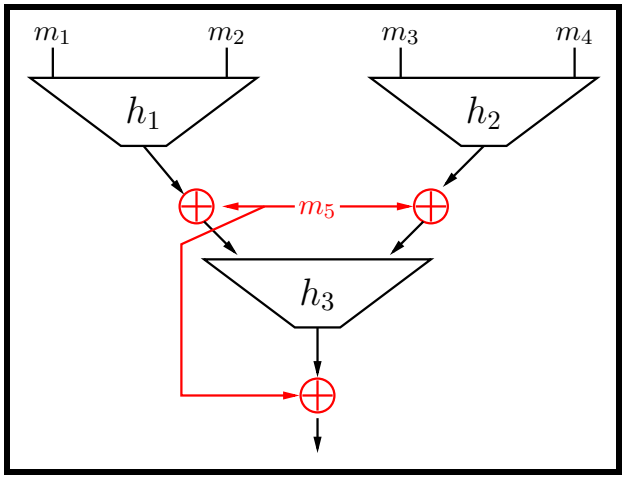


**Fig. 1.** The $T_5$ construction with five message blocks $m_1, m_2, m_3, m_4, m_5$ and three compression function calls.

improving over the state-of-the-art of Merkle-Damgård (with $IV$ counted as message block) and Merkle trees by processing an additional message block with the same number of compression function calls and essentially the same level of collision security.

Although $T_5$ is of independent theoretical interest to the construction of a compression function, we will also investigate Merkle-Damgård and Merkle trees when instantiated with $T_5$.

$T_5$ WITH MERKLE-DAMGÅRD. Our variant of the Merkle-Damgård construction, depicted in Figure 4, processes $t$ message blocks using $3t/4$ calls to the $2n$-to-$n$ compression functions. If the chaining value is provided as $m_5$ in $T_5$, then $h_1$ and $h_2$ can not only be computed in parallel, but independently of the chaining value. This allows a fully parallel implementation of $h_1$, $h_2$, and $h_3$ (with $h_3$ "one-round behind"), which is four times faster than MD which requires four sequential compression function calls to process a single input of the same length.

$T_5$ WITH MERKLE TREES. For our variant of Merkle trees, depicted in Figure 5, we will consider how they are often used in practice: for proof-of-inclusion of data in a larger set.

A *full opening* of a Merkle tree corresponds to the list of all message blocks, which can be used to verify that the message corresponds to a given hash value. An advantage of Merkle trees is it is possible to provide a *local opening*: to verify that one message block belongs to the tree, it suffices to provide a list of compression function outputs that is proportional to the depth of the tree.

These two types of openings give rise to three different notions of collision resistance for trees:

- *full-full* collision resistance, the "traditional" notion that requires finding two distinct messages that result in the same hash value,
- *local-local* collision resistance, where the goal is to find two local openings with the same hash value,
- *full-local* collision resistance, the setting of finding a collision between a full tree and a local opening.

The full-local setting is relevant in the common scenario where a hash value is honestly computed, but the proof is composed by an untrusted party. This happens, for example, when a user sends a message to a cloud server after hashing it, and then later wants to retrieve some message block from the server. Another natural application is the Merkle accumulator, where a protocol accumulates message blocks using a Merkle tree, and later parties provide proofs that a message block is in the tree.

Standard Merkle trees provide the same level of security under all three notions of collision resistance, and the same holds for our Merkle tree variant using $T_5$ if all four siblings are opened. In this case, our variant of the Merkle tree will process $t$ message blocks using $0.75(t-1)$ instead of $t-1$ compression function calls, and depth $0.86 \log_2 t$ instead of $\log_2 t$. Due to the need to open four siblings, the opening proof will increase from $\log_2 t$ to $1.72 \log_2 t$, and the verification time increases as well from $\log_2 t$ to $1.29 \log_2 t$

However, we also propose an aggressive variant of our construction that only opens three siblings. See Figure 3. When this saving of one element for $T_5$ is translated to the whole Merkle tree, one gets a smaller local opening proof of $1.29 \log_2 t$, and a shorter verification time of $0.86 \log_2 t$. We prove that this more aggressive variant nevertheless provides the same full-local collision resistance, under a conjecture related to the 3-XOR problem. For local-local collision resistance, we prove security up to $2^{n/3}$ under a conjecture related to the 4-XOR problem. The $k$-XOR problem is the subject of the well-studied generalized birthday problem by Wagner [33], and used in proof-of work algorithms such as Equihash. A full comparison of these three constructions will be given in Table 1 of Section 7.2.

Our Results for $T_5$. We prove the following security results for $T_5$ in terms of adversarial advantage after $q$ queries to the inner compression functions, and provide matching attacks in Section 8:[6]

- $q^2/2^n$ collision resistance, i.e., full-full collision resistance (CR) security (Theorem 1);
- $q^3/2^{2n} + q/2^n$ preimage resistance (Theorem 2).
- $q^2/2^n$ full-local CR security under a conjecture related to the 3-XOR problem (Proposition 1);
- $q^3/2^n$ full-local CR security unconditionally, i.e., 128-bit security for $n = 384$ (Theorem 3);
- $q^3/2^n$ local-local CR security under a conjecture related to the 4-XOR problem (Proposition 2);
- $q^4/2^n$ local-local CR security unconditionally, i.e., 128-bit security for $n = 512$ (Theorem 3).

These results almost immediately imply corresponding security claims for our Merkle-Damgård variant (see Section 6) and our Merkle tree variant (see Section 7 and Table 1).

## 2 Related Work

The design of a hash function is usually based on one or more *primitives* with fixed-length inputs and outputs. Historically, the most common choice for these primitives were block ciphers. This gives rise to the following question: how can we construct a hash function with the minimum number of block cipher calls?

This question motivated a significant research effort into efficient block-cipher-based hash function constructions. Block ciphers such as Triple-DES and AES have a block size of 64 and 128 bits respectively, which may not provide sufficient collision security when used in the Merkle-Damgård construction. Therefore, one line of work focuses on combining smaller primitives to produce a wider hash function. Results of interest include the Knudsen-Preneel construction based on linear error correcting codes [15], and a double-length construction by Nandi et al. [22] that was generalized by Peyrin et al. [26], and by Seurin and Peyrin [28], which interestingly also links the security to a conjecture related to the 3-sum problem.

A related line of research attempted to improve upon the Merkle-Damgård construction to process additional message blocks. A brief overview of some constructions and an impossibility result was given by Black et al. [6, 7]. More specifically, they considered hash functions that make one block cipher call (under a small set of keys) for each message block to be hashed, and showed that all such constructions are vulnerable to a simple attack.

This work was later generalized by Rogaway and Steinberger [27], and refined in subsequent papers by Stam [29], and by Steinberger et al. [30, 31]. This result, commonly known as "Stam's bound," puts a limit on the efficiency (in terms of primitive calls) of any secure hash function construction.

Stam's bound states that there always exists a collision attack and a preimage attack with at most $2^{n(\lambda-(t-0.5)/r)}$ and $2^{n(\lambda-(t-1)/r)}$ queries respectively on a $tn$-to-$n$-bit hash function making $r$ calls to $\lambda n$-to-$n$-bit compression functions. We have $t = 5$, $\lambda = 2$, and $r = 3$ in the case of $T_5$, thereby showing that we cannot hope to do better than $2^{n/2}$ collision and $2^{2n/3}$ preimage security.

As explained by Stam [29], the bound applies to hash functions that satisfy the *uniformity assumption*, and applies to cryptographic permutations ($\lambda = 1$) as well as compressing primitives ($\lambda \geq 1$). However, the main focus of this line of work had been on combining smaller non-compressing primitives (see e.g., Mennink and Preneel [17, 18]).

[6] For simplicity of reading, we only list dominant terms, and ignore constant and even small $\mathsf{poly}(n)$ factors; i.e., omit $\tilde{O}$ notation below.

Recently, McQuoid et al. [16] provided a general framework to prove the collision and second-preimage security of various hash functions. Our $\mathrm{T}_5$ construction is covered by their framework. Unfortunately, their framework does not provide tight collision and second-preimage security bounds for $\mathrm{T}_5$.

Lastly, we recall that a series of papers have investigated optimal trade-offs between time and space for Merkle tree traversal, e.g., Jakobsson et al. [14], Szydlo [32], and Berman et al. [4]. Given that we propose $\mathrm{T}_5$ inside a standard Merkle tree, these trade-offs can also be directly applied to the constructions in this paper. We would also like to mention Haitner et al. [12]'s construction which only has depth one, at the cost of making significantly more calls than the standard Merkle tree.

## 3 Preliminaries

### 3.1 Notation

If $S$ is a set, $x \xleftarrow{\$} S$ 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 $\mathsf{A}(x)$, respectively.

For positive integers $m, n$, we let $\mathrm{Func}(m,n)$ denote the set of all functions mapping $\{0,1\}^m$ into $\{0,1\}^n$. We write $h \xleftarrow{\$} \mathrm{Func}(m,n)$ to denote random sampling from the set $\mathrm{Func}(m,n)$ and assignment to $h$, and say that $h$ is modeled as an ideal hash function. For fixed $m$ and $n$, such modeling is attempting to approximate the security of real-world, keyless, fixed-input-size compression functions, such as the compression function of SHA-2.

### 3.2 Security Definitions of Hash Functions

An *adversary* $\mathsf{A}$ is a probabilistic algorithm, possibly with access to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_\ell$ denoted by $\mathsf{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_\ell}$. Our definitions of collision (Coll), and preimage (Pre) security are given for any general fixed-input length hash function $H$ built upon the compression functions $h_i$ for $i = 1, \ldots, \ell$ where $h_i$ are modeled as ideal functions. Namely, for a fixed adversary $\mathsf{A}$ and for all $i = 1$ to $\ell$ with $h_i \xleftarrow{\$} \mathrm{Func}(2n,n)$, we define the following advantage functions:

$$\mathbf{Adv}^{\mathrm{Coll}}_{H}(\mathsf{A}) = \Pr\Big[ H^{h_1,\ldots,h_\ell}(M) = H^{h_1,\ldots,h_\ell}(M') \text{ and } M \neq M' \\ \mid (M, M') \xleftarrow{\$} \mathsf{A}^{h_1,\ldots,h_\ell}() \Big]$$

and

$$\mathbf{Adv}^{\mathrm{Pre}}_{H}(\mathsf{A}) = \Pr\Big[ H^{h_1,\ldots,h_\ell}(M) = H^{h_1,\ldots,h_\ell}(M') \\ \mid M \xleftarrow{\$} \mathcal{M}_H, M' \xleftarrow{\$} \mathsf{A}^{h_1,\ldots,h_\ell}(H^{h_1,\ldots,h_\ell}(M)) \Big]$$

We define the $\mathbf{Adv}^{\mathrm{atk}}_{H}(q)$ against the $\mathrm{atk} = \{\mathrm{Coll}, \mathrm{Pre}\}$-security of $H$ as the maximum advantage over all adversaries making at most $q$ total queries to its oracles.

### 3.3 Local Opening Security

We define local opening security of a hash function output (viewed as a commitment of a message). Given a function $H$ built upon compression functions $h_1, h_2, \ldots$ where $h_i$ are all modeled as ideal functions, a local opening $\mathsf{Open}^{h_1,\cdots}(\cdot,\cdot)$ for $H^{h_1,\cdots}$ maps a pair $(M,i)$ to $\pi$ (called proof) where $M = (m_1, m_2, \ldots, m_c)$ is a message (a tuple of blocks) and $1 \leq i \leq c$ is an index.

Correctness of Local Opening. There is an efficient function $\mathsf{Ver}^{h_1,\cdots}$ such that for all message $M$, all index $i$,

$$\mathsf{Ver}^{h_1,\cdots}(i, m_i, \mathsf{Open}^{h_1,\cdots}(M,i), H(M)) = 1.$$

Security of Local Opening. We provide two notions of local opening security. For the stronger variant, which we call "local-local," the adversary wins if it produces an output $f$ corresponding to two contradicting local openings for some position $i$. For the weaker variant, which we call "full-local," the adversary wins if it produces an output $f$ corresponding to a local opening contradicting a full opening.

**Definition 1 (local-local and full-local opening advantage).** *Let $H$ be a hash function and* $\mathsf{Open}$ *is a correct local opening for $H$ with* $\mathsf{Ver}$ *is the verification function. For any adversary* $\mathsf{A}$*, we define the local-local opening advantage as*

$$\mathbf{Adv}_H^{\textit{local-local}}(\mathsf{A}) = \Pr\Big[\mathsf{Ver}(i,m,\pi,f) = \mathsf{Ver}(i,m',\pi',f) = 1, m \neq m' \mid (i,m,m',\pi,\pi',f) \xleftarrow{\$} \mathsf{A}^{h_1,\cdots}\Big]$$

*We define full-local opening advantage (a weaker variant of the above) of* $\mathsf{A}$ *as*

$$\mathbf{Adv}_H^{\textit{full-local}}(\mathsf{A}) = \Pr\Big[\mathsf{Ver}(i,m',\pi',H(M)) = 1, m' \neq m_i \mid (i,M,m',\pi') \xleftarrow{\$} \mathsf{A}^{h_1,\cdots}\Big]$$

*For a function $H$ with local opening algorithms* $(\mathsf{Open},\mathsf{Ver})$ *and attack $z \in \{$local-local, full-local$\}$, we define $\mathbf{Adv}_H^z(q) = \max_{\mathsf{A}} \mathbf{Adv}_H^z(\mathsf{A})$ as the maximum advantage over all adversaries making at most $q$ total queries to its oracles.*

Intuitively, the weaker definition protects against situations where the initial commitment $f$ was honestly produced, using some long message $M$. Thus, a contradictory local opening will result in finding a collision between a local opening and a full opening. In contrast, the (traditional) stronger definition is directly concerned with somebody producing two contradictory local openings.

By-Pass Hash Computation. We say that $H$ has a *by-pass computation* $H_i$ corresponding to a local opening $\mathsf{Open}$ for a fixed index $1 \leq i \leq c$, if for all $M$,

$$H_i^{h_1,\cdots}(m_i, \mathsf{Open}^{h_1,\cdots}(M,i)) = H^{h_1,\cdots}(M).$$

In other words, given a proof (output of the $\mathsf{Open}$) 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 presence of by-pass computations $\mathcal{H} = \{H_i\}$ for all indices lead to a natural verification algorithm as follows: $\mathsf{Ver}(i,m,\pi,f) = 1$ whenever $H_i^{h_1,\cdots}(m,\pi) = f$. For a fixed index $i$, we define the *cross-collision* advantage between the hash function $H$ and a by-pass computation $H_i$ as

$$\mathbf{Adv}_{H,H_i}^{\mathrm{Coll}}(\mathsf{A}) = \Pr\Big[H(M) = H_i(m',\pi) \text{ and } M_i \neq m' \mid (M,m',\pi) \xleftarrow{\$} \mathsf{A}^{h_1\cdots}\Big]$$

Similarly, we define the *inter-collision* advantage for by-pass computation $H_i$ as

$$\mathbf{Adv}_{H_i}^{\mathrm{Coll}}(\mathsf{A}) = \Pr\Big[H_i(m,\pi) = H_i(m',\pi') \text{ and } m \neq m' \mid (m,\pi,m',\pi) \xleftarrow{\$} \mathsf{A}^{h_1\cdots}\Big]$$

Let $\mathcal{H} = \{H_i\}$ be the family of by-pass computations for all indices $i$. We define

$$\mathbf{Adv}_{H,\mathcal{H}}^{\mathrm{Coll}}(q) = \max_{\mathsf{A}} \max_i \mathbf{Adv}_{H,H_i}^{\mathrm{Coll}}(\mathsf{A}) \quad \text{and} \quad \mathbf{Adv}_{\mathcal{H}}^{\mathrm{Coll}}(q) = \max_{\mathsf{A}} \max_i \mathbf{Adv}_{H_i}^{\mathrm{Coll}}(\mathsf{A})$$

as the maximum advantage over all by-pass computations for all adversaries making at most $q$ total queries to its oracles.

