Short Collisions Separate Sponge from Merkle-Damgård

Ashrujit Ghoshal*

Abstract

We study short collisions in salted Merkle-Damgård (MD) and Sponge under the query-bounded preprocessing model introduced by Ghoshal and Tessaro (CRYPTO '23). The adversary makes at most T_1 oracle queries before a random salt is revealed and at most T_2 queries afterwards, and must output two colliding messages of at most B blocks.

Our main result is a separation between the two constructions. Let $N = 2^n$, give MD and Sponge matching n-bit salts, message blocks, and outputs, and set $T_1 = \Theta(N/T_2)$ with $T_2 = o(\sqrt{N})$. For Sponge, there is an attack that finds a collision with constant probability using only three-block messages. In contrast, every MD adversary restricted to $B = o(T_2)$ blocks succeeds with probability at most

$$O\left(\frac{B}{T_2} + \frac{T_2^2}{N}\right) = o(1).$$

Thus, under the same query budgets, three blocks suffice for Sponge, while every MD adversary with $B = o(T_2)$ blocks has success probability $o(1)$.

On the MD side, we prove an upper bound for arbitrary block bounds B. The proof uses a simple graph argument that distinguishes collisions already present in the preprocessing transcript from those that can actually be reached by a short computation starting from the subsequently sampled salt. This extends query-bounded preprocessing analysis beyond the previously understood one- and two-block cases.

On the Sponge side, we give a three-block attack that uses inverse permutation queries to make one precomputed collision useful from many different capacities. We complement the attack with a matching upper bound in a range containing the separation. In this range, allowing longer messages gives no asymptotic improvement: the optimal Sponge preprocessing advantage is already attained with at most three blocks.

Overall, our results show that collision length interacts fundamentally differently with preprocessing in the two constructions: under matched parameters and identical query budgets, Sponge admits constant-probability three-block collisions, whereas MD does not achieve constant success for any $B = o(T_2)$.

1 Introduction

Cryptographic hash functions map messages of arbitrary length to fixed-length digests and are a basic building block in cryptography. For example, a digital signature can authenticate a long message by signing its digest rather than the message itself. A central security requirement is *collision resistance*: it should be computationally difficult to find two distinct messages with the same digest.

A common way to hash variable-length messages is to iterate a fixed-input-length primitive, processing one message block at each step. The way in which these successive blocks are incorporated is therefore a fundamental part of the design and security of the hash function.

*IIT Madras. Email: ashrujit@cse.iitm.ac.in

Merkle–Damgård (MD) [Mer87, Dam87] and Sponge [BDPVA07] are two prominent approaches to this iteration, underlying SHA-2 and SHA-3, respectively [Nat15a, Nat15b]. Both process a message one block at a time while maintaining an internal state, but they update this state in very different ways.

In MD, the building block is a compression function $h : [N] \times [M] \to [N]$. A message is a sequence $m = (m_1, \ldots, m_\ell) \in [M]^\ell$, where each $m_i \in [M]$ is called a *message block*. Given an initial state $x_0 \in [N]$, the MD iteration computes $x_i = h(x_{i-1}, m_i), \quad i = 1, \ldots, \ell$, and outputs the final state x_ℓ. We write $\mathsf{MD}_h(x_0, (m_1, \ldots, m_\ell)) := x_\ell$ for this iterated computation.

Sponge instead iterates a permutation over a larger state. Fix a rate r and a capacity c, and let a state be $(u, b) \in \{0,1\}^r \times \{0,1\}^c$, where u is its *rate part* and b its *capacity part*. Let $\Pi : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ be a permutation. Given an initial state (u_0, b_0) and a message $m = (m_1, \ldots, m_\ell) \in (\{0,1\}^r)^\ell$, the Sponge iteration computes $(u_i, b_i) = \Pi(u_{i-1} \oplus m_i, b_{i-1})$, $i = 1, \ldots, \ell$. In the variant studied here, the hash output is the final rate u_ℓ. Thus MD applies a compression function to the current state and message block, whereas Sponge XORs the message block into part of a larger state and then applies a reversible transformation to the whole state.

We study idealized versions of these constructions: h is modelled as a random oracle [BR93, CDGS18] and Π is modelled as a random permutation [CDG18]. The adversary may query h in the MD setting, and both Π and Π^{-1} in the Sponge setting. This lets us isolate the security consequences of the two modes of iteration from the cryptanalysis of any particular compression function or permutation.

For an n-bit hash output, the most common benchmark for collision finding is the birthday bound: roughly $2^{n/2}$ evaluations suffice to find a collision with constant probability. This benchmark treats all of the attacker's work alike. In many settings, however, the primitive is fixed and public long before the particular computation to be attacked is known. An adversary can therefore spend substantial effort studying the primitive in advance. This leads to a different question: how much can such *preprocessing* reduce the work required once the eventual instance is revealed?

Preprocessing and time–space tradeoffs. If the entire hash computation were fixed during preprocessing, this question would be uninteresting: the adversary could simply find a collision in advance and store it. We therefore consider a salted setting in which the starting point of the computation is chosen only *after* preprocessing has ended. For MD, a salt $a \xleftarrow{\$} [N]$ becomes the initial state, so the computation starts from $x_0 = a$. For Sponge, a salt $a \xleftarrow{\$} \{0,1\}^c$ becomes the initial capacity, and the computation starts from $(0^r, a)$. The salt is public; its purpose is to ensure that preprocessing is performed before the starting point of the eventual computation is known.

To quantify the benefit of preprocessing, prior work has primarily used the time-space tradeoff model. The adversary may perform an arbitrarily expensive offline computation, but retain only S bits of advice. Once the salt is revealed, it is allowed T oracle queries to find a collision. A substantial line of work studies the tradeoff between these two resources [Hel80, Unr07, DTT10, CDGS18, CDG18, CGK18, CK19, ACDW20, CGLQ20, GLLZ21, FGK22, AGL22, FGK23, ADGL23, ABG+24].

