T_5: Hashing Five Inputs with Three Compression Calls

Yevgeniy Dodis
New York University, NY, USA

Dmitry Khovratovich
Ethereum Foundation, Luxembourg, Luxembourg
Dusk Network, Luxembourg, Luxembourg

Nicky Mouha
Strativia, Largo, MD, USA

Mridul Nandi
Indian Statistical Institute, Kolkata, India

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.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases hash functions, Merkle trees, Merkle-Damgård, collision resistance

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.24

Related Version *Full Version*: `https://eprint.iacr.org/2021/373.pdf` [11]

Funding *Yevgeniy Dodis*: Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

Acknowledgements Thanks to the organizers and participants of Dagstuhl Seminar 18021 "Symmetric Cryptography," held from January 7-12, 2018 at Schloss Dagstuhl – Leibniz Center for Informatics. The T_5 construction was first presented there by one of the authors of this paper, and the fruitful discussions there provided an initial starting point for this paper.

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 λn-to-n-bit compression functions. Two well-known and widely-deployed constructions follow this approach:

- the *Merkle-Damgård construction* [10, 21], a sequential construction that is used in hash functions such as MD5, SHA-1 and SHA-2, and
- the *Merkle tree* [20], 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,[1] they both process t message blocks using t and $(t-1)$ compression function calls, respectively.

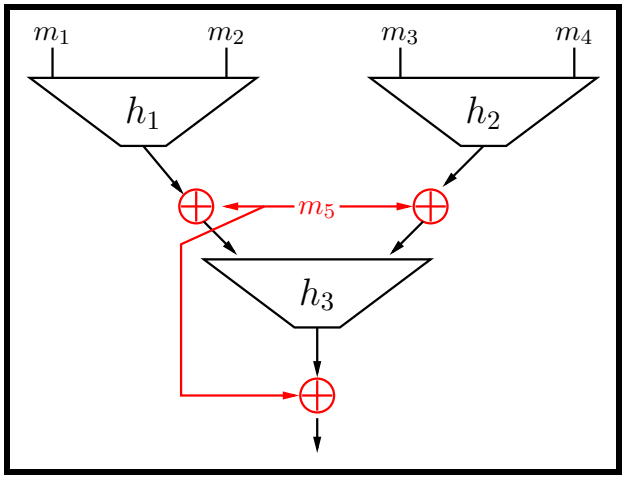

Figure 1 The T_5 construction with five message blocks m_1, m_2, m_3, m_4, m_5 and three compression function calls.

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 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

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

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.

$\mathbf{T_5}$ 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 $\mathbf{T_5}$. We prove the following security results for T_5 in terms of adversarial advantage after q queries to the inner compression functions:[2]

[2] 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.

- $q^2/2^n$ collision resistance, i.e., full-full collision resistance (CR) security (Theorem 2);
- $q^3/2^{2n} + q/2^n$ preimage resistance (Theorem 3).
- $q^2/2^n$ full-local CR security under a conjecture related to the 3-XOR problem (Proposition 7);
- $q^3/2^n$ full-local CR security unconditionally, i.e., 128-bit security for $n = 384$ (Theorem 4);
- $q^3/2^n$ local-local CR security under a conjecture related to the 4-XOR problem (Proposition 10);
- $q^4/2^n$ local-local CR security unconditionally, i.e., 128-bit security for $n = 512$ (Theorem 4).

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). Matching attacks for these security results are provided in the full version of this paper [11].

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 [16], 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 λn-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 [18, 19]).

Recently, McQuoid et al. [17] provided a general framework to prove the collision and second-preimage security of various hash functions. Our T_5 construction is covered by their framework. Unfortunately, their framework does not provide tight collision and second-preimage security bounds for 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. [15], Szydlo [32], and Berman et al. [4]. Given that we propose 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. [13]'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* 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 A and for all $i = 1$ to ℓ 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 π (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* Open *is a correct local opening for H with* Ver *is the verification function. For any adversary* 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 A *as*

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

And finally, 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 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 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

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

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

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

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

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

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 Ver satisfies

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

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 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.