Now we make a simple observation when by-pass computations $\mathcal{H} = \{H_i\}$ for all indices exist for a hash function $H$, the induced verification procedure $\mathsf{Ver}$ satisfies

$$\mathbf{Adv}_H^{\textit{full-local}}(q) \leq \mathbf{Adv}_{H,\mathcal{H}}^{\mathrm{Coll}}(q) \text{ and } \mathbf{Adv}_H^{\textit{local-local}}(q) \leq \mathbf{Adv}_{\mathcal{H}}^{\mathrm{Coll}}(q) \tag{1}$$

The above observation helps us to reduce the local opening security to cross-collision or inter-collision security problem for the family $\mathcal{H}$. As all the function $H_i$ in this family are often symmetric, a proof for a fixed function $H_i$ implies the one for the entire family $\mathcal{H}$.

## 4 Construction $\mathrm{T}_5$

We define $\mathrm{T}_5 : \{0,1\}^{5n} \to \{0,1\}^n$ based on the $2n$-to-$n$-bit compression functions $h_1, h_2, h_3$ as follows:

$$\mathrm{T}_5(m_1, m_2, m_3, m_4, m_5) := h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5) \oplus m_5$$

For all our proofs we will assume that the compression functions $h_i$ for $i = 1$ to 3 are ideal functions.

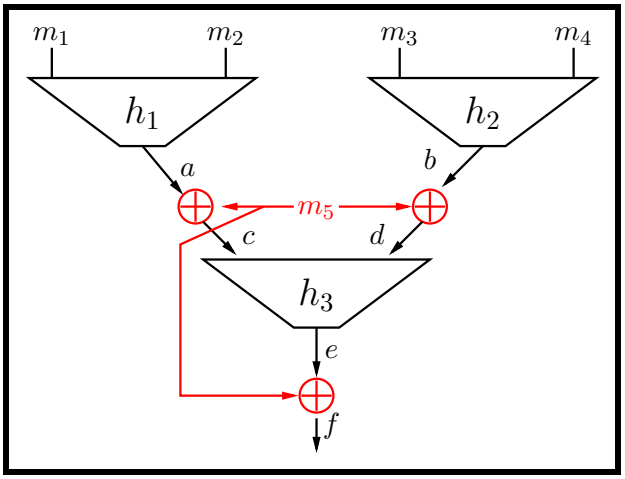


**Fig. 2.** Modified 2-level Merkle tree $\mathrm{T}_5(m_1, m_2, m_3, m_4, m_5)$ with an extra input $m_5$ for the same 3 hash calls.

*Notation.* As shown in Figure 2, we use the variables

- $m_1$ and $m_2$ (resp. $m_3$ and $m_4$) to denote the left and right halves of various inputs to $h_1$ (resp. $h_2$);
- $a$ (resp. $b$) to denote various outputs of $h_1$ (resp. $h_2$);
- $c$ and $d$ to denote the left and right halves of various inputs to $h_3$;
- $e$ to denote various outputs of $h_3$;
- $M = (m_1, m_2, m_3, m_4, m_5)$ to denote various inputs to $\mathrm{T}_5$;
- $f$ to denote various outputs of $\mathrm{T}_5$.

Hence, a valid computation of $\mathrm{T}_5(M) = \mathrm{T}_5(m_1, \ldots, m_5)$ proceeds as follows:

1. Set $a = h_1(m_1, m_2)$, $b = h_2(m_3, m_4)$.
2. Set $c = a \oplus m_5$, $d = b \oplus m_5$.
3. Set $e = h_3(c, d)$, and output $f = e \oplus m_5$.

We say that a triple of queries $((m_1, m_2), a)$ to $h_1$, $((m_3, m_4), b)$ to $h_2$, and $((c, d), e)$ to $h_3$ is *consistent* if

$$a \oplus b = c \oplus d, \tag{2}$$

in which case we define $m_5 = a \oplus c = b \oplus d$, and say that this consistent triple of queries (uniquely) defines a valid $\mathrm{T}_5$ evaluation $(M, f)$, where $M = (m_1, m_2, m_3, m_4, m_5)$ and $f = e \oplus a \oplus c = e \oplus b \oplus d$.

*Main results of the section.* The main result of this paper is to provide the collision and preimage security of the $\mathrm{T}_5$ hash function. The following theorem shows that $\mathrm{T}_5$ achieves nearly birthday collision security, despite hashing one more input than the traditional Merkle-Damgård function of depth 2.

**Theorem 1.** *The* $\mathrm{T}_5$ *construction achieves nearly birthday-bound collision security:*

$$\mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{T}_5}(q) \leq \frac{(n^2 + 10)q^2}{2^n} \tag{3}$$

The full formal proof of this result is somewhat subtle, and will be given in the Appendix (see App. A). But an informal proof intuition for a representative special case will be given in Section 4.1.

As our second main result, we also show that $\mathrm{T}_5$ maintains nearly optimal preimage security $\tilde{O}(q/2^n)$ for $q < 2^{n/2}$, which means it offers optimal preimage *and* collision-resistance security in the common range of $q < 2^{n/2}$.

However, even when $q$ grows above $2^{n/2}$, $\mathrm{T}_5$ still offers non-trivial security $O(q^3/2^{2n})$ for values of $q < 2^{2n/3-1}$, which is likely sufficient for most applications. As we show in Section 8.2, $\mathrm{T}_5$ is indeed not preimage resistant when $q > 2^{2n/3}$, so our result is tight.

**Theorem 2.** *Assuming $q \leq 2^{2n/3-1}$, the $\mathrm{T}_5$ construction achieves the following preimage security:*

$$\mathbf{Adv}^{\mathrm{Pre}}_{\mathrm{T}_5}(q) \leq \frac{2q^3}{2^{2n}} + O\left(\frac{qn}{2^n}\right) \tag{4}$$

We give a formal proof in the Appendix (see App. B), but present some proof intuition (for an important special case) in Section 4.2, similar to what was done in Section 4.1.

### 4.1 Proof Intuition for Collision Resistance of $\mathrm{T}_5$

Below we give the proof intuition for the simple, but natural special case where the adversary $\mathsf{A}$ makes all of its queries to $h_1$ and $h_2$ before any query to $h_3$ is made. As we explain in the formal proof in the Appendix (see App. A), this assumption is with a *significant* loss of generality, and several special arguments are needed to cover the fully general case. However, this simplified case will already demonstrate some of the main arguments of our analysis.

The proof will roughly consist in arguing that $\mathsf{A}$, making $q$ total hash queries, is unlikely — up to birthday advantage — to succeed in the following four tasks.
(These tasks are formalized later in Propositions 3,4,5,6.)

1. **Task 1**: finding any simple collision in $h_1$ or $h_2$.[7]
2. **Task 2**: finding some value $z$ for which there exist more than $n$ distinct pairs of queries $((m_1, m_2), (m_3, m_4))$ to $h_1$ and $h_2$ result in[8]

$$h_1(m_1, m_2) \oplus h_2(m_3, m_4) = z$$

3. **Task 3**: generate more than $nq$ valid evaluations of $\mathrm{T}_5$.
4. **Task 4**: generate a non-trivial collision of $\mathrm{T}_5$.

The argument of each subsequent task will inductively assume that the adversary indeed failed in the previous task.

For Task 1, this is the trivial birthday bound on $h_1$ or $h_2$.

For Task 2, we will formally define the set $C_{12}(z)$ to consist of pairs of queries $((m_1, m_2), a)$ to $h_1$ and $((m_3, m_4), b)$ satisfying $a \oplus b = z$. Using the fact that no simple collisions in $h_1$ and $h_2$ are found, it is easy to see that each of the $n$ queries to $h_1$ and $h_2$ inside $C_{12}(z)$ must be distinct. Moreover, the latter of the two queries $((*, a), (*, b)) \in C_{12}(z)$ must collide with a fixed value $z$ plus the former of the two queries. E.g., if the query $(*, a)$ to $h_1$ was made before $(*, b)$ to $h_2$, this tuple will fall inside $C_{12}(z)$ only if $b = z \oplus a$, which happens with probability $2^{-n}$. Taking the union bound over all values $z$ and all possible choices of $2n$ out of $q$ queries to be included inside $C_{12}(z)$, we see that

$$\Pr\left[\,\exists z \text{ s.t. } |C_{12}(z)| \geq n\,\right] \leq 2^n \cdot q^{2n} \cdot \left(\frac{1}{2^n}\right)^n \leq \left(\frac{2q^2}{2^n}\right)^n \ll O\left(\frac{q^2}{2^n}\right)$$

For Task 3, we will use our simplifying assumption that $\mathsf{A}$ makes all of its queries to $h_1$ and $h_2$ before any query to $h_3$ is made. In this case, all consistent triples of queries to $h_1$, $h_2$ and $h_3$ defining a valid input-output $(M, f)$ to $\mathrm{T}_5$ get created by making a call to $h_3$. For each of at most $q$ such queries to $h_3$ on some input $(c, d)$, we claim that this query will "match up" with a pair

[7] For the general case, we will also need no collisions in a slightly modified variant of $h_3$.
[8] For the general case, we will also need similar guarantees for the combinations of $h_1 + h_3$ and $h_2 + h_3$.

of earlier queries $(*, a)$ to $h_1$ and $(*, b)$ to $h_2$ only if $m_5 = a \oplus c = b \oplus d$, which is equivalent to $a \oplus b = c \oplus d$, which means that

$$((*, a), (*, b)) \in C_{12}(c \oplus d)$$

But we already assumed that $|C_{12}(z)| \leq n$ for all $z$, meaning that each query $(c, d)$ can form a consistent tuple with at most $n$ pairs of queries to $h_1$ and $h_2$. Summing over all (up to) $q$ queries to $h_3$, the total number of evaluations will be at most $nq$.[9]

Finally, for Task 4 that we care about, we will once again use our simplifying assumption. In particular, under our assumption, such a collision can only be caused by a call to $h_3$ on some input $(c, d)$. From the previous argument, we already know that this query will "match up" with a pair of earlier queries $(*, a)$ to $h_1$ and $(*, b)$ to $h_2$ only if $((*, a), (*b)) \in C_{12}(c \oplus d)$, meaning there are at most $n$ new evaluations of $\mathrm{T}_5$ caused by this query. Also, any two of these $n$ new evaluations cannot collide among themselves, as they have two different values $a \neq a'$ (remember, no collisions in $h_1$), so the final outputs $f = h_3(c, d) \oplus c \oplus a \neq h_3(c, d) \oplus c \oplus a' = f'$. Thus, the only chance the adversary has is if one of these $n$ new evaluations of $\mathrm{T}_5$ (call the output $f$) collides with one of at most $nq$ already defined previous evaluations $f'$ of $\mathrm{T}_5$.

But each of the $n$ new output values $f$ will be *individually random*, as it is equal to random $e = h_3(c, d)$ plus $m_5 = a \oplus c$. Hence, this individually random $f$ can collide with the previously defined output $f'$ of $\mathrm{T}_5$ with probability at most $nq/2^n$, because from failing Task 3 we know there are at most $nq$ previous evaluations of $\mathrm{T}_5$ completed so far. Taking the union bound over $n$ values of $f$, and $q$ queries to $h_3$, the final bound $O(n^2q^2/2^n)$ follows.

*General case.* We will not fully detail the general case (see full argument in the Appendix (see App. A)), but briefly demonstrate that making queries to $h_1$ or $h_2$ after some queries to $h_3$ could be potentially helpful to the adversary, and will require adjustments to our proof strategy above.

For example, our proof sketch above showed that, modulo very rare events, the number of valid evaluations of $\mathrm{T}_5$ can increase by at most $n$ for each new query to $h_3$. However, imagine that $\mathsf{A}$ first makes a query to $h_2$ with output $b$. Then $\mathsf{A}$ can make $\Omega(q)$ queries $(c_i, d_i)$ all satisfying $c_i \oplus d_i = z$ (for some $z$). Now, a query to $h_1$ (made after these $\Omega(q)$ queries to $h_3$) has a chance to simultaneously match with $\Omega(q) \gg n$ tuples $((*, b), ((c_i, d_i), *)$. Of course, in order for this to happen, the random answer $a$ must match $b \oplus z$, which happens with tiny probability. In fact, we could apply Markov's equality to argue that the probability a query to $h_1$ will produce more than $n$ new evaluation points is at most (what turns out to be by an easy calculation) $O(q^2/2^n)$. By itself, this is good enough, but it will not "survive" a union bound over up to $q$ potential queries to $h_1$. Instead, we will use linearity of expectation to make a global, "stochastic" argument that *all* such (up to) $q$ queries to $h_1$ and $h_2$ will define more than $nq$ new evaluations with at most "birthday" probability. See Proposition 5 for the details.

Overall, the full proof in the general case will be noticeably more subtle than the proof intuition given above, but will still follow the same high-level structure.

### 4.2 Proof Intuition for Preimage Resistance of $\mathrm{T}_5$

As in Section 4.1, we will only consider the special case when the adversary $\mathsf{A}$ makes all of its queries to $h_1$ and $h_2$ before any query to $h_3$ is made, as it will contain most of the ideas needed in the general proof. Also, we will assume that $q = \Omega(\sqrt{n} \cdot 2^{n/2})$, as this is the case where the "unexpected" term $q^3/2^{2n}$ appears.[10]

In this setting, there are several differences from the case of collision-resistance we considered so far. First, $\mathsf{A}$ is given a specific target $f$ to invert. In particular, we will not care about local collisions in functions $h_1$, $h_2$, and the $c$- or $d$-"shifted" versions of $h_3$, as such collisions will happen, but will not help the adversary invert $f$. Instead, we will care that in each new call to $h_3(c, d)$, the number of valid new evaluations of $\mathrm{T}_5$ will be not much higher than what we expect. Recall, in our special case of only $h_3$ queries causing all new evaluations of $\mathrm{T}_5$, this number of new evaluations

[9] As we will see, in the general case a very different proof strategy will be needed to extend this argument to valid evaluations of $\mathrm{T}_5$ completed by calls to $h_1$ and $h_2$.

[10] Our general proof will not treat this case separately, but the calculations are slightly easier to write for the "beyond-birthday" case.

is bounded by $|C_{12}(c \oplus d)|$, where $C_{12}(z)$ is the set of pairs of queries $((m_1, m_2), a)$ to $h_1$ and $((m_3, m_4), b)$ satisfying $a \oplus b = z$. And since each such evaluation defines an individually random output value $f' = h_3(c, d) \oplus c \oplus a$, the probability this $f'$ matches $f$ is $2^{-n}$, meaning $\mathsf{A}$'s overall chance to invert $f$ in this query is $|C_{12}(c \oplus d)|/2^n$.

Of course, the adversary can select *any* value of $z = c \oplus d$ to make sure it chooses the largest set $C_{12}(z)$. Thus, if we want to upper bound $\mathsf{A}$'s probability of success, we must argue that $K = \max_z |C_{12}(z)|$ is not much higher than its expectation with high probability. In the collision resistance proof, we manage to upper bound $K \leq n$ with "good enough" probability $(q^2/2^n)^n$. Indeed, this was enough to withstand the union bound over $z$ to give the final probability $(2q^2/2^n)^n \leq 2q^2/2^n$ that $K \geq n$ in that setting.

We will do a similar union bound in our case as well, except that we have $q \gg 2^{n/2}$, so even in the best case scenario we expect $|C_{12}(z)| \geq q^2/2^n \gg n$ for any given $z$, let alone the $z$ chosen by the adversary. Moreover, the final birthday bound will not be good enough for us in this setting as well. But first let us optimistically assume that for any fixed $z$, we managed to get the following very strong concentration bound:[11]

$$\Pr\left[\, |C_{12}(z)| \geq \frac{2q^2}{2^n} \,\right] \leq \frac{1}{2^{2n}} \tag{5}$$

Then we are done, because we can take the union bound over $z$ to conclude that $\Pr\left[\, K \geq 2q^2/2^n \,\right] \leq 2^{-n}$. And, finally, there will be at most $2q^2/2^n$ new evaluations $f'$ per each query to $h_3$. Thus, taking the union bound over at most $q$ such queries, $\mathsf{A}$'s overall inversion probability (ignoring $2^{-n}$ failure event above) is upper bounded by:

$$q \cdot \frac{2q^2}{2^n} \cdot \frac{1}{2^n} = \frac{2q^3}{2^{2n}}$$

*High concentration bound via tabulation hashing.* So it remains to argue the high concentration bound in (5). This turns out to be much harder than in the collision-resistance case, where the bound we needed was the much weaker $O((q^2/2^n)^n)$, which is meaningless when $q > 2^{n/2}$.