For an iterated hash function, however, there is another resource that matters: the length of the colliding messages. To see why, suppose preprocessing finds two distinct message blocks $m, m' \in [M]$ such that $h(x, m) = h(x, m')$ for some $x \in [N]$. After learning the salt a, if the adversary finds a prefix p such that $\mathsf{MD}_h(a, p) = x$, then $p\|m$ and $p\|m'$ form a collision for the salted MD computation. Thus a collision prepared offline can remain useful even though the starting point is sampled only afterwards.

The length of the prefix p is important. Every block of p gives the adversary another state update with which to connect the freshly sampled starting point to information prepared offline,

but those blocks also become part of the final colliding messages. Consequently, an attack that is effective when arbitrarily long messages are allowed may cease to be useful when the collision must be completed within only a few blocks.

This phenomenon led to the study of *short collisions* under preprocessing. Akshima, Cash, Drucker, and Wee [ACDW20] brought the length of the colliding messages into focus by considering collisions whose messages contain at most B blocks. A sequence of subsequent works studied such bounded-length collisions for MD [GK22, AGL22] and for Sponge [FGK22, ADGL23]. In particular, these works show that the ability to exploit preprocessing can change substantially with the number of message blocks available to the adversary. Moreover, Freitag, Ghoshal, and Komargodski [FGK22] showed that one-block collisions can be found more efficiently in Sponge than in MD under preprocessing. Their result gives a separation between the two iteration paradigms, but in the particularly *restrictive* setting where the colliding messages consist of a single block. Once additional blocks are allowed, MD gains new ways to reach and exploit structure prepared during preprocessing. It is therefore unclear whether the separation is a peculiarity of the one-block setting or reflects a more general difference between MD and Sponge.

Charging for preprocessing. The (S, T) model leaves one resource uncharged: the work required to *produce* the advice. This distinction can be substantial. A collision has a short description and therefore occupies little space, but finding it may already require a birthday search. More generally, information that is useful from many possible starting points may be far more expensive to compute than to store.

Ghoshal and Tessaro [GT23] therefore considered a complementary query-bounded preprocessing model. The adversary makes at most T_1 oracle queries before the salt is sampled and at most T_2 queries afterwards, while being allowed to retain its entire preprocessing transcript. Thus the model charges for *constructing* the preprocessing information rather than for storing it. They obtained strong bounds for one- and two-block MD collisions.

The restriction to one or two blocks is important. As the short-collision literature shows, additional message blocks can give an adversary new ways to exploit information prepared during preprocessing. Consequently, bounds for one- and two-block collisions do not determine what happens when the adversary is allowed more blocks. Understanding query-bounded preprocessing for general block bounds therefore remains open.

This brings together the two threads above: how the power of preprocessing depends on the allowed collision length, and how costly the preprocessing itself is to construct. We ask:

Under query-bounded preprocessing, does restricting collisions to be short affect Merkle–Damgård and Sponge differently?

Our results. We show that it does, and that the earlier one-block separation of [FGK22] reflects a much broader phenomenon. In the query-bounded preprocessing model, under matched parameters, there are preprocessing and online query budgets for which a Sponge adversary finds a collision with constant probability using only three message blocks, whereas every MD adversary with the same budgets has vanishing success even when allowed a growing number of blocks. In particular, the separation persists for every $B = o(T_2)$, rather than only for a fixed constant number of blocks. Moreover, in a range containing the separation, our Sponge bounds are tight: allowing longer Sponge messages gives no asymptotic improvement, and the optimal advantage is already attained with at most three blocks.

For a concrete comparison, fix n and let $N = 2^n$. We give both constructions n-bit salts, n-bit message blocks, and n-bit outputs. MD has an n-bit state, while Sponge has rate $r = n$ and capacity

$c = n$. Thus, for any fixed message length, both constructions have the same $\Theta(\sqrt{N})$ birthday scale in the absence of preprocessing.

Theorem 1.1 (Short-message separation)**.** *Let $T_2 = o(\sqrt{N})$, and let $B \geq 3$ satisfy $B = o(T_2)$. Set $T_1 = \lceil N/T_2 \rceil$. Then:*

- *there is a Sponge adversary making at most T_1 preprocessing queries and T_2 online queries that finds a collision between two three-block messages with constant probability; whereas*
- *every MD adversary with the same query budgets, whose output messages have at most B blocks, succeeds with probability*

$$O\left(\frac{B}{T_2} + \frac{T_2^2}{N}\right) = o(1).$$

Thus three blocks already suffice for Sponge to obtain constant success, whereas every MD adversary with $B = o(T_2)$ blocks has vanishing success under the same query budgets. Our more general MD bound below gives a sharper interpretation: among block bounds $B \leq T_2$, achieving any fixed positive success probability requires $B = \Omega(T_2)$. In other words, under the separating budgets, Sponge can fully exploit its preprocessing within three blocks, whereas comparable MD success requires a message-length allowance on the order of the online query budget.

For example, take $T_2 = N^{1/3}$ and $T_1 = N^{2/3}$. Sponge finds a three-block collision with constant probability, whereas every MD adversary restricted to any fixed number of blocks has success probability $O(N^{-1/3})$. Viewed instead in terms of preprocessing cost, if the MD message length remains fixed and the online budget is $N^{1/3}$, then constant success requires $\Omega(N^{5/6})$ preprocessing queries, compared with $O(N^{2/3})$ for Sponge.

The separation follows from three results, which we describe next.