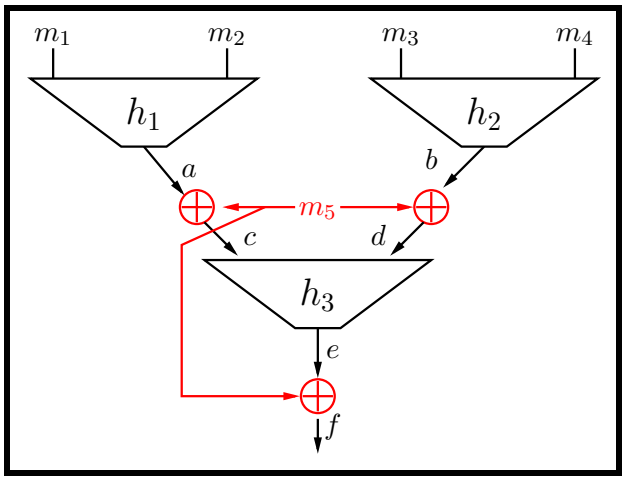

Figure 2 Modified 2-level Merkle tree $\mathrm{T}_5(m_1, m_2, m_3, m_4, m_5)$ with an extra input m_5 for the same 3 hash calls.

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 T_5;

- f to denote various outputs of 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 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 T_5 hash function. The following theorem shows that T_5 achieves nearly birthday collision security, despite hashing one more input than the traditional Merkle-Damgård function of depth 2.

▶ **Theorem 2.** *The* 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 full version of this paper [11]. 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 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 $2^{n/2}$, 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 the full version of this paper [11], T_5 is indeed not preimage resistant when $q > 2^{2n/3}$, so our result is tight.

▶ **Theorem 3.** *Assuming* $q \leq 2^{2n/3-1}$*, the* 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 full version of this paper [11], 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 T_5

Below we give the proof intuition for the simple, but natural special case where the adversary 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 full version of this paper [11], 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 A, making q total hash queries, is unlikely – up to birthday advantage – to succeed in the following four tasks.
(These tasks are formalized in the full version of this paper [11].)

1. **Task 1**: finding any simple collision in h_1 or h_2.[3]
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[4]

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

3. **Task 3**: generate more than nq valid evaluations of T_5.
4. **Task 4**: generate a non-trivial collision of 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 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 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 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.[5]

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 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 T_5 (call the output f) collides with one of at most nq already defined previous evaluations f' of 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

[3] For the general case, we will also need no collisions in a slightly modified variant of h_3.
[4] For the general case, we will also need similar guarantees for the combinations of $h_1 + h_3$ and $h_2 + h_3$.
[5] As we will see, in the general case a very different proof strategy will be needed to extend this argument to valid evaluations of T_5 completed by calls to h_1 and h_2.

the previously defined output f' of T_5 with probability at most $nq/2^n$, because from failing Task 3 we know there are at most nq previous evaluations of 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 full version of this paper [11]), 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 T_5 can increase by at most n for each new query to h_3. However, imagine that A first makes a query to h_2 with output b. Then 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 the full version of this paper [11] 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 T_5

As in Section 4.1, we will only consider the special case when the adversary 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.[6]

In this setting, there are several differences from the case of collision-resistance we considered so far. First, 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 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 T_5, this number of new evaluations 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 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 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"

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

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:[7]

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

Then we will be 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, 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.

[7] 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 $q \approx 2^{n/2}$.

5 Aggressive Opening for T_5

In this section we describe a non-trivial opening for T_5. We first note that a straightforward way to open a block m_i in T_5 is to provide all four siblings $m_{j\neq i}$. For a single T_5 the full-local and the local-local security (Section 3.3) definitions are the same and correspond to the collision security of 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 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.