The next naive attempt is to write $|C_{12}(z)|$ as a sum of $q^2$ indicator variables $X_{ij}$, equal to 1 is the $i$-th output $a_i$ of $h_1$ and the $j$-th output $b_j$ of $h_2$ satisfy $a_i \oplus b_j = z$. And then try to use a Chernoff bound to argue that the probability that the sum of these indicator variables is twice as large as its expectation is exponentially low. Unfortunately, the random variables $X_{ij}$ are not even 4-wise independent; e.g., $(a_1 \oplus b_1) \oplus (a_1 \oplus b_2) \oplus (a_2 \oplus b_1) \oplus (a_2 \oplus b_2) = 0$. So we cannot apply the Chernoff bound, and the Chebyshev inequality for pairwise independent random variables is not strong enough. Indeed, the question of getting our concentration bound turned out to be quite deep.

Fortunately, the setting we need turns out to be equivalent to the classical hashing problem, called *simple tabulation hashing*, introduced in the seminal paper of Carter and Wegman [9]. Applied to our setting, given two random "hash tables" $T_1$ and $T_2$ with range of size $N = 2^n$, tabulation hashing would map a "ball" $y = (u, v)$ into a "bin" $z = T_1[u] \oplus T_2[v]$. The classical question studied by tabulation hashing is to upper bound occupancy of any such bin $z$ after some $Q$ balls are thrown using tabulation hashing. We defer the details to the formal proof in Section 4.2, but point out that in our setting the tables are implemented using hash functions $h_1$ and $h_2$, and the number of balls $Q \leq q^2$ corresponds to all pairs of queries to $h_1$ and $h_2$.

Designing strong enough concentration bound for tabulation hashing (which is exactly what we need!) was an open problem for many years, until the breakthrough result of Pătraşcu and Thorup [24] showed a Chernoff-type concentration bound which enables us to show (5), and thus complete the proof.

## 5 Aggressive Opening for $\mathrm{T}_5$

In this section we describe a non-trivial opening for $\mathrm{T}_5$. We first note that a straightforward way to open a block $m_i$ in $\mathrm{T}_5$ is to provide all four siblings $m_{j \neq i}$. For a single $\mathrm{T}_5$ the full-local and the

[11] This bound, as stated, is only true if $q = \Omega(\sqrt{n} \cdot 2^{n/2})$. In the general case the term becomes $|C_{12}(z)| \geq \Omega(n) + 2q^2/2^n$, as we expect $\Omega(n)$ multi-collisions already when $q \approx 2^{n/2}$.

local-local security (Section 3.3) definitions are the same and correspond to the collision security of $\mathrm{T}_5$ which we have already studied. The performance of this method in a full tree is somewhat less attractive and is given in detail in Section 7.2. Thus we call it *conservative*.

Now we provide another point on the security-performance tradeoff for $\mathrm{T}_5$, which we call *aggressive*. We see that even though the provable security bounds decrease, the heuristic security (as the complexity of best attacks) remains the same under plausible conjectures. Both openings are depicted in Figure 3.

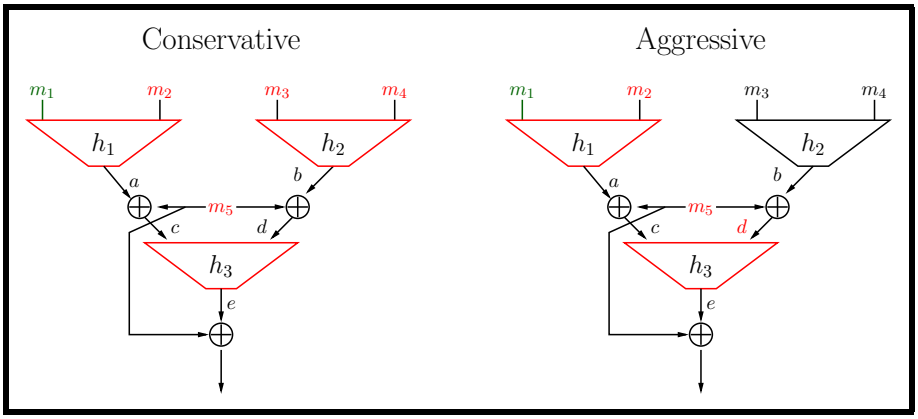


**Fig. 3.** Conservative and aggressive openings for $\mathrm{T}_5$. Green $m_1$ is opened. Red are opening elements (4 vs 3) and recomputed compression functions (3 vs 2).

Our (aggressive) local opening $\mathsf{Open}$ for $\mathrm{T}_5$ is defined as follows. Let $m = (m_1, m_2, m_3, m_4, m_5)$.

1. $\mathsf{Open}^{h_1,h_2,h_3}(1, m) = (m_2, m_5, h_2(m_3, m_4) \oplus m_5)$.
2. $\mathsf{Open}^{h_1,h_2,h_3}(2, m) = (m_1, m_5, h_2(m_3, m_4) \oplus m_5)$.
3. $\mathsf{Open}^{h_1,h_2,h_3}(3, m) = (m_4, m_5, h_1(m_1, m_2) \oplus m_5)$.
4. $\mathsf{Open}^{h_1,h_2,h_3}(4, m) = (m_3, m_5, h_1(m_1, m_2) \oplus m_5)$.
5. $\mathsf{Open}^{h_1,h_2,h_3}(5, m) = (m_1, m_2, h_2(m_3, m_4) \oplus m_5)$.

We first show that the above defined opening has a by-pass computation $\mathrm{T}'_5$ for index 1 (one can similarly show for the other indices) and hence it satisfies the correctness condition of an opening. For the sake of simplicity, we skip the hash oracles notation $h_1, h_2, h_3$. We define $\mathrm{T}'_5 : \{0,1\}^{4n} \to \{0,1\}^n$ based on the $2n$-to-$n$-bit compression functions $h_1, h_3$ as follows:

$$\mathrm{T}'_5(m'_1, m'_2, m'_3, m'_4) := h_3(h_1(m'_1, m'_2) \oplus m'_3, m'_4) \oplus m'_3$$

A straightforward calculation shows that $\mathrm{T}'_5(m_1, \mathsf{Open}(1, m)) = \mathrm{T}_5(m)$. Hence $\mathrm{T}'_5$ is a by-pass computation of $\mathrm{T}_5$ for the opening function defined above.

**Theorem 3.**

$$\mathbf{Adv}^{\mathit{full\text{-}local}}_{\mathrm{T}_5}(q) \le \mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{T}_5,\mathrm{T}'_5}(q) \le \frac{nq^3 + 9q^2}{2^n} \tag{6}$$

*and*

$$\mathbf{Adv}^{\mathit{local\text{-}local}}_{\mathrm{T}_5}(q) \le \mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{T}'_5}(q) \le \frac{q^4}{2^n} \tag{7}$$

We note that the relation between the collision advantage and the opening advantage is already described in (1). So it only remains to bound the cross-collision probability and the collision probability for a family of by-pass hash computations. The full formal proof of the cross-collision bound is somewhat similar to the collision security analysis of $\mathrm{T}_5$, and is provided in the Appendix (see App. C). But an informal proof intuition for a representative special case will be given in Section 5.1 below. However, the bound for collision advantage of the by-pass family is more or less straightforward and described afterwards.

### 5.1 Proof Intuition for Theorem 3 (Cross-Collision Bound)

Here, we give a proof sketch of the cross-collision advantage between $\mathrm{T}_5$ and $\mathrm{T}'_5$. We first note that in the case of $\mathrm{T}'_5$, all queries are consistent. Hence $q$ queries each to $h_1$ and $h_3$ can generate a maximum of $q^2$ $\mathrm{T}'_5$ evaluations. And, as we have already seen before, $q$ oracle queries can generate at most $nq$ evaluations of $\mathrm{T}_5$ if bad events $B_1$ and $B_2$ don't happen. Now, we approach as in the earlier theorem. We break the event that a collision between $\mathrm{T}_5$ and $\mathrm{T}'_5$ has occurred into three parts, depending upon which oracle was queried by the $i$-th query and give an upper bound to the collision probability. Then, we apply a union bound over the $q$ queries to give the intended result. Let $X_i$ be the event that none of bad events $B_1$, $B_2$ or $B_3$ happened. The three cases are as follows:

- The $i$-th query is $((m_1, m_2), a)$ to $h_1$. As in the earlier proof, a collision can happen in two ways, either this output $a$ induces a $\mathrm{T}_5$ tuple and a $\mathrm{T}'_5$ tuple which collide, or there is some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ such that the random answer $h_1(m_1, m_2) = a$ caused the collision with this $f$. The former case implies only the trivial collision, which we have ruled out. In the latter case, for a collision to happen, there are two subcases depending on whether $(*, f)$ comes from a previous evaluation of $\mathrm{T}_5$ or $\mathrm{T}'_5$ evaluation. If $(*, f)$ comes from a $\mathrm{T}_5$ evaluation, the random answer $a$ can combine with any of the queries $((c, d), e)$ of $h_3$ such that $f = a \oplus c \oplus e$. We note that there are a maximum of $nq$ prior $\mathrm{T}_5$ evaluations, and $q$ $h_3$ queries. The probability that $f = a \oplus c \oplus e$ is $1/2^n$ for each such query. Therefore,

$$\Pr[X_i] \leq nq \cdot q \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

  If $(*, f)$ comes from a $\mathrm{T}'_5$ evaluation, we note that there are at most $q^2$ such evaluations. There exist tuples $((*, b), ((c, d), e)) \in C_{23}(f)$, which can be at most $n$ in number, and the random answer $a = h_1(m_1, m_2)$ should be equal to $b \oplus c \oplus d$. This again gives

$$\Pr[X_i] \leq q^2 \cdot n \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

- The $i$-th query is $((m_3, m_4), b)$ to $h_2$. The only way this query can cause a collision is that there exists some previous value $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}'_5}$ such that the random answer creates a $\mathrm{T}_5$ output equal to $f$. This case is exactly the same as the first subcase of Case 1.
- The $i$-th query is $((c, d), e)$ to $h_3$. The case generated by this query is the same as that in the first case. If this query generates a $\mathrm{T}_5$ tuple and a $\mathrm{T}'_5$ tuple that collide, we again get only the trivial collision. If there is some previous $\mathrm{T}_5$ evaluation with which this query collides, then again, $q$ such queries can combine with $n$ evaluations of $\mathrm{T}_5$, and the collision probability for each combination is $1/2^n$, resulting in the same probability as in the first case. If this query collides with some previous $\mathrm{T}'_5$ evaluation, which are $nq$ in number, we note that there exist tuples $((*, a), (*, b)) \in C_{12}(c \oplus d)$ which can be at most $n$ in number. Again, we get the same probability.

Taking a union bound over the $q$ queries, we find that

$$\Pr\left[\, B_4 \cap \overline{B_1 \cap B_2 \cap B_3} \,\right] \leq q \cdot \max_i \Pr[X_i] \leq \frac{nq^3}{2^n}.$$

### 5.2 Proof of (7) (Collision Bound of Family of By-Pass Hash)

Now we prove the second part of the result (collision of by-pass hash family). Here we show the collision probability for $\mathrm{T}'_5$ (i.e., for the index 1). The proof for the other indices will be very similar and will have the same bound. As we take the maximum collision probability for all indices, the result will follow. Following a similar notation, let $h_1(m_1, m_2) = b, h_1(m'_1, m'_2) = b'$, $c = b \oplus m_5$ and $c' = b' \oplus m'_5$. So, the hash outputs are $\mathrm{T}'_5(m_1, m_2, m_5, d) = f = h_3(c, d) \oplus m_5$ and $\mathrm{T}'_5(m'_1, m'_2, m'_5, d') = f' = h_3(c', d) \oplus m'_5$. If $f = f'$ with $(m'_1, m'_2, m'_5, d') \neq (m_1, m_2, m_5, d)$ (i.e., collision happens) then we have

$$h_1(m'_1, m'_2) \oplus h_1(m'_1, m'_2) \oplus (c \oplus h_3(c, d)) \oplus (c' \oplus h_3(c', d')) = 0 \tag{8}$$

Let us write $h_3(x,y) \oplus x$ as $h'_3(x,y)$. It is obvious that $h'_3$ behaves exactly like a random function (independent with $h_1, h_2$). Thus, we have shown that collision problem of $\mathrm{T}'_5$ is reduced to finding $((m_1,m_2),(c,d)) \neq ((m'_1,m'_2),(c',d'))$ such that

$$h_1(m_1,m_2) \oplus h_1(m'_1,m'_2) \oplus h'_3(c,d) \oplus h'_3(c',d') = 0 \tag{9}$$

We call this problem 4-XOR′ (a variant of 4-XOR problem described in the following section). It is also not difficult to construct a collision pair from four pairs satisfying a 4-XOR′ relation. In other words, 4-XOR′ is equivalent to finding a collision of $\mathrm{T}'_5$. Now, by applying a union bound, the collision probability can be simply bounded above by $q^4/2^n$.

### 5.3 Reduction of Local Opening Security to 3-XOR/4-XOR Problem

*k-XOR problem.* Let $F_1, F_2, \ldots, F_k$ be $k$ oracles that output $n$-bit strings. Find $x_1, x_2, \ldots, x_k$ such that

$$F_1(x_1) \oplus F_2(x_2) \oplus \cdots F_k(x_k) = 0.$$

**Reduction for full-local.** When $k = 3$, the $k$-XOR problem has an information-theoretical security of $n/3$-bits, however the best algorithm requires more than $2^{n/2}/\sqrt{n}$ time (ignoring a $\log n$ factor) and queries [8, 23]. Bridging this gap would imply a solution to the long-standing 3-XOR problem, which would be a substantial breakthrough.

**Conjecture 1 (3-XOR hardness)** *For any algorithm $\mathcal{A}$ that makes less than $q < 2^{n/2}$ queries to $F_1, F_2, F_3$ and runs for time $q$, the probability that for randomly chosen $F_1, F_2, F_3$ it solves the 3-XOR problem is at most $n^2q^2/2^n$.*

Note that we have kept some margin on the power of $n$ in our conjecture. Until now we have attack with advantage about $nq^2/2^n$. Now we provide a heuristic reduction of full-local security to the 3-XOR problem. Let us look at the full-local security definition. To break it, we are required to find $(m_1, m_2, m_3, m_4, m_5, m'_1, m'_2, m'_5, d')$ such that

$$\mathrm{T}_5(m_1,m_2,m_3,m_4,m_5) = h_3(h_1(m'_1,m'_2) \oplus m'_5, d') \oplus m'_5$$

or equivalently

$$\mathrm{T}_5(m_1,m_2,m_3,m_4,m_5) = h_3(c',d') \oplus h_1(m'_1,m'_2) \oplus c'. \tag{10}$$

In our reduction we consider only algorithms which we call *valid-aware*. Those (1) make all queries to $h_1, h_2$ before $h_3$ and (2) for any query $(c,d)$ they make to $h_3$ are aware of all valid $\mathrm{T}_5$ executions that are created this way. Concretely, with any query $(c,d)$ they attach (a potentially empty) list of all pairs $h_1, h_2$ that are valid with $c, d$. This list cannot be long as we had argued for Theorem 1 where there cannot be more than $n$ of them. We stress that this is a natural assumption as every valid execution ever considered by an algorithm must be discovered in some way, and if all $h_3$ queries are made last, it only makes sense to make queries that yield valid executions. The speculative nature of this argument makes us claim that the reduction is only heuristic.

**Lemma 1.** *Given any valid-aware algorithm $\mathsf{A}$, which makes at most $q$ queries to oracles $h_1, h_2, h_3$, which solves (10) with probability $\epsilon$, there is an algorithm $\mathsf{A}'$ making at most $q < 2^{n/2}$ queries to oracles $F_1, F_2, F_3$ (with runtime almost the same as $\mathsf{A}$) that solves 3-XOR problem with probability at least $\epsilon/(4n)$.*

*Proof.* First we note that a solution to (10) must have $(c,d) \neq (c',d')$, i.e., $h_3$ gets a non-zero difference. Indeed, otherwise the output $e$ of $h_3$ and thus $m_5$ must both have a zero difference, which in turn implies that $a$ and $b$ have a zero difference, i.e., we have found a collision in $h_1$ or $h_2$ which we are ruling out (or we can extend our reduction with this outcome).