An upper bound for short MD collisions. For MD with a uniformly random compression function $h : [N] \times [M] \to [N]$, let $\mathsf{Adv}^{\leq B}_{\mathsf{MD},N,M}(T_1, T_2)$ denote the maximum success probability of an adversary whose two output messages have at most B blocks. For every $T_2 \geq B$, we prove

$$\mathsf{Adv}^{\leq B}_{\mathsf{MD},N,M}(T_1, T_2) \leq O\left(\min\left\{1, \frac{BT_1^2T_2}{N^2} + \frac{BT_1T_2^2}{N^2} + \frac{T_2^2}{N}\right\}\right). \tag{1}$$

When $T_1 \geq T_2$, the first term absorbs the second. Substituting $T_1 = N/T_2$ gives the MD part of Theorem 1.1.

A direct collision-counting argument cannot give the bound we need. In the regime relevant to our separation, preprocessing may make far more than $\sqrt{N}$ queries, so its transcript can already contain many collisions. The relevant question is therefore not whether preprocessing creates collisions, but whether the subsequently sampled salt can reach and exploit one of them within a short MD computation.

Our proof captures this distinction through a simple graph argument. We view the preprocessing transcript as a directed graph on the MD states and delete a controlled set of edges so that the remaining graph is acyclic and has at most one incoming edge at every vertex. Short paths in the resulting graph have very limited structure: a vertex has at most one predecessor at each distance. We then show that every successful collision either uses one of the deleted edges or gives rise to a short path between states sampled only after preprocessing. Bounding these two possibilities yields (1), with the same argument applying uniformly to arbitrary block bounds B.

A useful comparison is with the two-block analysis of Ghoshal and Tessaro [GT23]. Even for two blocks, their query-bounded argument is technically involved: they themselves describe it as a "fairly

involved analysis." The viewpoint above avoids a block-by-block treatment: once the preprocessing transcript is reduced to a graph with simple path structure, the message-length bound enters only through the length of the paths under consideration. The same argument therefore handles arbitrary B, including B growing with N. Our bound does not subsume all of the quantitative guarantees of [GT23]; rather, this simpler structural argument provides the generality needed for our MD-Sponge separation.

A THREE-BLOCK SPONGE ATTACK. Our second result is a query-bounded preprocessing attack on Sponge that uses only three-block messages. For rate r and capacity c, write $R = 2^r$ and $C = 2^c$.

Theorem 1.2 (Three-block Sponge attack, informal)**.** *Once the preprocessing budget is large enough to find a rate collision by a birthday search, there is a three-block Sponge adversary achieving*

$$\Omega\left(\min\left\{1, \frac{T_1 T_2}{C}\right\}\right)$$

success in the regime relevant to our separation.

The attack exploits inverse access to the permutation. A single collision found during preprocessing can be made useful from many different capacities, allowing the cost of the birthday search to be shared across the preprocessing. In the balanced setting $R = C = N$, this yields constant success with $T_1 = \Theta(N/T_2)$ preprocessing queries.

Inverse permutation queries were already central to the short-collision attacks of [FGK22]. Here, the key point is that this advantage survives when the queries used to construct the preprocessing are themselves charged: one birthday collision can be reused across many possible online starting points.

THE SPONGE TRADEOFF IS ALREADY ATTAINED WITH THREE BLOCKS. Our third result shows that the attack above is essentially optimal in a range containing the separation. For every $T_2 \geq B$, we prove

$$\mathsf{Adv}^{\leq B}_{\mathsf{Sp},R,C}(T_1, T_2) \leq O\left(\min\left\{1, \frac{T_1 T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right\}\right). \tag{2}$$

The first term is attained by our three-block preprocessing attack, while the remaining two terms are attained by birthday attacks using at most two blocks.

Consequently, allowing the Sponge adversary longer messages gives no asymptotic improvement: the optimal advantage is already attainable with at most three blocks. Under the separating budgets, constant-success MD attacks require a message-length allowance that grows with the online budget.

HOW ADDITIONAL BLOCKS HELP MD. We also give explicit MD attacks showing how additional message blocks can be used to reuse collisions prepared offline. Roughly, preprocessing creates collisions from a set of states, and an additional block is used to search for a transition into that set. Iterating this idea yields attacks for general block bounds. These attacks do not match our MD upper bound in every regime, and we do not claim a complete characterization of the optimal MD tradeoff as a function of B. Such a characterization is not needed for the separation.

INTERPRETATION. Our results concern idealized salted iterations on block sequences, without the padding and finalization rules of the standardized functions; moreover, our separating choice $r = c = n$ does not match the parameters of standardized SHA-3. Our theorem should therefore not be read as a security comparison between standardized SHA-2 and SHA-3.

It does, however, reveal a concrete distinction between the iteration paradigms underlying them. Under matched salt, block, and output lengths, Sponge admits a constant-success collision attack using only three message blocks at query budgets for which every MD adversary restricted to

$B = o(T_2)$ blocks has success probability $o(1)$. More generally, under these budgets, constant MD success requires a message-length allowance $\Omega(T_2)$, whereas three blocks already attain the optimal Sponge preprocessing tradeoff. In this sense, short-message collision resistance exposes an advantage of the MD-style iteration underlying SHA-2 that is not visible from the usual birthday-scale comparison with the Sponge-style iteration underlying SHA-3.

Paper Organization. In Section 2 we present an overview of our techniques for all the results and set up the prelimaries in Section 3. Then in Section 4 we present the upper bound proof for MD. We present the three-block attack for Sponge in Section 5 and formally prove our separation in Section 5.1. We present the upper bound for Sponge in Section 6. Finally, we present our MD attack in Appendix A.

2 Technical overview

At a high level, preprocessing is useful only if the structure created before the salt is known can still be reached and exploited afterwards. For short collisions this becomes a reachability question: every additional message block gives the adversary one more state update with which to use its preprocessing.