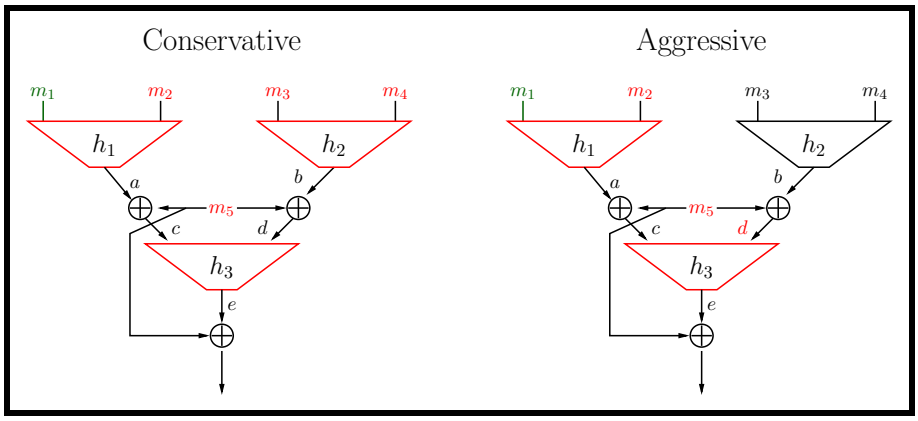

Figure 3 Conservative and aggressive openings for T_5. Green m_1 is opened. Red are opening elements (4 vs 3) and recomputed compression functions (3 vs 2).

Our (aggressive) local opening Open for T_5 is defined as follows, where we have $m = (m_1, m_2, m_3, m_4, m_5)$.

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 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 T'_5 is a by-pass computation of T_5 for the opening function defined above.

▶ **Theorem 4.**

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

and

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

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 T_5, and is provided in the full version of this paper [11]. 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 4 (Cross-Collision Bound)

Here, we give a proof sketch of the cross-collision advantage between T_5 and T'_5. We first note that in the case of T'_5, all queries are consistent. Hence q queries each to h_1 and h_3 can generate a maximum of q^2 T'_5 evaluations. And, as we have already seen before, q oracle queries can generate at most nq evaluations of 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 T_5 and 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 T_5 tuple and a 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 T_5 or T'_5 evaluation. If $(*, f)$ comes from a 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 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 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 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 T_5 tuple and a T'_5 tuple that collide, we again get only the trivial collision. If there is some previous T_5 evaluation with which this

query collides, then again, q such queries can combine with n evaluations of 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 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 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 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 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 strings of n bits. 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 5.** *(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 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 2 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 6.** *Given any valid-aware algorithm* A*, which makes at most* q *queries to oracles* h_1, h_2, h_3*, which solves* (10) *with probability* ϵ*, there is an algorithm* A' *making at most* $q < 2^{n/2}$ *queries to oracles* F_1, F_2, F_3 *(with runtime almost the same as* 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, A works as follows:

- It calls A to find a collision between T_5 and T'_5.
- When A queries $h_1(m_1, m_2)$ it is given $F_1(0||m_1||m_2)$.
- When A queries $h_2(m_3, m_4)$ it is given $F_1(1||m_3||m_4)$.
- When 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, A' returns $F_2(c||d) \oplus c$ to A. Otherwise A' flips a coin:
  - For tails A' selects a random entry of L and returns $F_3(c||d) \oplus c \oplus F_1(0||m_1||m_2)$ to A.
  - For heads A' returns $F_2(c||d) \oplus c$ to A.

Now suppose 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 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 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 7.** *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 8.** *(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 9.** *Given any algorithm A′ making at most q queries to all its oracles which solves the 4-XOR′ problem with probability ϵ there is an algorithm A making at most q queries to all its oracles (with run time almost same as A′) which solves 4-XOR problem with probability at least $\epsilon/4$.*

Proof. The reduction from A′ to A works as follows. We run 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 A returns $F_b(x)$. Similarly, if it is an $F'(z)$ query, A returns $F_{2+b}(z)$ to A′. Finally, A′ returns (x, y, z, w) such that $F(x) \oplus F(y) \oplus F'(z) \oplus F'(w) = 0$. Now, 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 A′ is independent of b. In other words, the view of A′ remains independent with the b values chosen by A. Thus, A succeeds with probability $1/4$ given that A′ succeeds. ◀

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

▶ **Proposition 10.** *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 T_5. This function is depicted in Figure 4.