Now, $\mathsf{A}$ works as follows:

- It calls $\mathsf{A}$ to find a collision between $\mathrm{T}_5$ and $\mathrm{T}'_5$.
- When $\mathsf{A}$ queries $h_1(m_1,m_2)$ it is given $F_1(0||m_1||m_2)$.
- When $\mathsf{A}$ queries $h_2(m_3,m_4)$ it is given $F_1(1||m_3||m_4)$.
- When $\mathsf{A}$ queries $h_3(c,d)$ with list $L$ of valid tuples $(m_1,m_2,m_3,m_4)$ such that $h_1 \oplus h_2 = c \oplus d$. If the list is empty, $\mathsf{A}'$ returns $F_2(c||d) \oplus c$ to $\mathsf{A}$. Otherwise $\mathsf{A}'$ flips a coin:

- For tails $\mathsf{A}'$ selects a random entry of $L$ and returns $F_3(c||d) \oplus c \oplus F_1(0||m_1||m_2)$ to $\mathsf{A}$.
- For heads $\mathsf{A}'$ returns $F_2(c||d) \oplus c$ to $\mathsf{A}$.

Now suppose $\mathsf{A}$ finds a solution to (10). This implies that

$$h_1(m_1, m_2) \oplus h_3(c, d) \oplus c = h_1(m'_1, m'_2) \oplus h_3(c', d') \oplus c'.$$

Recall that $(c, d) \neq (c', d')$. Now note that:

- $(c, d)$ yields a valid execution of $\mathrm{T}_5$. Note that the validity was known at the time of query.
- With probability at least $1/(2n)$ the value for $h_3(c, d)$ is selected as $F_3(c||d) \oplus c \oplus F_1(0||m_1||m_2)$ (i.e., we had selected the same $m_1, m_2$ as in $h_1$ of the solution) since we have at most $n$ pairs $(h_1, h_2)$ with given difference $c \oplus d$.
- With probability $1/2$ the value for $h_3(c', d')$ is selected as $F_3(c'||d') \oplus c'$.
- The value for $h_1(m_1, m_2)$ is $F_1(0||m_1||m_2)$ and should cancel out due to our outcome for $h_3(c, d)$.

Therefore with total probability at least $1/(4n)$ the solution found by $\mathsf{A}$ is translated into

$$F_3(c||d) = F_1(0||m'_1||m'_2) \oplus F_2(c'||d')$$

which yields a solution for the 3-XOR problem.

Together with the hardness conjecture, we obtain the following proposition.

**Proposition 1** *Assuming the 3-XOR hardness and that the best algorithm that breaks full-local aggressive collision security is valid-aware, the adversary that runs in time $q$ finds a full-local collision with probability at most $4n^3q^2/2^n$.*

**Reduction for local-local.** The best known algorithm for solving 4-XOR problem runs in $O(n2^{n/3})$ time and $O(2^{n/3})$ queries [33]. So, we pose a similar conjecture for the 4-XOR problem.

**Conjecture 2 (4-XOR hardness)** *For any algorithm $\mathcal{A}$ that makes $q < 2^{n/3}$ queries to random oracles $F_1, F_2, F_3, F_4$, runs for time $q$, the probability that it solves the 4-XOR problem is at most $q^3/2^n$.*

Now we consider a simple variant of 4-XOR problem, called 4-XOR$'$, which uses two lists. Let $F, F'$ be oracles that output random $n$-bit strings. Find $x, y, z, w$ with $(x, y) \neq (z, w)$ such that

$$F(x) \oplus F(y) \oplus F'(z) \oplus F'(w) = 0.$$

**Lemma 2.** *Given any algorithm $\mathsf{A}'$ making at most $q$ queries to all its oracles which solves the 4-XOR$'$ problem with probability $\epsilon$ there is an algorithm $\mathsf{A}$ making at most $q$ queries to all its oracles (with run time almost same as $\mathsf{A}'$) which solves 4-XOR problem with probability at least $\epsilon/4$.*

*Proof.* The reduction from $\mathsf{A}'$ to $\mathsf{A}$ works as follows. We run $\mathsf{A}'$ and it makes two types of queries, namely to $F$ and to $F'$. For each query we choose $b$ randomly from $\{1, 2\}$. If it is an $F(x)$ query, then $\mathsf{A}$ returns $F_b(x)$. Similarly, if it is an $F'(z)$ query, $\mathsf{A}$ returns $F_{2+b}(z)$ to $\mathsf{A}'$. Finally, $\mathsf{A}'$ returns $(x, y, z, w)$ such that $F(x) \oplus F(y) \oplus F'(z) \oplus F'(w) = 0$. Now, $\mathsf{A}$ succeeds if $F(x) = F_b(x)$, $F(y) = F_{3-b}(y)$ and $F'(z) = F_{2+b'}(z)$, $F'(w) = F_{5-b'}(w)$. We note that the $b$ values are chosen randomly. As $F$ and $F'$ are independent random functions, the output to $\mathsf{A}'$ is independent of $b$. In other words, the view of $\mathsf{A}'$ remains independent with the $b$ values chosen by $\mathsf{A}$. Thus, $\mathsf{A}$ succeeds with probability $1/4$ given that $\mathsf{A}'$ succeeds. □

We have already seen that given a collision adversary $\mathsf{B}$ of $\mathrm{T}'_5$ we can construct an algorithm $\mathsf{A}'$ for solving the 4-XOR$'$ problem and hence we can construct an algorithm $\mathsf{A}$ solving the 4-XOR problem. Moreover the success probability of solving the 4-XOR is at least $\frac{1}{4} \cdot \mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{T}'_5}(\mathsf{B})$. This leads us to conclude with the following claim.

**Proposition 2** *Assuming the 4-XOR hardness and that the best algorithm that breaks local-local aggressive collision security is valid-aware, the adversary that runs in time $q$ finds a local-local collision with probability at most $4q^3/2^n$.*

## 6 Merkle-Damgård Variant

We can plug in our 5-to-1 compression function to the standard Merkle-Damgård (MD) mode to get a sequential hash $t$-to-1 function making only $3t/4$ calls to the underlying $2n$-to-$n$ compression functions $h_1, h_2$ and $h_3$, and inheriting the birthday security of $\mathrm{T}_5$. This function is depicted in Figure 4.

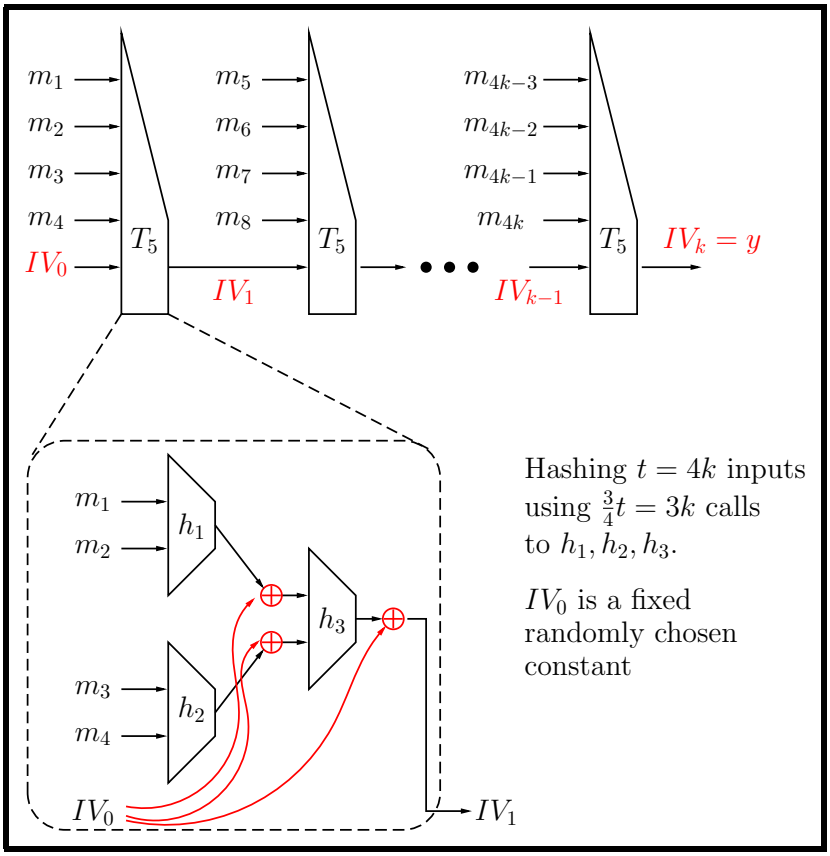


**Fig. 4.** Merkle-Damgård Hash based on $\mathrm{T}_5$.

Parallel Implementation. In addition to providing a 25% speedup when compared to the traditional MD mode applied to a $2n$-to-$n$-compression function $h$, another advantage of our new variant is that it is easily parallelizable for architectures that support parallel execution.

For example, if we have three execution units $P_1, P_2, P_3$, the unit $P_i$ can be responsible for all $h_i$ computations. This allows to compute a hash of $t = 4k$ message blocks using only $(k+2) = (t/4+2)$ parallel rounds of hashing, saving a factor (almost) 4 over the traditional MD mode. The execution unit $P_3$ will be "one-round behind" units $P_1$ and $P_2$ (so that in the first round only $P_1$ and $P_2$ hash $(m_1, m_2)$ and $(m_3, m_4)$, and in the last round only $P_3$ produces the final hash). Namely, when $P_3$ just completed the computation of the previous initial value $IV_j$, $P_1$ completed $a_j = h_1(m_{4j+1}, m_{4j+2})$, and $P_2$ completed $b_j = h_2(m_{4j+3}, m_{4j+4})$, in the next round $P_3$ will set the next initial value

$$IV_{j+1} = h_3(IV_j \oplus a, IV_j \oplus b) \oplus IV_j,$$

while $P_1$ and $P_2$ respectively compute $a_{j+1} = h_1(m_{4j+5}, m_{4j+5})$ and $b_{j+1} = h_2(m_{4j+6}, m_{4j+7})$.

Similarly, two execution units $P_1', P_2'$ can perform the same task in only $2k = t/2$ parallel rounds, saving a factor 2 over the traditional MD mode.

Larger Compression. Although our results are stated for $2n$-to-$n$ compression functions, we can also easily extend them to the $\lambda n$-to-$n$ case, by simply adding $(\lambda - 2)$ "dummy" inputs to each of $h_1, h_2, h_3$. For example, for $\lambda = 3$, this gives us an $8n$-to-$n$ analog of $\mathrm{T}_5$, using three calls

to some $3n$-to-$n$ hash function. Which gives a new MD mode for $t = 7k$ block messages, using only $3k = 3t/7$ compression calls. In contrast, a naive MD mode will use $t/2$ calls, saving a factor $(1 - (3t/7)/(t/2)) = 1/7 \approx 14.2\%$. Similar calculations can be done for larger $\lambda$.

# 7 Merkle Tree Variant

Our 2-level construction extends straightforwardly to a full-blown $t$-to-1 tree $H$. In Section 7.1 we briefly recall how to build Merkle Trees (MT) from smaller compression functions (such as $\mathrm{T}_5$). We then present our faster and shallower MT variant and its properties in Section 7.2.

## 7.1 General Merkle Trees and Their Security

The Merkle tree is a data structure to store long lists of $t$ $n$-bit elements, so that insertion, deletion, update, or proof of membership for an element need only $O(\log t)$ time and space. It is built on a $\lambda n$-to-$n$ compression function $H_\lambda$ (typically $\lambda = 2$ but other values are used too) and retains all its security properties regarding collision and preimage resistance from the compression function. The tree is defined recursively:

$$\mathsf{MT}_\lambda(\underbrace{m_1, \ldots, m_\lambda}_{m_{[1:\lambda]}}) = H_\lambda(m_1, \ldots, m_\lambda);$$

$$\mathsf{MT}_{\lambda t}(m_{[1:\lambda t]}) = H_\lambda(\mathsf{MT}_\lambda(m_{[1:\lambda]}), \ldots, \mathsf{MT}_\lambda(m_{[\lambda t-t+1:\lambda t]}))$$

where $t = \lambda^k \geq 2$, with the last hash called a root and $m_i$ called leafs.

As for the compression function, we define the terms full opening and local opening for the entire $\mathsf{MT}$, with the former being all $t$ elements and the latter for a leaf in a tree is a sequence of $\log_\lambda t$ local openings of an element in all compression functions on the path from the leaf to the root. In the simplest case $\lambda = 2$ an opening is one element per tree layer, whereas for wider compression functions it is $\lambda - 1$ or fewer (in case $H_\lambda$ has "aggressive" local opening). The full-local security for the tree is defined analogously to the compression function and corresponds to the case of a public tree to which an adversary makes a forged membership proof. The local-local security matches the case when the adversary provides two valid openings for an alleged tree root but the full tree is unknown. Both full-local and local-local security for the tree follows from their compression function counterparts. So overall we have the following parameters:

- Efficiency $E(t)$ as the number of compression function calls is $(t-1)/(\lambda-1)$.
- Depth $D(t)$ of the tree is $\log_\lambda t$.
- Update/insert/delete complexity as the number of compression function recomputations equals $D(t)$.
- The total length of *conservative* opening $L(t)$ is $(\lambda - 1)\log_\lambda t$.
- In case $H_\lambda$ might have a more compact (i.e., "aggressive") local opening of length $\ell \leq \lambda - 1$, then the total length of the resulting "aggressive" local opening for $\mathsf{MT}_\lambda$ becomes $L(t) = \ell \log_\lambda t$.
- Number of calls to $F$ needed to verify the opening: $V(t) = \log_\lambda t$.
- Collision, full-local, and local-local opening security the same as that of $H_\lambda$ (ideally $2^{n/2}$, if one uses a $2n$-to-$n$ hash function as last building block).
- Preimage security the same as that of $H_\lambda$ (ideally $2^n$, if one uses $2n$-to-$n$ hash function as last building block).

## 7.2 Faster and Shallower Merkle Tree based on $\mathrm{T}_5$

Our construction extends straightforwardly to a full-blown $t$-to-1 tree $H$ of any depth $k$, as depicted in Figure 5, with optimal $t = 5^k$. Notice, unlike our compression function $\mathrm{T}_5$ for $H$, which needed domain separation for functions $h_1, h_2, h_3$ (see Section 8.4), the final tree $H$ can reuse the same $h_1, h_2, h_3$ across all invocations, which follows from general security properties of standard Merkle trees.

The overall trade-offs of our construction are summarized in Table 1, but we expand on it below.

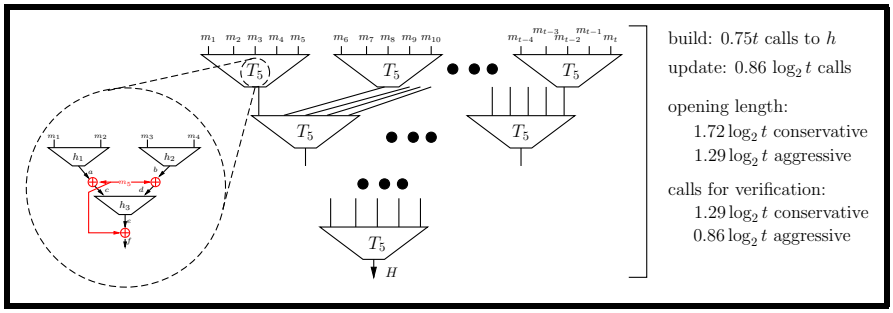


**Fig. 5.** Full-blown tree based on $\mathrm{T}_5$. Security is $2^{n/2}$ (tight) for conservative openings, $2^{n/3}$ provable and $2^{n/2}$ heuristic for full-local aggressive opening, $2^{n/4}$ provable and $2^{n/3}$ heuristic for local-local aggressive opening.