The separation comes from a basic difference between the two primitives. For Sponge, inverse access to the permutation lets preprocessing manufacture a transition *into a chosen capacity*. One collision can therefore be found once and then made useful from many different capacities at very little additional cost. For MD there is no corresponding inverse operation. Preprocessing may create many collisions, merges, and paths, but after the salt is sampled a short computation can reach only a limited part of this structure. Our MD upper bound formalizes this statement without assuming that the adversary follows any particular attack strategy.

In the balanced setting $R = C = N$, preprocessing $T_1 = \Theta(N/T_2)$ suffices for a constant-success three-block Sponge collision, whereas every MD adversary restricted to $B = o(T_2)$ blocks has vanishing success when $T_2 = o(\sqrt{N})$.

We first explain the three-block Sponge attack, since it makes the source of the separation transparent. We then turn to the MD upper bound and show how pruning the preprocessing graph reduces arbitrary B-block attacks to short-path counting. Combining the two gives the separation. Finally, we explain the matching Sponge upper bound and the MD attacks that illustrate how additional message blocks can help.

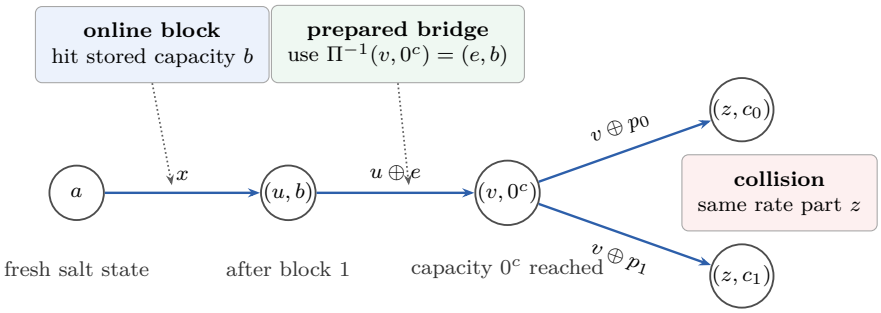


Figure 1: The three-block Sponge attack. The first online block only needs to land on a stored capacity b. A preprocessed inverse query then takes the walk to capacity 0^c, after which the same precomputed birthday collision yields two outputs with the same rate part z.

ONE SPONGE COLLISION, MANY USEFUL CAPACITIES. The Sponge attack has two phases. During *preprocessing*, before the salt is known, the adversary first prepares one collision and then builds a table that makes this same collision useful from many different capacities. After the salt is revealed, the *online* phase searches for one of these prepared capacities.

Preprocessing: finding one collision. The adversary queries $\Pi(p, 0^c)$ for $\Theta(\sqrt{R})$ distinct $p \in \{0,1\}^r$ and looks for two outputs with the same rate part. Since there are only R possible rate values, the standard birthday bound implies that, with constant probability, it finds distinct p_0, p_1 such that $\Pi(p_0, 0^c)$ and $\Pi(p_1, 0^c)$ have the same rate part.

This pair can be used to complete a collision from any state whose capacity is 0^c. Indeed, from a state $(v, 0^c)$, choosing the next block as $v \oplus p_0$ makes the next permutation input $(p_0, 0^c)$, while choosing $v \oplus p_1$ makes it $(p_1, 0^c)$. Since the corresponding outputs have the same rate part, these two one-block continuations collide.

Preprocessing: making the collision useful from many capacities. The adversary now makes inverse queries of the form

$$\Pi^{-1}(v, 0^c) = (e, b).$$

Such a query creates a one-block transition from every state whose capacity is b to a state whose capacity is 0^c. More precisely, from any state (u, b), choosing the next block as $u \oplus e$ makes the next permutation input (e, b) and hence the next state $(v, 0^c)$. From there, one further block completes the collision above.

Thus every inverse query adds another capacity b from which the same precomputed collision can be used. The expensive collision search is performed only once, additional useful capacities cost only one inverse query each.

Online phase. Only now is the random salt a revealed. The adversary tries first blocks x and observes

$$\Pi(x, a) = (u, b).$$

If b is one of the capacities stored during preprocessing, say the table contains

$$\Pi^{-1}(v, 0^c) = (e, b),$$

then the adversary outputs

$$(x,\ u \oplus e,\ v \oplus p_0) \qquad \text{and} \qquad (x,\ u \oplus e,\ v \oplus p_1).$$

The first block is chosen after the salt is known and reaches a prepared capacity b, the second follows the transition prepared offline to capacity 0^c, and the third completes the collision prepared offline. Hence the attack always outputs three-block messages. We give an illustration our attack in Figure 1.

At the end of preprocessing, suppose the table contains k distinct capacities. After the salt is revealed, each fresh online query produces an output whose capacity hits this table with probability on the order of k/C. Thus T_2 online trials succeed with probability $\Omega(\min\{1, kT_2/C\})$. After paying $O(\sqrt{R})$ preprocessing queries to obtain the collision, the remaining preprocessing budget can create $k = \Theta(T_1)$ useful capacities in the regime we consider. Hence the overall success probability is

$$\Omega\left(\min\left\{1, \frac{T_1 T_2}{C}\right\}\right).$$

In the balanced setting $R = C = N$, taking $T_1 = \Theta(N/T_2)$ therefore gives constant success while the output messages still have only three blocks.

There is one dependence issue hidden by this description: the inverse table, the birthday search, and the online phase all query the same random permutation. The formal proof handles this by constructing the inverse table first and conditioning on the resulting transcript. Since preprocessing exposes only a small fraction of the permutation, enough randomness remains both for the birthday search to succeed with constant probability and for each fresh online query to hit one of the stored capacities with probability $\Omega(k/C)$.

WHY SHORT MD COMPUTATIONS CANNOT EXPLOIT PREPROCESSING AS EFFICIENTLY. For MD, suppose preprocessing has found a collision from some state x. To use it after the salt a is sampled, the adversary must somehow make the MD computation reach x. With a long enough message there may be many ways to do this. With at most B blocks, however, only states reachable within B compression-function steps can be useful.