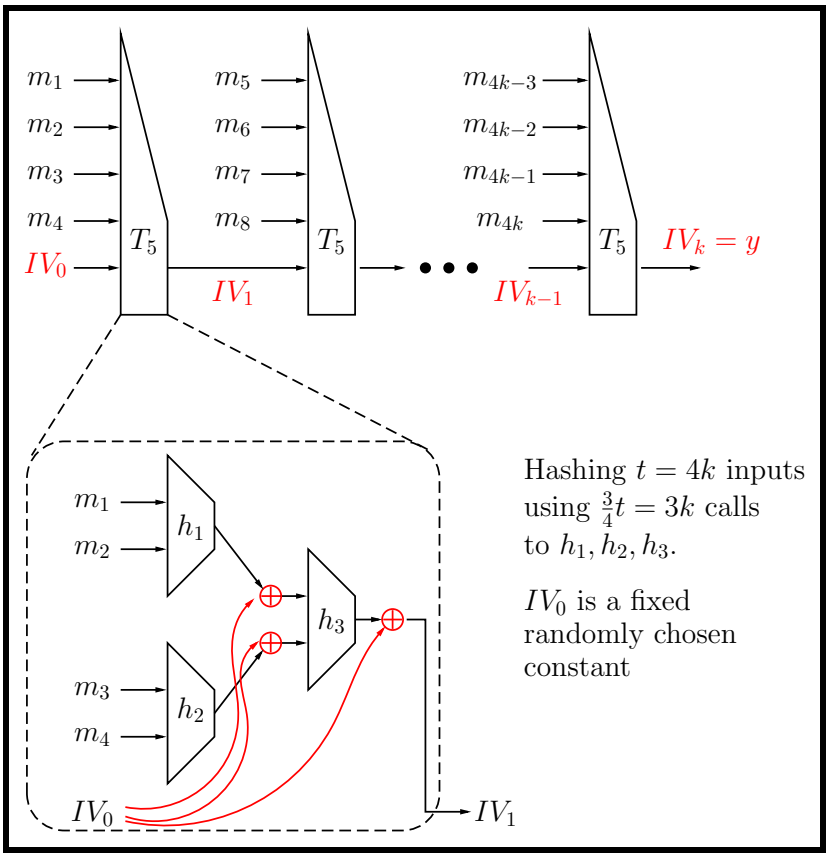

Figure 4 Merkle-Damgård Hash based on T_5.

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 λn-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 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 λ.

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 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 λn-to-n compression function H_λ (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 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_λ 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_λ 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 MT_λ 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_λ (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_λ (ideally 2^n, if one uses $2n$-to-n hash function as last building block).

Table 1 We compare standard Merkle trees to two variants of Merkle trees with 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	$n/2$	$n/2$	$n/3 \to \boxed{n/2}$
local-local CR security	$n/2$	$n/2$	$n/4 \to \boxed{n/3}$

7.2 Faster and Shallower Merkle Tree based on T_5

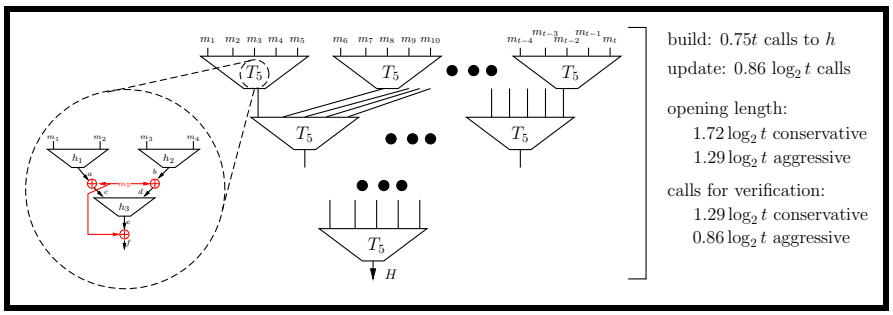

Figure 5 Full-blown tree based on T_5. Security is $2^{n/2}$ (tight) for conservative openings, $2^{n/3}$ provable and $2^{n/2}$ heuristic for full-local aggressive opening, $2^{n/4}$ provable and $2^{n/3}$ heuristic for local-local aggressive opening.