**Table 1.** We compare standard Merkle trees to two variants of Merkle trees with $\mathrm{T}_5$. The conservative variant requires opening four siblings in a local opening proof, compared to three siblings in the aggressive variant. The boxed formulas are conjectures based on the 3-XOR and 4-XOR problems. We have $2/\log_2 5 \approx 0.86$, $3/\log_2 5 \approx 1.29$, $4/\log_2 5 \approx 1.72$. Note that full-full collision resistance (CR) security is listed for completeness, this is the "traditional" collision resistance involving two distinct messages (and the entire corresponding Merkle trees).

| | Standard Merkle | Merkle with $T_5$ (conservative) | Merkle with $T_5$ (aggressive) |
|---|---|---|---|
| build calls/$t$ | 1 | 0.75 | 0.75 |
| update/$\log_2 t$ | 1 | 0.86 | 0.86 |
| verify/$\log_2 t$ | 1 | 1.29 | 0.86 |
| opening/$\log_2 t$ | 1 | 1.72 | 1.29 |
| full-full CR security | $n/2$ | $n/2$ | $n/2$ |
| full-local CR security | $n/2$ | $n/2$ | $n/3 \rightarrow \boxed{n/2}$ |
| local-local CR security | $n/2$ | $n/2$ | $n/4 \rightarrow \boxed{n/3}$ |

*Efficiency.* Let us summarize the performance of our tree construction:

- **Build**: In $H$ we compress every 5 inputs using our $5n$-to-$n$ compression function $\mathrm{T}_5$, therefore using $(t-1)/4$ calls to $\mathrm{T}_5$, which is equivalent to

$$E(t) = 0.75(t-1)$$

  calls to the $h_j$'s, thus giving us 25% improvement over the regular Merkle tree.
- **Depth/update**: The depth $D(t)$ of the tree, measured in the calls to the $h_j$'s, reduces from $\log_2 t$ to

$$D(t) = 2\log_5 t \approx 0.86 \log_2 t$$

  which is a 14% saving compared to standard Merkle trees.
- **Opening length (conservative)**: In the conservative opening we open all 4 siblings, thus $L(t)$ increases from $\log_2 t$ of standard Merkle trees to $4\log_5 t \approx 1.72\log_2 t$ (loss of 72%). It is still useful though when bandwidth is not critical which is sometimes the case.
- **Opening length (aggressive)**: In the aggressive opening we open 3 elements, thus $L(t)$ increases to only $3\log_5 t \approx 1.29\log_2 t$ (loss of 29%, which may be tolerated for some applications).
- **Verification time (conservative)**: the number of $h_i$ calls needed to verify the proof increases to $V(t) = 3\log_5 t \approx 1.29\log_2 t$.
- **Verification time (aggressive)**: the number of $h_i$ calls needed to verify the proof decreases to $V(t) = 2\log_5 t \approx 0.86\log_2 t$. This is very handy in applications when opening verification time is crucial (such as zero-knowledge membership proofs where proving time linearly depend on the circuit size needed for the opening verification). However we pay for this with security (see below).

*Security.* Since our construction applies the standard Merkle paradigm to the 5-to-1 compression function $\mathrm{T}_5$, the resulting hash function $H$ inherits the same global (or local opening) collision and preimage security as the compression function $\mathrm{T}_5$:

- **Full-local (aggressive) security**: provable security up to (taking $poly(n)$ factors aside) $2^{n/3}$ queries (Theorem 3), heuristic security up to $2^{n/2}$ running time (Proposition 1).
- **Local-local (aggressive) security**: provable security up to $2^{n/4}$ queries (Theorem 3), heuristic security up to $2^{n/3}$ running time (Proposition 2).
- **Both full-local and local-local (conservative) security**: provable security up to $2^{n/2}$ queries (Theorem 1).

Additionally, non-trivial preimage security holds up to $2^{2n/3}$ queries (and at optimal level $O(q/2^n)$, in the region of interest where $q \leq 2^{n/2}$).

*Applications.* Merkle trees are used in a number of protocols, but their exhaustive list is beyond the scope of this paper. To name just a few: anonymous cryptocurrencies and mixers [13, 25], interactive oracle proof (IOP) compilers [2,3], and post-quantum hash-based signatures [5]. Among them, cryptocurrencies and mixers are the examples of publicly controlled Merkle trees, i.e., where the full-local opening makes sense and where the aggressive opening with its improved verification time is appealing.

From all this diverse range of applications, the ones who benefit also from the conservative opening strategy are those for which the performance and depth are more important than the local opening size. Of course, basic hashing by itself is a very important example of such an application. The next examples are zero-knowledge proof systems where circuit depth is a major performance factor [34, 35]. Finally, a more speculative area where our construction could help is multiparty computation protocols applied to functionalities involving hashing, whose complexity depends on the circuit depth (e.g., variants of the original BGW [1] and GMW [11] protocols).

*Summary.* Our construction has clear advantage over the regular Merkle tree in build and update efficiency, but loses in the opening length. For the verification time and security we have a tradeoff: an aggressive opening needs fewer $h_i$ calls but only heuristic security argument of $2^{n/2}$ with provable security reaching $2^{n/3}$ only, whereas the conservative opening has the same security properties as in the regular tree but requires 30% more verification calls. Thus whether or not our construction outperforms the regular Merkle tree depends on the setting.

## 8 Matching Attacks and Warnings

In this section we discuss the tightness of our results (by providing matching attacks), as well as show that certain natural improvement or simplification ideas do not work.

### 8.1 Matching Collision Attack on $\mathrm{T}_5$

A simple birthday collision search with complexity $2^{n/2}$ applies to $\mathrm{T}_5$ in the full-full and full-local collision settings, matching Theorem 1 and Proposition 1.

### 8.2 Matching Preimage Attack on $\mathrm{T}_5$

We provide a brief description of a preimage attack with query complexity $2^{2n/3}$, which naturally follows from a bad event we aimed to exclude in the preimage security proof.

*Preimage attack.* Given challenge $f$, let us make $2^{2n/3}$ queries to both $h_1$ and $h_2$. There will be about $2^{4n/3-n} = 2^{n/3}$ collisions between $h_1$ and $h_2$. Let $\mathcal{A}$ denote the $2^{n/3}$ outputs of these collisions, so that $|\mathcal{A}| = 2^{n/3}$ and we know a valid preimage $(m_1, m_2, m_3, m_4)$ for the pair $(a, a)$ under $h_1$ and $h_2$, for each $a \in \mathcal{A}$.

Now we make $2^{2n/3}$ queries to $h_3$ in form $(c, c)$, and for each output $e$ check if there exists $a \in \mathcal{A}$ s.t.

$$a \oplus c = e \oplus f$$

Each such equation holds with probability $2^{-n}$, but we try $2^{n/3} \cdot 2^{2n/3} = 2^n$ different choices of $a$ and $c$, so expect one such equation to hold with constant probability. When the match happens, we can set $m_5 = a \oplus c = e \oplus f$, and the tuple $(m_1, m_2, m_3, m_4, m_5)$ will be a valid preimage of $f$, where $(m_1, m_2, m_3, m_4)$ is the colliding preimage of $(a, a)$ under $h_1$ and $h_2$.

*Analysis.* This attack requires $2^{2n/3}$ time and memory but can be computed memoryless using an unbalanced meet-in-the-middle attack [21] with time complexity about $2^{3n/4}$.

### 8.3 Matching Attack on Aggressive Opening Mode

Here we demonstrate a matching attack on the aggressive opening, which uses 3 (vs. 4) siblings to open a $\mathrm{T}_5$ commitment.

Concretely we construct a collision for $\mathrm{T}'_5$. Consider the equation

$$h_3(c, d) \oplus c \oplus h_1(m_1, m_2) = h_3(c', d') \oplus c' \oplus h_1(m'_1, m'_2)$$

By making $2^{n/4}$ queries to $h_1$ and $h_3$, we expect such colliding tuples will exist with constant probability, even if we insist on having $m_1 \neq m'_1$. Given such a tuple, let

$$h_3(c, d) \oplus c \oplus h_1(m_1, m_2) = f = h_3(c', d') \oplus c' \oplus h_1(m'_1, m'_2),$$

$m_5 = c \oplus h_1(m_1, m_2)$, and $m'_5 = c' \oplus h_1(m'_1, m'_2)$. Then it is easy to see that $(m_2, m_5, d)$ and $(m'_2, m'_5, d')$ are valid preimages of $f$ to $m_1 \neq m'_1$, respectively.

A similar attack works for any other permutation involving opening some $m_i$ using only 3 other values. Note however, that even though theoretically $2^{n/4}$ queries suffice, the best complexity of this attack is $2^{n/3}$ rather than $2^{n/4}$ (cf. the 4-XOR problem in Section 5).

### 8.4 Attack on $\mathrm{T}_5$ with Identical Compression Functions

Now we show that our requirement that all $h_i$ are distinct is crucial: without it, a subtle attack is possible. Let $\mathrm{T}^1_5 : \{0,1\}^{5n} \to \{0,1\}^n$ based on the *identical* $2n$ to $n$-bit compression functions $h_1, h_2, h_3 = h$ be defined as follows:

$$\mathrm{T}^1_5(m_1, m_2, m_3, m_4, m_5) := h(h(m_1, m_2) \oplus m_5, h(m_3, m_4) \oplus m_5) \oplus m_5$$

*Collision attack on* $\mathrm{T}^1_5$. Note that the only difference here with $\mathrm{T}_5$ is the fact that the internal compression functions in $\mathrm{T}^1_5$ are identical. In the $2^{n/4}$ query attack below, we will only use $h_1 = h_2$, though, so we could keep the index $h_3$ intact. However, a similar attack works if we only use $h_1 = h_3$ or $h_2 = h_3$, meaning that domain separation is needed on all three compression functions.

We make $2^{n/4}$ queries $(m_1, m_2)$ to $h$, and let $\mathcal{A}$ be the set of $((m_1, m_2), a)$ input-output pairs obtained. Also, make $2^{n/4}$ queries $(c, c)$ to $h_3 = h$, and let $\mathcal{C}$ be the set of $((c, c), e)$ input-output pairs obtained. Solve the equation

$$a \oplus (e \oplus c) = a' \oplus (e' + c') \tag{11}$$

for $a \neq a'$, where $((m_1, m_2), a), ((m'_1, m'_2), a') \in \mathcal{A}$, $((c, c), e), ((c', c'), e') \in \mathcal{C}$. We expect to find a solution with constant probability. Define

$$m_3 = m_1,\ m_4 = m_2,\ m_5 = a \oplus c,\ m'_3 = m'_1,\ m'_4 = m'_2,\ m'_5 = a' \oplus c'$$

Then, using (11),

$$\begin{aligned}
\mathrm{T}^1_5(m_1, m_2, m_3, m_4, m_5) &= \mathrm{T}^1_5(m_1, m_2, m_1, m_2, a \oplus c) \\
&= h(a \oplus (a \oplus c), a \oplus (a \oplus c)) \oplus (a \oplus c) \\
&= e \oplus a \oplus c = e' \oplus a' \oplus c' \\
&= h(a' \oplus (a' \oplus c'), a' \oplus (a' \oplus c')) \oplus (a' \oplus c') \\
&= \mathrm{T}^1_5(m'_1, m'_2, m'_1, m'_2, a' \oplus c') \\
&= \mathrm{T}^1_5(m'_1, m'_2, m'_3, m'_4, m'_5).
\end{aligned}$$

### 8.5 Attack on $\mathrm{T}_5$ Without the Final XOR

The next attack is on the modified $\mathrm{T}_5$ construction where the final XOR into the output of $h_3$ from the $\mathrm{T}_5$ construction is omitted; this attack provides more insight on the rationale behind the main $\mathrm{T}_5$ construction, by showing that the XOR of $m_5$ into the output of $h_3$ is essential.

Let $\mathrm{T}_5^2 : \{0,1\}^{5n} \to \{0,1\}^n$ based on the $2n$ to $n$-bit compression functions $h_1, h_2, h_3$ be defined as follows:

$$\mathrm{T}_5^2(m_1, m_2, m_3, m_4, m_5) := h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5)$$

*Collision attack on* $\mathrm{T}_5^2$. The attack follows from the generalized birthday attack which finds a quartet of distinct inputs $((m_1, m_2), (m_3, m_4), (m'_1, m'_2), (m'_3, m'_4))$ such that

$$h_1(m_1, m_2) \oplus h_1(m'_1, m'_2) \oplus h_2(m_3, m_4) \oplus h_2(m'_3, m'_4) = 0.$$

This can be done after $2^{n/4}$ queries to each of the hash function with constant probability. Let $h_1(m_1, m_2) \oplus h_1(m'_1, m'_2) = h_2(m_3, m_4) \oplus h_2(m'_3, m'_4) = \Delta$, then by setting $m'_5 = m_5 \oplus \Delta$ we derive

$$\begin{aligned}
\mathrm{T}_5^2(m_1, m_2, m_3, m_4, m_5) &= h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5) \\
&= h_3(h_1(m'_1, m'_2) \oplus \Delta \oplus m_5, h_2(m'_3, m'_4) \oplus \Delta \oplus m_5) \\
&= h_3(h_1(m'_1, m'_2) \oplus m'_5, h_2(m'_3, m'_4) \oplus m'_5) \\
&= \mathrm{T}_5^2(m'_1, m'_2, m'_3, m'_4, m'_5)
\end{aligned}$$

## 9 Generalizations and Future Work

Our construction is likely to be extended to wider compression functions. For example, a natural generalization of our construction can hash $T = 3 \cdot 2^k - 1$ inputs using only $E = 2 \cdot 2^k - 1 = (2T-1)/3$ evaluations, where standard $2n$-to-$n$ hash $h$ corresponds to $k = 0$, and our $\mathrm{T}_5$ corresponds to $k = 1$. As $k$ increases, $E(k)/T(k) \to 2/3$, which matches Stam's bound for building $Tn$-to-$n$ hash functions from $2n$-to-$n$ compression functions. Unfortunately, as $k$ grows, the local opening size also grows, so it is unclear if this overall hash saving is worthwhile for applications. We leave the security analysis of this, and other optimized hashing constructions to future work.

Finally, it remains to prove the reduction of the full-local aggressive security to the 3-XOR problem unconditionally, i.e., without restrictions on the adversary.

**Acknowledgments.** The authors would like to thank the organizers and participants of Dagstuhl Seminar 18021 "Symmetric Cryptography," which was held from January 7-12, 2018 at Schloss Dagstuhl – Leibniz Center for Informatics. The $\mathrm{T}_5$ construction was first presented by one of the authors of this paper during the seminar, and the fruitful discussions there provided an initial starting point for this paper. Yevgeniy Dodis was partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

## References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10. ACM (1988)
2. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed-Solomon interactive oracle proofs of proximity. In: ICALP. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
3. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: TCC (B2). Lecture Notes in Computer Science, vol. 9986, pp. 31–60 (2016)
4. Berman, P., Karpinski, M., Nekrich, Y.: Optimal trade-off for merkle tree traversal. Theor. Comput. Sci. **372**(1), 26–36 (2007)
5. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 368–397. Springer (2015)

6. Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 526–541. Springer (2005)
7. Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. J. Cryptol. **22**(3), 311–329 (2009)
8. Bouillaguet, C., Delaplace, C., Fouque, P.: Revisiting and improving algorithms for the 3xor problem. IACR Trans. Symmetric Cryptol. **2018**(1), 254–276 (2018)
9. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. **18**(2), 143–154 (1979)
10. Damgård, I.: A design principle for hash functions. In: CRYPTO. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer (1989)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC. pp. 218–229. ACM (1987)
12. Haitner, I., Ishai, Y., Omri, E., Shaltiel, R.: Parallel hashing via list recoverability. In: CRYPTO. Lecture Notes in Computer Science, vol. 9216, pp. 173–190. Springer (2015)
13. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification: Version 2019.0-beta-37 [overwinter+sapling]. Tech. rep., Zerocoin Electric Coin Company (2019), available at `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`
14. Jakobsson, M., Leighton, F.T., Micali, S., Szydlo, M.: Fractal merkle tree representation and traversal. In: CT-RSA. Lecture Notes in Computer Science, vol. 2612, pp. 314–326. Springer (2003)
15. Knudsen, L.R., Preneel, B.: Construction of secure and fast hash functions using nonbinary error-correcting codes. IEEE Trans. Inf. Theory **48**(9), 2524–2539 (2002)
16. McQuoid, I., Swope, T., Rosulek, M.: Characterizing collision and second-preimage resistance in linicrypt. In: Hofheinz, D., Rosen, A. (eds.) TCC. Lecture Notes in Computer Science, vol. 11891, pp. 451–470. Springer (2019)
17. Mennink, B., Preneel, B.: Hash functions based on three permutations: A generic security analysis. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 330–347. Springer (2012)
18. Mennink, B., Preneel, B.: Efficient parallelizable hashing using small non-compressing primitives. Int. J. Inf. Sec. **15**(3), 285–300 (2016)
19. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy. pp. 122–134. IEEE Computer Society (1980)
20. Merkle, R.C.: One way hash functions and DES. In: CRYPTO. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer (1989)
21. Morita, H., Ohta, K., Miyaguchi, S.: A switching closure test to analyze cryptosystems. In: CRYPTO. Lecture Notes in Computer Science, vol. 576, pp. 183–193. Springer (1991)
22. Nandi, M., Lee, W., Sakurai, K., Lee, S.: Security analysis of a 2/3-rate double length compression function in the black-box model. In: FSE. Lecture Notes in Computer Science, vol. 3557, pp. 243–254. Springer (2005)
23. Nikolic, I., Sasaki, Y.: Refinements of the k-tree algorithm for the generalized birthday problem. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 9453, pp. 683–703. Springer (2015)
24. Patrascu, M., Thorup, M.: The power of simple tabulation hashing. J. ACM **59**(3), 14:1–14:50 (2012)
25. Pertsev, A., Semenov, R., Storm, R.: Tornado cash privacy solution version 1.4 (2019), `https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf`
26. Peyrin, T., Gilbert, H., Muller, F., Robshaw, M.J.B.: Combining compression functions and block cipher-based hash functions. In: ASIACRYPT. vol. 4284, pp. 315–331. Springer (2006)
27. Rogaway, P., Steinberger, J.P.: Security/efficiency tradeoffs for permutation-based hashing. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 4965, pp. 220–236. Springer (2008)
28. Seurin, Y., Peyrin, T.: Security analysis of constructions combining FIL random oracles. In: FSE. vol. 4593, pp. 119–136. Springer (2007)
29. Stam, M.: Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In: CRYPTO. Lecture Notes in Computer Science, vol. 5157, pp. 397–412. Springer (2008)
30. Steinberger, J.P.: Stam's collision resistance conjecture. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 597–615. Springer (2010)
31. Steinberger, J.P., Sun, X., Yang, Z.: Stam's conjecture and threshold phenomena in collision resistance. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 384–405. Springer (2012)
32. Szydlo, M.: Merkle tree traversal in log space and time. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 3027, pp. 541–554. Springer (2004)
33. Wagner, D.A.: A generalized birthday problem. In: CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 288–303. Springer (2002)
34. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society (2018)
35. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: CRYPTO. Lecture Notes in Computer Science, vol. 11694, pp. 733–764. Springer (2019)

## A Formal Proof of Collision Resistance of $T_5$

Consider any collision finding adversary $\mathsf{A}$ making a total of $q$ queries to $h_1, h_2, h_3$. Without loss of generality, we assume that $\mathsf{A}$ does not duplicate any of its queries, and is deterministic. Thus, all the probabilities are computed over the randomness of the ideal hash functions $h_1, h_2, h_3$.

For any index $i = 0 \dots q$, the run of $\mathsf{A}$ defines the following random variables:

- $Q_1^i, Q_2^i, Q_3^i$: set of input-output pairs of $h_1, h_2, h_3$ known to $\mathsf{A}$ after its $i$-th query. In particular, $((m_1, m_2), a) \in Q_1^i$ if and only if one of the first $i$ queries of $\mathsf{A}$ was a query $(m_1, m_2)$ to $h_1$, and $a$ was the answer to this query. Same for $h_2$ and $h_3$.
- $\mathbf{Eval}^i$: the set of input-output pairs for the entire construction $T_5$ known to $\mathsf{A}$ after its $i$-th query. Formally, we look at all triples of *consistent* queries $((m_1, m_2), a) \in Q_1^i$, $((m_3, m_4), b) \in Q_2^i$ and $((c, d), e) \in Q_3^i$, and for each such triple we add the resulting input-output pair $(M, f)$ defined by this triple to the set $\mathbf{Eval}^i$.
- For any $z \in \{0,1\}^n$,

$$C_{12}^i(z) = \{ \ ( \ ((m_1, m_2), a) \in Q_1^i, ((m_3, m_4), b) \in Q_2^i \ ) \ \mid \ a \oplus b = z \ \} \tag{12}$$

  Jumping ahead, these sets will determine how many fresh consistent triples of queries can be obtained by $\mathsf{A}$, by making a new query $(c, d)$ to $h_3$.
- For any $f \in \{0,1\}^n$,

$$C_{13}^i(f) = \{ \ ( \ ((m_1, m_2), a) \in Q_1^i, ((c, d), e) \in Q_3^i \ ) \ \mid \ a \oplus e = f \oplus c \ \}$$
$$C_{23}^i(f) = \{ \ ( \ ((m_3, m_4), b) \in Q_2^i, ((c, d), e) \in Q_3^i \ ) \ \mid \ b \oplus e = f \oplus d \ \}$$

  Jumping ahead, for any output value $f$ of $T_5$ (presumably obtained by some consistent triple of queries), the set $C_{13}^i(f)$ (resp. $C_{23}^i(f)$) will determine how many tuples of queries to $h_1$ (resp. $h_2$) and $h_3$ have a chance to cause a collision with $f$, before a new query to $h_2$ (resp. $h_1$) is made.

We let $Q_1 = Q_1^q$, $Q_2 = Q_2^q$, $Q_3 = Q_3^q$, $C_{12}(z) = C_{12}^q(z)$, $C_{13}(f) = C_{13}^q(f)$, $C_{23}(f) = C_{23}^q(f)$ and $\mathbf{Eval} = \mathbf{Eval}^q$ be the final values of the corresponding sets at the end of execution. We can now define the following "bad events" at the end of $\mathsf{A}$'s execution:

- $B_1$: collision in either $Q_1$, $Q_2$, or "shifted" $Q_3$. Formally:
  - $\exists((m_1, m_2), a), ((m_1', m_2'), a) \in Q_1$ with $(m_1, m_2) \neq (m_1', m_2')$, or
  - $\exists((m_3, m_4), b), ((m_3', m_4'), b) \in Q_2$ with $(m_3, m_4) \neq (m_3', m_4')$, or
  - $\exists((c, d), e), ((c', d'), e') \in Q_3$ with $(c, d) \neq (c', d')$ and either $e \oplus c = e' \oplus c'$ or $e \oplus d = e' \oplus d'$.
- $B_2$: $\max_z |C_{12}(z)| \geq n$, or $\max_f |C_{13}(f)| \geq n$, or $\max_f |C_{23}(f)| \geq n$.
- $B_3$: $|\mathbf{Eval}| \geq qn$; i.e., number of valid evaluations of $T_5$ is "too large".
- $B_4$: collision on $T_5$ found; i.e., $(M, f), (M', f) \in \mathbf{Eval}$ with $M \neq M'$.

Our goal is to upper bound the probability of $B_4$. For that, we split this probability into four disjoint events, as follows:

$$\Pr[B_4] \leq \Pr[B_1] + \Pr\left[B_2 \cap \overline{B_1}\right] + \Pr\left[B_3 \cap \overline{B_1 \cap B_2}\right] + \Pr\left[B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\right]$$

In the following propositions (proven in subsequent subsections), we bound each of these 4 terms separately.

**Proposition 3** $\Pr[B_1] \leq \frac{2q^2}{2^n}$

**Proposition 4** $\Pr\left[B_2 \cap \overline{B_1}\right] \leq 3 \cdot \left(\frac{2q^2}{2^n}\right)^n$

**Proposition 5** $\Pr\left[B_3 \cap \overline{B_1 \cap B_2}\right] \leq \frac{2q^2}{n2^n}$

**Proposition 6** $\Pr\left[B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\right] \leq \frac{n^2q^2}{2^n}$

Combining the results of Propositions 3,4,5,6, we get our final bound in (3):

$$\Pr[B_4] \leq \frac{2q^2}{2^n} + 3 \cdot \left(\frac{2q^2}{2^n}\right)^n + \frac{2q^2}{n2^n} + \frac{n^2q^2}{2^n} \leq \frac{(n^2+10)q^2}{2^n},$$

assuming that $\frac{2q^2}{2^n} \leq 1$. Hence, it suffices to prove Propositions 3,4,5,6.

**Bounding Simple Collisions (Proof of Proposition 3)** For completeness, we repeat the claimed bound:

$$\Pr\left[\, B_1 \,\right] \leq \frac{2q^2}{2^n}$$

Each of the 4 events in $B_1$ (collision in $h_1$, $h_2$, and two types of collisions in $h_3$) correspond to standard birthday bounds in ideal hash functions $h_1(m_1, m_2)$, $h_2(m_3, m_4)$, $h_3(c, d) \oplus c$ and $h_3(c, d) \oplus d$. Each such bound is $q^2/2^{n+1}$, so the final bound is this birthday bound times 4.

**Bounding Multi-Collisions (Proof of Proposition 4)** For completeness, let us repeat the claimed bound:

$$\Pr\left[\, B_2 \cap \overline{B_1} \,\right] \leq 3 \cdot \left(\frac{2q^2}{2^n}\right)^n$$

Intuitively, each of the sets $C_{jk}(\cdot)$ corresponds to the fact that "many" pairs of queries to $h_j$ and $h_k$ produced outputs whose XOR is equal to a fixed value. Clearly, this probability gets lower and lower as we increase the value of "many." The calculation below shows that setting it to $n$ drives this probability below the "birthday" bound we are seeking. We use the standard union bound several times to upper bound the desired probability as follows:

$$\begin{aligned}
\Pr\left[\, B_2 \cap \overline{B_1} \,\right] \leq\ & \Pr\left[ (\max_z |C_{12}(z)| \geq n) \cap \overline{B_1} \right] + \\
& \Pr\left[ (\max_f |C_{13}(f)| \geq n) \cap \overline{B_1} \right] + \\
& \Pr\left[ (\max_f |C_{23}(f)| \geq n) \cap \overline{B_1} \right] \\
\leq\ & 2^n \cdot \max_z \Pr\left[\, (|C_{12}(z)| \geq n) \cap \overline{B_1} \,\right] + \\
& 2^n \cdot \max_f \Pr\left[\, |(C_{13}(f)| \geq n) \cap \overline{B_1} \,\right] + \\
& 2^n \cdot \max_f \Pr\left[\, |(C_{23}(f)| \geq n) \cap \overline{B_1} \,\right]
\end{aligned}$$

We upper bound each of the three probabilities below (see (15), (16), and (17)) by $(q^2/2^n)^n$, which completes the proof, since then

$$\Pr\left[\, B_2 \cap \overline{B_1} \,\right] \leq\ 3 \cdot 2^n \cdot \left(\frac{q^2}{2^n}\right)^n = 3 \cdot \left(\frac{2q^2}{2^n}\right)^n$$

Let us start with $\Pr\left[\, (|C_{12}(z)| \geq n) \cap \overline{B_1} \,\right]$ for any fixed $z$. First, recall that $C_{12}(z) = \{\ (\ (*, a) \in Q_1, (*, b) \in Q_2\ )\ \mid\ a \oplus b = z\ \}$. Define $\Delta^i_{12}(z) = |C^i_{12}(z) \backslash C^{i-1}_{12}(z)|$, which is the number of tuples added to $C^i_{12}(z)$ because a new evaluation in query $i$ was made. Then $|C_{12}(z)| = \sum_{i=1}^{q} \Delta^i_{12}(z)$. Now, in order for the sum of $q$ non-negative integers to be at least $n$, either one of the terms is strictly greater than 1, or at least $n$ of the terms are non-zero:

$$\begin{aligned}
\Pr\left[\, (|C_{12}(z)| \geq n) \cap \overline{B_1} \,\right] =\ & \Pr\left[ \left( \sum_{i=1}^{q} \Delta^i_{12}(z) \geq n \right) \cap \overline{B_1} \right] \\
\leq\ & \Pr\left[\, (\exists i \text{ s.t. } \Delta^i_{12}(z) > 1) \cap \overline{B_1} \,\right] + \\
& \Pr\left[\, \exists \geq n \text{ indices } i \text{ s.t. } \Delta^i_{12}(z) > 0 \,\right] \\
\leq\ & q \cdot \max_i (\Pr\left[\, (\Delta^i_{12}(z) > 1) \cap \overline{B_1} \,\right]) + \qquad (13) \\
& q^n \cdot \max_{i_1 \ldots i_n} (\Pr\left[\, \forall i \in \{i_1 \ldots i_n\},\ \Delta^i_{12}(z) > 0 \,\right]) \qquad (14)
\end{aligned}$$

Now, let us look at the $i$-th query of $\mathsf{A}$. Clearly, a query to $h_3$ does not increase $|C^i_{12}(z)|$, so by symmetry let us assume that query $i$ was a query $(m_1, m_2)$ to $h_1$. Assume the random answer is

$h_1(m_1, m_2) = a$. This new query $((m_1, m_2), a)$ will "match up" with an old query $((m_3, m_4), b) \in Q_2^i$ only if $a = b \oplus z$. Moreover, the only way it will also match up with another such query $((m_3', m_4'), b') \in Q_2^i$ is if $a = b \oplus z = b' \oplus z$. But this means that $b = b'$ is a collision in $Q_2$, triggering the bad event $B_1$. Hence, the probability in (13) is 0:

$$\Pr\left[\, (\Delta_{12}^i(z) > 1) \cap \overline{B_1} \,\right] = 0$$

Moreover, since $|Q_2^i| \leq q$, there are at most $q$ values $\{b \oplus z \mid (*, b) \in Q_2^i\}$. And since the answer $a$ to $h_1(m_1, m_2)$ is truly random, the chance it will be equal one of the values in this set is at most $q/2^n$, which means

$$\Pr\left[\, \Delta_{12}^i(z) > 0 \,\right] \leq \frac{q}{2^n}$$

Coming back to (14), we see that the $n$ events corresponding to distinct indices $i_1 \ldots i_n$ are independent, meaning

$$\Pr\left[\, \forall i \in \{i_1 \ldots i_n\},\ \Delta_{12}^i(z) > 0 \,\right] \leq \left(\frac{q}{2^n}\right)^n$$

Hence, for any $z$,

$$\Pr\left[\, (|C_{12}(z)| \geq n) \cap \overline{B_1} \,\right] \leq\ q \cdot 0 + q^n \cdot \left(\frac{q}{2^n}\right)^n \leq \left(\frac{q^2}{2^n}\right)^n \tag{15}$$

Completely analogously, we can argue that for any $f$,

$$\Pr\left[\, (|C_{13}(f)| \geq n) \cap \overline{B_1} \,\right] \leq\ \left(\frac{q^2}{2^n}\right)^n \tag{16}$$

$$\Pr\left[\, (|C_{23}(f)| \geq n) \cap \overline{B_1} \,\right] \leq\ \left(\frac{q^2}{2^n}\right)^n \tag{17}$$

For example, the only difference between the argument for $C_{13}(f)$ and $C_{12}(z)$ is that the key equation becomes $a \oplus e = f \oplus c$ (instead of $a = b \oplus z$). When the $i$-th query is to $h_3(c, d)$, the answer $e$ matching up with two values $a$ and $a'$ will imply a collision in $Q_1$, as before (since $a = a' = e \oplus f \oplus c$). However, when the $i$-th query is to $h_1(m_1, m_2)$, the answer $a$ matching up with two values $e$ and $e'$ will imply a collision in "shifted" $Q_3$, because now we have $a \oplus f = e \oplus c = e' \oplus c'$. And this is why we defined this as the "$h_3$-collision" when defining the bad event $B_1$. Similarly, the argument for $C_{23}(f)$ will result in $b \oplus f = e \oplus d = e' \oplus d'$.

This completes the proof.