An upper bound cannot assume that the adversary explicitly prepares a table of collisions and then searches for it. It may choose its offline queries adaptively, create merging paths and cycles, and combine offline and online edges in arbitrary ways. We therefore work directly with the complete preprocessing transcript.

Represent every fresh offline query $h(x, m) = y$ by a directed edge $x \xrightarrow{m} y$ on the N MD states, where "fresh" means that the full compression-function input (x, m) has not previously been queried. The graph may be complicated: in the regime relevant to the separation, T_1 can be much larger than $\sqrt{N}$, so the transcript may already contain many collisions. Simply upper bounding the probability that preprocessing creates a collision is therefore useless. What matters is whether a short computation starting from the later sampled salt can *reach* the part of the transcript where such collisions become useful.

Our first step is to remove enough offline edges so that the remaining graph has at most one incoming edge at every vertex and contains no directed cycle. Process the offline queries in their original order and delete an edge if its output has either appeared earlier as an output or or appeared as the source state of the current or an earlier query. Let H be the graph of retained edges. We illustrate the edge-deletion procedure with a toy example in Figure 2.

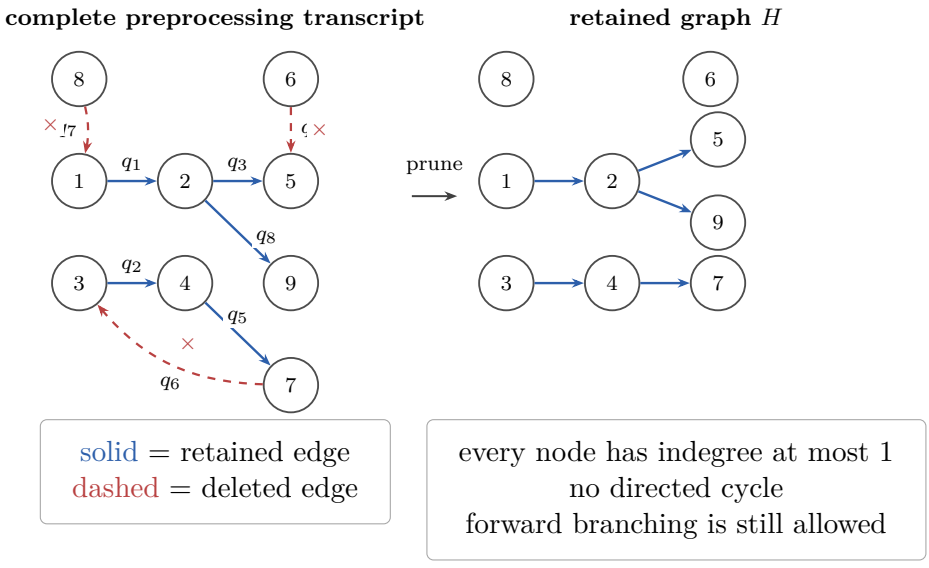


Figure 2: Removing offline edges from MD transcript. Deleted edges are precisely the ones whose output either repeats an earlier output or has already appeared as a source. The retained graph H therefore has indegree at most 1 and no directed cycle, but it may still branch forward.

Two properties are immediate. First, every vertex has at most one incoming edge: a second edge entering the same vertex would repeat an earlier output. Second, H is acyclic: on any directed cycle, the last queried edge would point to the source state of an edge that had already been queried, and would therefore have been deleted.

These two properties make short paths in H extremely rigid. Fix a vertex y. Since y has at most one incoming edge, there is at most one state at distance one from y when edges are followed backwards, at most one at distance two, and so on. Hence at most $B+1$ states can reach y along an H-path of length at most B. More globally, the number of ordered pairs connected by a positive-length H-path of length at most B is at most BT_1: for every possible endpoint and every distance, there is at most one predecessor.

The deleted edges are also controlled. Before the i^{th} fresh offline query is answered, at most $2i-1$ states can cause its edge to be deleted: the outputs seen earlier together with the source states of the current or an earlier query. The answer to the fresh query is uniform in $[N]$, even though the query itself may have been chosen adaptively. Summing over the preprocessing queries therefore gives at most T_1^2/N deleted edges *in expectation*.

Fix now the complete offline transcript. The adversary may continue to use any of the compression-function values revealed during preprocessing, but these are now fixed. The only new random states that can arise are the salt and the answers to compression-function inputs that are queried for the first time after preprocessing. Let Z_0 denote the salt, and let $Z_1, Z_2, \ldots$ denote these fresh answers, including any additional fresh answers needed to evaluate the two messages eventually output by the adversary. The online phase contributes at most T_2 such answers, and evaluating the two output messages contributes at most $2B$ more. Since $B \le T_2$, there are only $O(T_2)$ states Z_i altogether.

We first consider collisions whose evaluation uses one of the deleted offline edges. The key point is that creating many such edges during preprocessing is not by itself enough: after the salt is revealed, the evaluation must still be able to reach one of them within its B-block budget. For any fixed deleted edge, only $O(B)$ states can reach its tail within B retained edges. Thus, for a deleted edge to matter, one of the random states Z_i must land in this small set of possible starting states. This contributes an additional factor of $O(B/N)$ beyond the cost of creating the deleted edge itself.

Consider now a successful collision whose evaluation uses a deleted edge, and take the first deleted edge encountered. Immediately before this edge, the evaluation can only have followed retained offline edges since the most recent state among the Z_i's. Thus some Z_i reaches the tail of the deleted edge by an H-path of length at most B.