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 T_5 for H, which needed domain separation for functions h_1, h_2, h_3 (see the full version of this paper [11]), 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.

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 T_5, therefore using $(t-1)/4$ calls to 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 T_5, the resulting hash function H inherits the same global (or local opening) collision and preimage security as the compression function T_5:

- **Full-local (aggressive) security**: provable security up to (taking $poly(n)$ factors aside) $2^{n/3}$ queries (Theorem 4), heuristic security up to $2^{n/2}$ running time (Proposition 7).
- **Local-local (aggressive) security**: provable security up to $2^{n/4}$ queries (Theorem 4), heuristic security up to $2^{n/3}$ running time (Proposition 10).
- **Both full-local and local-local (conservative) security**: provable security up to $2^{n/2}$ queries (Theorem 2).

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 [14, 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 [12] 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 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 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.

References

1 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.

2 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

3 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

4 Piotr Berman, Marek Karpinski, and Yakov Nekrich. Optimal trade-off for merkle tree traversal. *Theor. Comput. Sci.*, 372(1):26–36, 2007.

5 Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer, 2015.

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

7 John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. *J. Cryptol.*, 22(3):311–329, 2009.

8 Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving algorithms for the 3xor problem. *IACR Trans. Symmetric Cryptol.*, 2018(1):254–276, 2018.

9 Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

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

11 Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi. T5: hashing five inputs with three compression calls. *IACR Cryptology ePrint Arch.*, 2021:373, 2021. Full version of this paper.

12 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

13 Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *CRYPTO*, volume 9216 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2015.

14 Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification: Version 2019.0-beta-37 [overwinter+sapling]. Technical report, Zerocoin Electric Coin Company, 2019. available at `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`.

15 Markus Jakobsson, Frank Thomson Leighton, Silvio Micali, and Michael Szydlo. Fractal merkle tree representation and traversal. In *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 2003.

16 Lars R. Knudsen and Bart Preneel. Construction of secure and fast hash functions using nonbinary error-correcting codes. *IEEE Trans. Inf. Theory*, 48(9):2524–2539, 2002.

17 Ian McQuoid, Trevor Swope, and Mike Rosulek. Characterizing collision and second-preimage resistance in linicrypt. In Dennis Hofheinz and Alon Rosen, editors, *TCC*, volume 11891 of *Lecture Notes in Computer Science*, pages 451–470. Springer, 2019.

18 Bart Mennink and Bart Preneel. Hash functions based on three permutations: A generic security analysis. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 330–347. Springer, 2012.

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

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

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

22 Mridul Nandi, Wonil Lee, Kouichi Sakurai, and Sangjin Lee. Security analysis of a 2/3-rate double length compression function in the black-box model. In *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2005.

23 Ivica Nikolic and Yu Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In *ASIACRYPT*, volume 9453 of *Lecture Notes in Computer Science*, pages 683–703. Springer, 2015.

24 Mihai Patrascu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14:1–14:50, 2012.

25 Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4, 2019. URL: `https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf`.

26 Thomas Peyrin, Henri Gilbert, Frédéric Muller, and Matthew J. B. Robshaw. Combining compression functions and block cipher-based hash functions. In *ASIACRYPT*, volume 4284, pages 315–331. Springer, 2006.

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

28 Yannick Seurin and Thomas Peyrin. Security analysis of constructions combining FIL random oracles. In *FSE*, volume 4593, pages 119–136. Springer, 2007.

29 Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 397–412. Springer, 2008.

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

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

32 Michael Szydlo. Merkle tree traversal in log space and time. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 541–554. Springer, 2004.

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

34 Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society, 2018.

35 Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764. Springer, 2019.