**Bounding Evaluation Set of $\mathbf{T_5}$ (Proof of Proposition 5)** For completeness, we repeat the claimed bound:

$$\Pr\left[\, B_3 \cap \overline{B_1 \cap B_2} \,\right] \leq \frac{2q^2}{n2^n}$$

Recall, $B_3$ states that $|\mathbf{Eval}| \geq qn$, meaning $\mathsf{A}$ completed at least $qn$ evaluations of $\mathrm{T}_5$ in its $q$ queries. Let us denote by $q_j$, for $j = 1, 2, 3$, the (random variable equal to the) number of calls to $h_j$. And by $\mathbf{Eval}_j$ the number of new evaluations of $\mathrm{T}_5$ resulting from the $q_j$ calls to $h_j$; namely, it was the call to $h_j$ that finalized the consistent triplet which defined this evaluation of $\mathrm{T}_5$. Clearly, $q_1 + q_2 + q_3 = q$ and $|\mathbf{Eval}| = |\mathbf{Eval}_1| + |\mathbf{Eval}_2| + |\mathbf{Eval}_3|$, which means that

$$\Pr\left[\, (|\mathbf{Eval}| \geq qn) \cap \overline{B_1 \cap B_2} \,\right] \leq\ \sum_{j=1}^{3} \Pr\left[\, (|\mathbf{Eval}_j| \geq q_j n) \cap \overline{B_1 \cap B_2} \,\right] \tag{18}$$

Namely, to complete more than $qn$ total evaluations, at least one of the functions $h_1, h_2, h_3$ should have been "responsible" for at least $q_j n$ such evaluations. To complete the proof of the proposition, it thus suffices to show that

$$\Pr\left[\, (|\mathbf{Eval}_1| \geq q_1 n) \cap \overline{B_1 \cap B_2} \,\right] \leq \Pr\left[\ |\mathbf{Eval}_1| \geq q_1 n\ \right] \leq \frac{q^2}{n2^n} \tag{19}$$

$$\Pr\left[\, (|\mathbf{Eval}_2| \geq q_2 n) \cap \overline{B_1 \cap B_2} \,\right] \leq \Pr\left[\ |\mathbf{Eval}_2| \geq q_2 n\ \right] \leq \frac{q^2}{n2^n} \tag{20}$$

$$\Pr\left[\, (|\mathbf{Eval}_3| \geq q_3 n) \cap \overline{B_1 \cap B_2} \,\right] = 0 \tag{21}$$

We start with the easier (21), showing that queries to $h_3$ simply cannot cause so many new evaluations without triggering the bad event $B_2$. Recall, the set $\mathbf{Eval}_3$ can grow from 0 to its final value in $q_3$ queries to $h_3$ (somehow dispersed among $\mathsf{A}$'s $q$ total queries). And to achieve the final value $q_3 n$, at least one such query to $h_3$ must define at least $n$ new evaluations of $\mathrm{T}_5$. Namely, for some query $1 \le i \le q$, this query $(c, d)$ was made to $h_3$, and $|\mathbf{Eval}^i \backslash \mathbf{Eval}^{i-1}| \ge n$. But the only way this query $(c, d)$ could "match up" to two prior queries $(*, a) \in Q_1^i$ and $(*, b) \in Q_2^i$ is if $m_5 = a \oplus c = b \oplus d$. This is equivalent to $a \oplus b = c \oplus d$, which is in turn equivalent to saying that the tuple of queries $((*, a), (*, b)) \in C_{12}(c \oplus d)$. So in order for such query $(c, d)$ to match up with at least $n$ tuples $((*, a), (*, b)) \in Q_1^i \times Q_2^i$, it must be the case that $|C_{12}(c \oplus d)| \ge n$. But this immediately triggers the bad event $B_2$, meaning that the probability in question is 0, completing the proof of (21).

Thus, it remains to show (19), as (20) is symmetric. Assume $i_1, \ldots, i_{q_1} \in \{1...q\}$ are the indices of the $q_1$ queries to $h_1$, and let $\Delta^k = |\mathbf{Eval}^{i_k} \backslash \mathbf{Eval}^{(i_k-1)}|$ be the number of new evaluations of $\mathrm{T}_5$ during the $k$-th query to $h_1$, where $1 \le k \le q_1$. Then $|\mathbf{Eval}_1| = \Delta_1 + \ldots + \Delta_{q_1}$. Moreover, we claim that the expected value of $\Delta_k$ is at most $q^2/2^n$: $\mathbb{E}[\Delta_k] \le q^2/2^n$. To see this, $|Q_2|, |Q_3| \le q$, and each pair $((*, b), ((c, d), *)) \in Q_2 \times Q_3$ would only match the value $a$ returned by $h_1$ if $a = b \oplus c \oplus d$. The probability of this event is $1/2^n$, and the maximum number of such pairs is at most $|Q_2 \times Q_3| \le q^2$, giving $\mathbb{E}[\Delta_k] \le q^2/2^n$. But this means

$$\mathbb{E}[\ |\mathbf{Eval}_1|\ ] = \sum_{k=1}^{q_1} \mathbb{E}[\Delta_k] \le \frac{q_1 q^2}{2^n}$$

By Markov's inequality,

$$\Pr[\ |\mathbf{Eval}_1| \ge q_1 n\ ] \le \Pr\left[\ |\mathbf{Eval}_1| \ge \mathbb{E}[|\mathbf{Eval}_1|] \cdot \frac{n2^n}{q^2}\ \right] \le \frac{q^2}{n2^n}$$

establishing (19), and completing the proof of proposition.

**Final Collision Probability (Proof of Proposition 6)** For completeness, we repeat the claimed bound:

$$\Pr\left[\ B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\ \right] \le \frac{n^2 q^2}{2^n}$$

Let $X_i$ be the event that none of bad events $B_1$, $B_2$ or $B_3$ happened, and that the $i$-th query of $\mathsf{A}$ generates a fresh collision to $\mathrm{T}_5$, where $1 \le i \le q$. We upper bound $\Pr[X_i]$ separately depending on whether this was the call to $h_1$, $h_2$ or $h_3$. In each of these case we show $\Pr[X_i] \le n^2 q/2^n$, meaning that this bound is correct irrespective of the identity of the $i$-th query.

*Cases 1-2: query to $h_1$ or $h_2$.* Since these cases are symmetric, assume the $i$-th query is the query $(m_1, m_2)$ to $h_1$. In order for this query to cause a collision to $\mathrm{T}_5$, one of two things must happen. First, it could be that this very query $(m_1, m_2)$ defined a value $a$ which induced two distinct consistent triples

$$\begin{aligned} &(\ ((m_1, m_2), a), ((m_3, m_4), b\ ), ((c\ , d\ ), e\ )\ ) \\ \neq &(\ ((m_1, m_2), a), ((m_3', m_4'), b'), ((c', d'), e')\ ) \end{aligned}$$

which collided. But this means that $f = a \oplus (e \oplus c) = a \oplus (e' \oplus c')$, or $e \oplus c = e' \oplus c'$. But since we assumed $B_1$ failed, there are no collisions in "$c$-shifted" $h_3$, which means $(c, d) = (c', d')$. But then, since both tuples are consistent,

$$b = a \oplus c \oplus d = a \oplus c' \oplus d' = b'$$

which means that there is a collision on $h_2$, which again violates $B_1$.

Hence, in order for the query $(m_1, m_2)$ to cause a collision in $\mathrm{T}_5$, there must be some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ s.t. the random answer $h_1(m_1, m_2) = a$ caused the collision with this $f$. In turn, this means that there are two queries $(*, b) \in Q_2^{i-1}$ and $((c, d), e) \in Q_3^{i-1}$ such that the values $m_5 = b \oplus d = e \oplus f$ match, which means that $b \oplus e = f \oplus d$, and, hence the tuple

$((*, b), ((c, d), e) \in C_{23}(f)$. Moreover, for the value $a$ to actually match this tuple, we must have $a = b \oplus c \oplus d$, which happens with probability $1/2^n$.

To summarize, in order for the $i$-th query to $h_1$ to cause the collision, the following events must happen:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}$.
- There exists a tuple $((*, b), ((c, d), e)) \in C_{23}(f)$.
- The random answer $a = h_1(m_1, m_2)$ should be equal to $b \oplus c \oplus d$.

Moreover, since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}| \leq |\mathbf{Eval}| < nq$ and $|C_{23}(f)| \leq n$ (as otherwise $X_i$ does not happen anyway). This means that

$$\Pr[X_i] \leq nq \cdot n \cdot \frac{1}{2^n} = \frac{n^2 q}{2^n}$$

*Case 3: query to $h_3$.* Assume the $i$-th query is the query $(c, d)$ to $h_3$. In order for this query to cause a collision to $\mathrm{T}_5$, one of two things must happen. First, it could be that this very query $(c, d)$ defined a value $e$ which induced two distinct consistent triples

$$\begin{aligned}&(\ ((m_1, m_2), a\ ), ((m_3, m_4), b\ ), ((c, d), e)\ )\\ \neq&(\ ((m'_1, m'_2), a'), ((m'_3, m'_4), b'), ((c, d), e)\ )\end{aligned}$$

which collided. But this means that $f = a \oplus (e \oplus c) = a' \oplus (e \oplus c)$, or $a = a'$, and, similarly, $f = b \oplus (e \oplus d) = b' \oplus (e \oplus d)$, or $b = b'$. But since we assumed there are no collisions on $h_1$ and $h_2$, this means that $(m_1, m_2) = (m'_1, m'_2)$, $(m_3, m_4) = (m'_3, m'_4)$, and the two consistent triples above are identical.

Hence, in order for the query $(c, d)$ to cause a collision in $\mathrm{T}_5$, there must be some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ s.t. the random answer $h_3(c, d) = e$ caused the collision with this $f$. In turn, this means that there are two queries $(*, a) \in Q_1^{i-1}$ and $(*, b) \in Q_2^{i-1}$ such that the values $m_5 = a \oplus c = b \oplus d$ match, which means that $a \oplus b = c \oplus d$, and, hence the tuple $((*, a), (*, b) \in C_{12}(c \oplus d)$. Moreover, for the value $e$ to actually match this tuple, we must have $e = f \oplus (a \oplus c)$ (which is then the same as $f \oplus (b \oplus d)$), which happens with probability $1/2^n$.

To summarize, in order for the $i$-th query to $h_3$ to cause the collision, the following events must happen:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}$.
- There exists a tuple $((*, a), (*, b)) \in C_{12}(c \oplus d)$.
- The random answer $e = h_3(c, d)$ should be equal to $f \oplus a \oplus c$.

Moreover, since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}| \leq |\mathbf{Eval}| < nq$ and $|C_{12}(c \oplus d)| \leq n$ (as otherwise $X_i$ does not happen anyway). This means that

$$\Pr[X_i] \leq nq \cdot n \cdot \frac{1}{2^n} = \frac{n^2 q}{2^n}$$

Having established that $\Pr[X_i] \leq n^2 q/2^n$ for all $i$ (irrespective of what the $i$-th query was), we simply take the union bound over $i \in \{1 \dots q\}$ to obtain

$$\Pr\left[\, B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\, \right] \leq q \cdot \max_i \Pr[X_i] \leq \frac{n^2 q^2}{2^n}$$

## B Formal Proof of Preimage Resistance of $\mathrm{T}_5$

Consider any (w.l.o.g., deterministic and not repeating queries) collision finding adversary $\mathsf{A}$ making a total of $q$ queries to $h_1, h_2, h_3$. Our proof will follow the same rough strategy as the collision security proof in App. A, but with some differences.

First, we are given a specific value $f = \mathrm{T}_5(M)$ for a random (unknown) $M$, and now need to invert this $f$. Since each evaluation of $\mathrm{T}_5$ happened on a truly random input, and the value of $f$

is truly random, it is statistically unlikely that A will make any of the 3 queries used to evaluate $T_5(M)$: the probability of this event is at most $O(q/2^n)$. Hence, we will assume that $f$ is truly random, and A starts with fresh (unqueried) hash functions $h_1, h_2, h_3$.

Next, we will define the same sets $Q_1^i, Q_2^i, Q_3^i, \mathbf{Eval}^i, C_{12}^i(z), C_{13}^i(f), C_{23}^i(f)$ as in the proof of App. A, and will omit superscripts at the end of the execution (i.e., when $i = q$). Note that we will only care about the sets $C_{13}(f), C_{23}(f)$ for our "challenge" value $f$ rather than any $f$. More significantly, our "bad events" at the end of A's execution will be defined differently. Intuitively, we will no longer care about individual collisions in various hash functions (old event $B_1$), as such collisions will not help the adversary to find an inversion. Instead, we will only care about certain sets becoming too large, as this could help A invert $f$ quicker than expected:

- $B_5$: for a particular constant $\alpha > 0$ defined later,[12]

$$\max(\ \max_z |C_{12}(z)|,\ |C_{13}(f)|,\ |C_{23}(f)|\ ) > \frac{2n}{\alpha} + \frac{2q^2}{2^n}$$

- $B_6$: $(*, f) \in \mathbf{Eval}$; i.e., the adversary found a preimage of $f$.

Notice, event $B_5$ is similar to the "old" event $B_2$, except we increased the "multi-collision threshold" from $n$ to $K = O(n) + 2q^2/2^n$, and the event $B_6$ is the one we care about. We show the following:

**Proposition 7** *For some constant $\alpha$,* $\Pr[\, B_5 \,] \le \frac{3}{2^n}$

**Proposition 8** $\Pr\left[\, B_6 \cap \overline{B_5} \,\right] \le \frac{2q^3}{2^{2n}} + \frac{2qn}{\alpha 2^n}$

Combining the results of Propositions 7,8 (proven below), we get our final bound in (4):

$$\Pr[\, B_6 \,] \le \Pr[\, B_5 \,] + \Pr\left[\, B_6 \cap \overline{B_5} \,\right] \le \frac{3}{2^n} + \frac{2q^3}{2^{2n}} + \frac{2qn}{\alpha 2^n} = \frac{2q^3}{2^{2n}} + O\left(\frac{qn}{2^n}\right)$$

**Bounding Multi-Collisions (Proof of Proposition 7)** Consider $C_{12}(z)$ for some fixed value $z$. For simplicity, we will allow the adversary to make exactly $q$ queries to each function $h_1$ and $h_2$ (rather than at most $q$ total). The random process defining $|C_{12}(z)|$ is equivalent to the following.

- Two random tables $T_1$ and $T_2$ are selected, each consisting of $q$ random $n$-bit strings $T_1[i] = a_i$ and $T_2[j] = b_j$, $1 \le i, j \le q$.
- All $Q = q^2$ "balls" $y = (i, j)$ are mapped to "bins" $z(y) = T_1[i] \oplus T_2[j]$.
- $C_{12}(z)$ is set to be the number of balls thrown into bin $z$.

This is a special case of the general process of hashing $Q$ balls into $N = 2^n$ bins using simple 2-tabulated hashing. For this case, we will use the following result of [24] (Theorem 1, Equation (2) of their paper), who proved very strong Chernoff-type bounds for simple tabulation hashing.

**Theorem 4 ( [24], Theorem 1, Equation (2)).** *Let $T_1, T_2$ be random tables as above. Consider hashing $Q$ balls into $N \ge Q^{3/4}$ bins by simple 2-tabulation, and let $X[z]$ be the number of balls in bin $z$. Then for any $\gamma > 0$ there exists a constant $\alpha = \alpha(\gamma) > 0$, such that any fixed $z$ and any $\delta \ge 1$:*

$$\Pr\left[\, X[z] > (1+\delta) \cdot \frac{Q}{N} \,\right] \le (1+\delta)^{-\alpha(1+\delta)\frac{Q}{N}} + \frac{1}{N^\gamma} \tag{22}$$

To map this to our setting, we let

$$\gamma = 2,\ \alpha = \alpha(2),\ Q = q^2,\ N = 2^n,\ X[z] = C_{12}(z)$$

The constraint $N \ge Q^{3/4}$ becomes $2^n \ge q^{3/2}$, or $q \le 2^{2n/3}$, which is the precondition of Theorem 2. We also write $K = (1+\delta)Q/N = (1+\delta)q^2/2^n$, and note that we can choose any $K \ge 2q^2/2^n$. Then, since $1 + \delta \ge 2$, we get that for any $K \ge 2q^2/2^n$, (22) implies:

$$\Pr[\, C_{12}(z) > K \,] \le \frac{1}{2^{\alpha K}} + \frac{1}{2^{2n}}$$

[12] We did not "dig out" this constant from the complex calculations of [24], whose result we use, since it will not matter asymptotically for our bound. We assume $\alpha > 0.01$, for example.