Conditioned on a transcript containing D deleted edges, at most $O(BD)$ states can reach the tail of one of them within B retained steps. Each Z_i therefore lands in this set with probability $O(BD/N)$. Averaging over $\mathbb{E}[D] = O(T_1^2/N)$ and union bounding over the $O(T_2)$ possible states Z_i gives

$$O\left(\frac{BT_1^2T_2}{N^2}\right).$$

The two factors of $1/N$ have different meanings: one comes from creating deleted edges during preprocessing, and the other from requiring a later random state to land within B retained steps of one of them.

It remains to consider successful collisions whose evaluations never use a deleted edge. Since H has indegree at most one, two distinct retained offline edges cannot be the final steps by which two paths merge. Since H is acyclic, a retained path cannot leave a state and later return to it. Consequently, after removing any common suffix of the two evaluations, a collision must involve randomness introduced after preprocessing: either two distinct post-preprocessing random states coincide, or two such states are connected by a nonempty H-path of length at most B. The

same conclusion covers unequal message lengths, since otherwise the unmatched part of the longer evaluation would form a directed cycle in H.

We are therefore left with one simple event: two distinct members of $\{Z_i\}$ are connected by an H-path of length at most B, where length zero corresponds to equality. There are N possible length-zero pairs and at most BT_1 positive-length pairs in H. A union bound over the $O(T_2^2)$ pairs of random states gives

$$O\left(\frac{T_2^2}{N} + \frac{BT_1T_2^2}{N^2}\right).$$

The first term is the ordinary birthday event $Z_i = Z_j$. For the second, H contains only $O(BT_1)$ ordered pairs joined by a positive-length path of length at most B: a fixed pair (Z_i, Z_j) lands on one of them with probability $O(BT_1/N^2)$, and there are $O(T_2^2)$ pairs of random states.

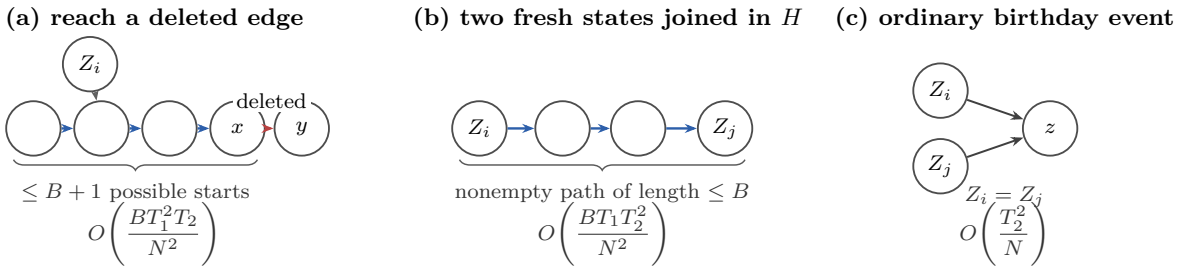


Figure 3: After deleting edges, every successful short MD collision is charged to one of three events: reaching a deleted edge, finding two fresh states joined by a short path in H, or the plain equality event $Z_i = Z_j$. These correspond exactly to the three terms in the upper bound.

Together with the deleted-edge case, this yields

$$O\left(\frac{BT_1^2T_2}{N^2} + \frac{BT_1T_2^2}{N^2} + \frac{T_2^2}{N}\right).$$

The three terms correspond respectively to reaching a deleted edge, connecting two post-preprocessing random states through a short retained path, and obtaining an ordinary birthday collision. No assumption is made about how the adversary organizes its preprocessing or interleaves offline and online information. We illustrate the three events underlying the bounds in the above expression in Figure 3.

The only dependence on B comes from how many predecessors can lie within distance B of a vertex. This is why the same argument extends directly to arbitrary block bounds.

Obtaining the separation. The separation now follows by putting the two arguments together. Return to the balanced setting $R = C = N$ and choose $T_1 = \Theta(N/T_2)$ with $T_2 = o(\sqrt{N})$. The Sponge attack has success $\Omega(T_1T_2/N) = \Omega(1)$ and still outputs only three-block messages.

For MD, substituting the same budgets into the upper bound gives

$$O\left(\frac{B}{T_2} + \frac{BT_2}{N} + \frac{T_2^2}{N}\right).$$

When $B = o(T_2)$, the first term is $o(1)$, the other two are also $o(1)$ because $T_2 = o(\sqrt{N})$. Hence the very preprocessing budget that already gives constant-success three-block collisions for Sponge is insufficient for every MD adversary whose message length is $o(T_2)$.

This is the structural source of the separation: MD gains from an extra block only by extending forward reachability by one more step, whereas Sponge inverse access can add another useful starting capacity without increasing the collision length at all.

Why longer messages do not improve the Sponge tradeoff. The attack mentioned earlier in the overview is enough for the separation. Our Sponge upper bound shows something stronger: in a range containing the separation, three blocks already achieve the best possible asymptotic preprocessing advantage.

The proof tracks capacities rather than full Sponge states. For every revealed permutation pair $\Pi(X) = Y$, draw an edge from the capacity of X to the capacity of Y and remember the full pair on that edge. A Sponge evaluation follows the directions of these edges, because absorbing a message block changes the rate of the next input but leaves its capacity equal to the capacity of the current output.

Fix the offline transcript. Unless the random salt is one of the capacities already exposed offline, the component containing the salt initially consists of a single vertex. After the salt is known, a fresh forward query lets the adversary choose the permutation input and returns a uniformly random unused output, a fresh inverse query lets it choose the output and returns a uniformly random unused input.

As long as the exploration neither reaches a capacity exposed during preprocessing nor repeats a capacity sampled after preprocessing, the component containing the salt remains a tree consisting entirely of post-preprocessing edges. The direction of a new edge is crucial. A forward query from a state in the salt component adds an edge from an existing vertex to a new child. An inverse query does the opposite: its random preimage contributes an edge from a new child *towards* an existing vertex. An evaluation beginning at the salt cannot traverse such an edge to reach the new child. Hence, until one of the relevant capacity collisions occurs, every permutation pair actually used by an evaluation from the salt was exposed by a forward query after preprocessing.

Because the salt component is a tree, two distinct messages follow distinct paths and therefore end on different forward-query edges. Their final states are distinct randomly sampled permutation outputs. For the two hashes to agree, these outputs must nevertheless have the same rate.

Thus, every successful collision must arise in one of three ways: a post-preprocessing computation reaches capacity information already exposed offline, two random permutation states revealed after preprocessing have the same capacity, or two random outputs revealed after preprocessing have the same rate. Making this fully precise is slightly more subtle because the adversary may use both forward and inverse queries; we defer the exact bad events and their accounting to the proof of Theorem 6.1. The three contributions are, respectively, $O\left(\frac{T_1 T_2}{C}\right)$, $O\left(\frac{T_2^2}{C}\right)$, $O\left(\frac{T_2^2}{R}\right)$. A union bound therefore gives

$$O\left(\frac{T_1 T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right).$$

Notice that no factor depending on B appears. The block bound is used only to ensure that the additional queries needed to evaluate the two output messages remain within $O(T_2)$.

Each of the three terms is achievable. The three-block preprocessing attack above attains the first. A birthday collision among the rates of first-block outputs attains the third using one-block messages. A birthday collision among their capacities attains the second using two-block messages: if the first-block outputs are (u, b) and (u', b), appending u and u' makes both next permutation inputs equal to $(0^r, b)$. Consequently, the upper bound is tight and every term is already achieved using at most three blocks. Allowing longer messages gives no asymptotic improvement.

How additional blocks help MD. The MD upper bound is sufficient for the separation, but explicit attacks also show how additional message blocks make preprocessing more useful. We first

describe the idea for two-block collisions.

Choose a set S of L states. During preprocessing, find a one-block collision starting from each state in S. A birthday search from one fixed state costs $O(\sqrt{N})$ queries, so preparing all L collisions costs $O(L\sqrt{N})$ queries.

Now consider a new starting state x from which we would like to prepare a two-block collision. For fresh message blocks m, the value $h(x, m)$ is uniform in $[N]$, and hence lands in S with probability L/N. After $O(N/L)$ queries in expectation, we therefore find a block m such that

$$h(x, m) = y \in S.$$

A one-block collision has already been prepared from y. Prepending m to the two colliding continuations gives a two-block collision from x.

Thus, starting from the L one-block collisions, we can prepare two-block collisions from K new states using an additional $O(KN/L)$ queries in expectation. The total preprocessing cost is therefore

$$O\left(L\sqrt{N} + \frac{KN}{L}\right).$$

Balancing the two terms at $L \asymp K^{1/2}N^{1/4}$ gives

$$O(K^{1/2}N^{3/4}).$$

The same idea can be applied recursively. To prepare b-block collisions from K states, first prepare $(b-1)$-block collisions from a set of L intermediate states, and then, from each of the K new starting states, search for one block whose compression-function output lands in that set. If $Q_b(K)$ denotes the resulting preprocessing cost, then

$$Q_b(K) \lesssim \min_L \left\{ Q_{b-1}(L) + \frac{KN}{L} \right\}, \qquad Q_1(K) = O(K\sqrt{N}).$$

Solving this recurrence gives, for every fixed b,

$$Q_b(K) = O_b\Big(K^{1/b}N^{1-1/(2b)}\Big).$$

How does this preprocessing help after the salt is revealed? There are two possibilities. The salt itself may be one of the K states from which a collision was prepared, in which case the adversary can use that collision immediately. Alternatively, the adversary can spend one online message block searching for a transition from the salt into the prepared set, and then append one of the stored collisions. Together with the ordinary online birthday attack, these give the MD attacks analyzed in Appendix A.

These attacks make explicit why the block bound matters for MD: each additional block gives another opportunity to reuse collisions prepared at other states. They do not match our upper bound in every regime, and we do not claim a complete characterization of the optimal MD tradeoff. For the separation, this is not needed: the upper bound already rules out every short MD strategy, while the Sponge attack achieves constant success with three blocks.

3 Preliminaries

We define the two hash constructions and the common experiment used to compare them. Throughout, algorithms and oracle queries are classical. The bounds concern query complexity: computation is unrestricted, and every dependence on the sampled primitive must come from the permitted queries.

3.1 Notation and constructions

For an integer $L \geq 1$, let $[L] = \{0, \ldots, L-1\}$, and identify $[2^k]$ with $\{0,1\}^k$ when applying bitwise exclusive-or, denoted $\oplus$. A message over an alphabet $\mathcal{M}$ is a nonempty block sequence $\mathbf{m} = (m_1, \ldots, m_\ell)$, of length $|\mathbf{m}| = \ell$. Write $\mathcal{M}^{\leq B} = \bigcup_{\ell=1}^{B} \mathcal{M}^\ell$. Distinct messages may have different lengths. Logarithms have base two. All asymptotic constants are absolute unless a dependence is indicated.

Merkle–Damgård. Let $N, M \geq 2$ and sample a uniform random function $h : [N] \times [M] \to [N]$. Given a salt $a \in [N]$ and a message $\mathbf{m} \in [M]^\ell$, define

$$s_0 = a, \qquad s_i = h(s_{i-1}, m_i) \quad (1 \leq i \leq \ell), \qquad \mathsf{MD}_a^h(\mathbf{m}) = s_\ell. \tag{3}$$

The adversary has oracle access to h.