Finally, we set $K = 2n/\alpha + 2q^2/2^n > 2q^2/2^n$. We get $\alpha K > 2n$, and thus

$$\Pr\left[ C_{12}(z) > \frac{2n}{\alpha} + \frac{2q^2}{2^n} \right] \leq \frac{2}{2^{2n}}$$

Taking the union bound over all $z$ implies

$$\Pr\left[ \max_z(C_{12}(z)) > \frac{2n}{\alpha} + \frac{2q^2}{2^n} \right] \leq \frac{2}{2^n}$$

The same argument as for $C_{12}(z)$ applies to $C_{23}(f)$ and $C_{13}(f)$, except we don't even need to take the union bound. This is because the question can be also reduced to tabulation hashing where the second table $T_2$ is using a "shifted" functions $h_3(c,d) \oplus c$ and $h_3(c,d) \oplus d$. Thus,

$$\Pr\left[ C_{13}(f) > \frac{2n}{\alpha} + \frac{2q^2}{2^n} \right] \leq \frac{2}{2^{2n}}$$

$$\Pr\left[ C_{23}(f) > \frac{2n}{\alpha} + \frac{2q^2}{2^n} \right] \leq \frac{2}{2^{2n}}$$

And overall $\Pr[B_5] \leq 2/2^n + 4/2^{2n} \leq 3/2^n$, as claimed.

**Bounding Inversion Probability (Proof of Proposition 8)** Let us set $K = \frac{2n}{\alpha} + \frac{2q^2}{2^n}$, and assume $B_5$ did not happen (otherwise $B_6 \cap \overline{B_5}$ does happen anyway), which means that $\max_z |C_{12}(z)|, |C_{13}(f)|, |C_{23}(f)| \leq K$.

It suffices to show that in this case $\Pr[(*, f) \in \mathbf{Eval}] \leq qK/2^n$, as this is the bound claimed in Proposition 8. For that it suffices to argue that for every query $i$ made by $\mathsf{A}$, $\Pr[(*, f) \in \mathbf{Eval}^i \backslash \mathbf{Eval}^{i-1}] \leq K/2^n$. Consider query $i$ made by $\mathsf{A}$, and split the argument into two cases.

*Case 1-2: query to $h_1$ or $h_2$.* Assume the $i$-th query is the query $(m_1, m_2)$ to $h_1$ (same proof for $h_2$). The same argument as we made in Proposition 6 implies that the answer $b$ defines a consistent triple producing an evaluation with output $f$ only if:

- There exists a tuple $((*, b), ((c, d), e)) \in C_{23}(f)$.
- The random answer $a = h_1(m_1, m_2)$ should be equal to $b \oplus c \oplus d$.

Since $|C_{23}(f)| \leq K$ and $a$ is random, the probability of this happening is at most $K/2^n$.

*Case 3: query to $h_3$.* Assume the $i$-th query is the query $(c, d)$ to $h_3$. The same argument as we made in Proposition 6 implies that the answer $e$ defines a consistent triple producing an evaluation with output $f$ only if:

- There exists a tuple $((*, a), (*, b)) \in C_{12}(c \oplus d)$.
- The random answer $e = h_3(c, d)$ should be equal to $f \oplus a \oplus c$.

Since $|C_{12}(c \oplus d)| \leq \max_z |C_{12}(z)| \leq K$ and $e$ is random, the probability of this happening is at most $K/2^n$.

This completes the proof.

## C Proof of Theorem 3 (cross-collision bound)

*Proof.* As in the case of $\mathrm{T}_5$, for $\mathrm{T}'_5$, we use the variables

- $m'_1$ and $m'_2$ to denote the left and right halves of various inputs to $h_1$;
- $a'$ to denote various outputs of $h_1$;
- $c'$ and $m'_4$ to denote the left and right halves of various inputs to $h_3$. We might use $d$ in place of $m'_4$ while talking about $h_3$ queries in general;
- $e'$ to denote various outputs of $h_3$;
- $M = (m'_1, m'_2, m'_3, m'_4)$ to denote various inputs to $\mathrm{T}'_5$;

– $f'$ to denote various outputs of $\mathrm{T}'_5$.

Hence, a valid computation of $\mathrm{T}'_5(M) = \mathrm{T}'_5(m'_1, \ldots, m'_4)$ proceeds as follows:

1. Set $a' = h_1(m'_1, m'_2)$.
2. Set $c' = a' \oplus m'_3$.
3. Set $e' = h_3(c', m'_4)$, and output $f' = e' \oplus m'_3$.

However, in case of $\mathrm{T}'_5$, all the queries are consistent. For any two queries $((m'_1, m'_2), a')$ and $((c', m'_4), e')$, we define $m'_3 = a' \oplus c'$, and say that this pair of queries (uniquely) defines a valid $\mathrm{T}'_5$ evaluation $(M', f')$, where $M' = (m'_1, m'_2, m'_3, m'_4)$ and $f' = e' \oplus a' \oplus c'$.

Most of the assumptions and definitions are the same as in the $\mathrm{T}_5$ collision proof. We only note the points of difference here.

– $\mathbf{Eval}^i$ now denotes the set of input-output pairs for the entire constructions $\mathrm{T}_5$ and $\mathrm{T}'_5$ known to $\mathsf{A}$ after its $i$-th query. We denote the set of $\mathrm{T}_5$ outputs by $\mathbf{Eval}^i_{\mathrm{T}_5}$ and similarly the set of $\mathrm{T}'_5$ outputs by $\mathbf{Eval}^i_{\mathrm{T}'_5}$. Thus, $|\mathbf{Eval}^i| = |\mathbf{Eval}^i_{\mathrm{T}_5}| + |\mathbf{Eval}^i_{\mathrm{T}'_5}|$.
– Bad event $B_3$: $|\mathbf{Eval}| \geq qn + q^2$.
– Bad event $B_4$: a non-trivial collision between $\mathrm{T}_5$ and $\mathrm{T}'_5$ is found; i.e., $(M, f), (M', f) \in \mathbf{Eval}$ with $M \in \{0,1\}^{5n}$, $M' \in \{0,1\}^{4n}$ and $g(M) \neq M'$, where $g(m_1, m_2, m_3, m_4, m_5) = (m_1, m_2, m_5, h_2(m_3, m_4) \oplus m_5)$.

Our goal remains the same, to upper bound the probability of $B_4$. We also note that the four bad events are still disjoint. Therefore, we have

$$\Pr[B_4] \leq \Pr[B_1] + \Pr\left[B_2 \cap \overline{B_1}\right] + \Pr\left[B_3 \cap \overline{B_1 \cap B_2}\right] + \Pr\left[B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\right]$$

The first two propositions and their proof are the same as above. However, for completion, we restate the propositions.

**Proposition 9** $\Pr[B_1] \leq \frac{2q^2}{2^n}$

**Proposition 10** $\Pr\left[B_2 \cap \overline{B_1}\right] \leq 3 \cdot \left(\frac{2q^2}{2^n}\right)^n$

**Proposition 11** $\Pr\left[B_3 \cap \overline{B_1 \cap B_2}\right] \leq \frac{2q^2}{n2^n}$

**Proposition 12** $\Pr\left[B_4 \cap \overline{B_1 \cap B_2 \cap B_3}\right] \leq \frac{nq^3}{2^n}$

Combining the results of Propositions 3,4,11,12, we get our final bound in (6):

$$\Pr[B_4] \leq \frac{2q^2}{2^n} + 3 \cdot \left(\frac{2q^2}{2^n}\right)^n + \frac{2q^2}{n2^n} + \frac{nq^3}{2^n} \leq \frac{nq^3 + 9q^2}{2^n}$$

Hence, it suffices to prove Propositions 11,12.

**Bounding Evaluation Set of $\mathrm{T}_5$ and $\mathrm{T}'_5$ (Proof of Proposition 11)** We break this into two parts. We note that since $|\mathbf{Eval}^i| = |\mathbf{Eval}^i_{\mathrm{T}_5}| + |\mathbf{Eval}^i_{\mathrm{T}'_5}|$, we must have $|\mathbf{Eval}| = |\mathbf{Eval}_{\mathrm{T}_5}| + |\mathbf{Eval}_{\mathrm{T}'_5}|$.

First, we prove that $\mathsf{A}$ cannot compute more than $q^2$ evaluations of $\mathrm{T}'_5$ in its $q$ queries.

Let us denote by $q_j$, for $j = 1, 2, 3$, the (random variable equal to the) number of calls to $h_j$. $\mathrm{T}'_5$ evaluations only depend on $q_1$ calls to $h_1$ and $q_3$ calls to $h_3$. Also, there is no computability condition for $\mathrm{T}'_5$. Hence, total number of $\mathrm{T}'_5$ evaluations is equal to $q_1 q_3$, which is less than $q^2$. Therefore, $|\mathbf{Eval}_{\mathrm{T}'_5}| \leq q^2$.

Now, we can follow the same steps as in the proof of Proposition 5 and the proof is completed.

**Final Collision Probability (Proof of Proposition 12)** For completeness, we repeat the claimed bound:

$$\Pr\left[\, B_4 \cap \overline{B_1 \cap B_2 \cap B_3} \,\right] \leq \frac{nq^3}{2^n}$$

Let $X_i$ be the event that none of bad events $B_1$, $B_2$ or $B_3$ happened, and that the $i$-th query of $\mathsf{A}$ generates a fresh collision between $\mathrm{T}_5$ and $\mathrm{T}'_5$, where $1 \leq i \leq q$. We upper bound $\Pr[X_i]$ separately depending on whether this was the call to $h_1$, $h_2$ or $h_3$. In each of these case we show $\Pr[X_i] \leq nq^2/2^n$, meaning that this bound is correct irrespective of the identity of the $i$-th query.

*Case 1: query to $h_1$.* Assume the $i$-th query is the query $(m_1, m_2)$ to $h_1$. In order for this query to cause a collision, one of two things must happen. First, it could be that this very query $(m_1, m_2)$ defined a value $a$ which induced two distinct consistent tuples

$$(\ ((m_1, m_2), a), ((m_3, m_4), b), ((c, d), e)\ ) \neq (\ ((m_1, m_2), a), ((c', d'), e')\ )$$

which collided. Here the first tuple corresponds to a valid $\mathrm{T}_5$ evaluation and the second tuple corresponds to a valid $\mathrm{T}'_5$ evaluation. But this means that $f = a \oplus (e \oplus c) = a \oplus (e' \oplus c')$, or $e \oplus c = e' \oplus c'$. But since we assumed $B_1$ failed, there are no collisions in "$c$-shifted" $h_3$, which means $(c, d) = (c', d')$. But this will imply the trivial collision which we have ruled out.

Hence, in order for the query $(m_1, m_2)$ to cause a collision in $\mathrm{T}_5$, there must be some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ s.t. the random answer $h_1(m_1, m_2) = a$ caused the collision with this $f$. Now again, there are two subcases:

*a) (*,f) comes from an evaluation of $\mathrm{T}'_5$.* In turn, this means that there is a query $(*, b) \in Q_2^{i-1}$ and $((c, d), e) \in Q_3^{i-1}$ such that the tuple $((*, b), ((c, d), e) \in C_{23}(f)$. Moreover, for the value $a$ to actually match this tuple, we must have $a = b \oplus c \oplus d$, which happens with probability $1/2^n$.

To summarize, in order for the $i$-th query to $h_1$ to cause the collision, the following events must happen:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}'_5}$ .
- There exists a tuple $((*, b), ((c, d), e)) \in C_{23}(f)$.
- The random answer $a = h_1(m_1, m_2)$ should be equal to $b \oplus c \oplus d$.

Moreover, since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}_{\mathrm{T}'_5}| \leq |\mathbf{Eval}_{\mathrm{T}'_5}| < q^2$ and $|C_{23}(f)| \leq n$ (as otherwise $X_i$ does not happen anyway). This means that

$$\Pr[X_i] \leq q^2 \cdot n \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

*b) (*,f) comes from an evaluation of $\mathrm{T}_5$.* Then, the query $(m_1, m_2)$ to $h_1$ must result in an evaluation of $\mathrm{T}'_5$ which causes a collision. For this to happen, the following conditions must be satisfied:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}_5}$ .
- The random answer $a$ can combine with any of the queries $((c, d), e)$ of $h_3$ such that $f = a \oplus c \oplus e$.

Since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}_{\mathrm{T}_5}| \leq |\mathbf{Eval}_{\mathrm{T}_5}| < nq$. Also, there are at most $q$ queries to $h_3$. The probability that $f = a \oplus c \oplus e$ is $1/2^n$ for each such query. Therefore,

$$\Pr[X_i] \leq nq \cdot q \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

*Case 2: query to $h_2$.* Assume the $i$-th query is the query $(m_3, m_4)$ to $h_2$. Now, this query itself cannot output two values that collide since this query does not give any new $\mathrm{T}'_5$ outputs. The only way this query can cause a collision is that there exists some previous value $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}'_5}$ such that the random answer creates a $\mathrm{T}_5$ output equal to $f$. This case is exactly similar to the first subcase of Case 1.

*Case 3: query to $h_3$.* Assume the $i$-th query is the query $(c, d)$ to $h_3$.

In order for this query to cause a collision, one of two things must happen. First, it could be that this very query $(c, d)$ defined a value $e$ which induced two distinct consistent tuples

$$(\ ((m_1, m_2), a), ((m_3, m_4), b), ((c, d), e)\ ) \neq (\ ((m'_1, m'_2), a'), ((c, d), e)\ )$$

which collided. Here the first tuple corresponds to a valid $\mathrm{T}_5$ evaluation and the second tuple corresponds to a valid $\mathrm{T}'_5$ evaluation. But this means that $f = a \oplus (e \oplus c) = a' \oplus (e \oplus c)$, or $a = a'$. But since we assumed there are no collisions on $h_1$, this means that $(m_1, m_2) = (m'_1, m'_2)$. But this will imply only the trivial collision, which we have ruled out.

Hence, in order for the query $(c, d)$ to cause a collision, there must be some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ s.t. the random answer $h_3(c, d) = e$ caused the collision with this $f$. Again, we have two subcases:

*a) (*,f) comes from an evaluation of* $\mathrm{T}'_5$. This means that there is a query $(*, a) \in Q_1^{i-1}$ and $(*, b) \in Q_2^{i-1}$ such that the values $m_5 = a \oplus c = b \oplus d$ match, which means that $a \oplus b = c \oplus d$, and, hence the tuple $((*, a), (*, b) \in C_{12}(c \oplus d)$. Moreover, for the value $e$ to actually match this tuple, we must have $e = f \oplus (a \oplus c)$ (which is then the same as $f \oplus (b \oplus d)$), which happens with probability $1/2^n$.

To summarize, in order for the $i$-th query to $h_3$ to cause the collision, the following events must happen:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}'_5}$.
- There exists a tuple $((*, a), (*, b)) \in C_{12}(c \oplus d)$.
- The random answer $e = h_3(c, d)$ should be equal to $f \oplus a \oplus c$.

Moreover, since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}_{\mathrm{T}'_5}| \leq |\mathbf{Eval}_{\mathrm{T}'_5}| < q^2$ and $|C_{12}(c \oplus d)| \leq n$ (as otherwise $X_i$ does not happen anyway). This means that

$$\Pr[X_i] \leq q^2 \cdot n \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

*b) (*,f) comes from an evaluation of* $\mathrm{T}_5$. Then, the query $(c, d)$ to $h_3$ must result in an evaluation of $\mathrm{T}'_5$ which causes a collision. For this to happen, the following conditions must be satisfied:

- There exists a tuple $(*, f) \in \mathbf{Eval}^{i-1}_{\mathrm{T}_5}$ .
- The random answer $e$ can combine with any of the queries $((m_1, m_2), a)$ of $h_1$ such that $f = a \oplus c \oplus e$.

Since we are only interested in the subcase when neither of the bad events $B_1, B_2, B_3$ happen, we can assume that $|\mathbf{Eval}^{i-1}_{\mathrm{T}_5}| \leq |\mathbf{Eval}_{\mathrm{T}_5}| < nq$. Also, there are at most $q$ queries to $h_1$. The probability that $f = a \oplus c \oplus e$ is $1/2^n$ for each such query. Therefore,

$$\Pr[X_i] \leq nq \cdot q \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

Having established that $\Pr[X_i] \leq nq^2/2^n$ for all $i$ (irrespective of what the $i$-th query was), we simply take the union bound over $i \in \{1 \ldots q\}$ to obtain

$$\Pr\left[\, B_4 \cap \overline{B_1 \cap B_2 \cap B_3} \,\right] \leq q \cdot \max_i \Pr[X_i] \leq \frac{nq^3}{2^n}.$$