Sponge. Let $r, c \geq 1$, put $R = 2^r$ and $C = 2^c$, and sample a uniform random permutation $\Pi : \{0,1\}^r \times \{0,1\}^c \to \{0,1\}^r \times \{0,1\}^c$. For $s = (u, v)$, write $\mathsf{rate}(s) = u$ and $\mathsf{cap}(s) = v$. Thus r and c are the rate and capacity in bits, and R and C are the sizes of the corresponding spaces. Given a salt $a \in \{0,1\}^c$ and a message $\mathbf{m} \in (\{0,1\}^r)^\ell$, set $s_0 = (0^r, a)$ and define

$$s_i = \Pi\big(\mathsf{rate}(s_{i-1}) \oplus m_i, \mathsf{cap}(s_{i-1})\big) \quad (1 \leq i \leq \ell), \qquad \mathsf{Sp}_a^\Pi(\mathbf{m}) = \mathsf{rate}(s_\ell). \tag{4}$$

The adversary has oracle access to both Π and Π^{-1}; either direction costs one query. Hash evaluation itself uses only the forward permutation Π.

A message block can therefore change the rate of the next permutation input to any chosen value, while its capacity must equal that of the current state. In particular, from a state (u, b), a known pair $\Pi(e, b) = (v, b')$ can be followed by choosing the block $u \oplus e$. We will use this observation in both the Sponge upper bound and the attack.

These definitions operate directly on block sequences, without additional padding, finalization, or squeezing. They fix what counts as a block in each result. Restricting the message domain preserves the upper bounds, but the attacks must then satisfy that restriction. Equality of Sponge outputs means equality of the final rates; equality of full states is not required.

Balanced parameters. The main separation sets $N = M = R = C = 2^n$ and $r = c = n$. Both salts, both message alphabets, and both output spaces then have size N. MD has an n-bit state; Sponge has a $2n$-bit state with n-bit rate and capacity components.

3.2 The collision experiment

An adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ whose code and initial randomness are independent of the sampled primitive. Its execution has two phases:

(i) Sample the primitive and run $\mathcal{A}_1$ with the oracle interface specified above. It makes at most T_1 queries and outputs a state σ.

(ii) Independently sample a uniform salt a and run $\mathcal{A}_2(\sigma, a)$ with access to the same primitive. It makes at most T_2 further queries and outputs $(\mathbf{m}, \mathbf{m}')$.

The experiment succeeds precisely when $\mathbf{m}, \mathbf{m}' \in \mathcal{M}^{\leq B}$, $\mathbf{m} \neq \mathbf{m}'$, and their hashes under salt a agree. An explicit failure output is unsuccessful. The salt space and message alphabet $\mathcal{M}$ are those of the chosen construction. The query budgets hold in every execution. There is no bound on $|\sigma|$, so the online algorithm may use the complete offline transcript. This is the (T_1, T_2) model of [GT23].

Write $\mathsf{Adv}^{\le B}_{\overline{\mathsf{MD}},N,M}(\mathcal{A})$ and $\mathsf{Adv}^{\le B}_{\overline{\mathsf{Sp}},R,C}(\mathcal{A})$ for the two success probabilities. Their suprema over adversaries with the specified budgets are denoted $\mathsf{Adv}^{\le B}_{\overline{\mathsf{MD}},N,M}(T_1,T_2)$ and $\mathsf{Adv}^{\le B}_{\overline{\mathsf{Sp}},R,C}(T_1,T_2)$. When the construction parameters and block bound are clear from context, we abbreviate these quantities by $\mathsf{Adv}^{\le B}_{\overline{\mathsf{MD}},N,M}(\mathcal{A})$ and $\mathsf{Adv}^{\le B}_{\overline{\mathsf{Sp}},R,C}(\mathcal{A})$, and by $\mathsf{Adv}^{\le B}_{\overline{\mathsf{MD}},N,M}(T_1,T_2)$ and $\mathsf{Adv}^{\le B}_{\overline{\mathsf{Sp}},R,C}(T_1,T_2)$, respectively. Budgets $T_1,T_2 \ge 0$ and block bounds $B \ge 1$ are integers; assumptions such as $T_2 \ge B$ are stated in the corresponding results. Evaluating the adversary's final output is part of the experiment, and does not count towards its query budget.

3.3 Revealing the primitive

A transcript lists queries and answers in order. A function query is *fresh* if its input has not been queried before. For a permutation, a revealed pair is always written as $\Pi(X) = Y$, regardless of the direction of the query that exposed it. A forward query at X is fresh if no revealed pair has input X, and an inverse query at Y is fresh if no revealed pair has output Y. Repeated queries can be answered from the transcript, so proofs need only track fresh queries.

Conditioned on a partial random-function transcript, every fresh answer is independent and uniform in $[N]$, even for an adaptively chosen input. For a random permutation, if q distinct pairs have been revealed, a fresh forward answer is uniform among the $RC - q$ unused outputs, and a fresh inverse answer is uniform among the $RC - q$ unused inputs. We call this answer the *random endpoint* of the pair; the other endpoint is specified by the query. If W is the random endpoint, then for sets $D \subseteq \{0,1\}^c$ and $E \subseteq \{0,1\}^r$ fixed before the answer is drawn,

$$\Pr[\mathsf{cap}(W) \in D] \le \frac{R|D|}{RC-q}, \qquad \Pr[\mathsf{rate}(W) \in E] \le \frac{C|E|}{RC-q}. \tag{5}$$

Both probabilities are conditional on the current transcript and the chosen query.

In an upper-bound proof, we also reveal any missing queries needed to evaluate the two output messages. This adds at most $2B$ fresh queries. Thus all edges used by a successful pair appear in the transcript after at most

$$Q := T_2 + 2B \tag{6}$$

fresh queries following preprocessing. When $T_2 \ge B$, we use $Q \le 3T_2$. This accounting allows the adversary to output messages whose evaluations it has not queried itself.

4 An upper bound for short MD collisions

The difficulty in bounding preprocessing attacks on Merkle–Damgård is to determine which of the collisions found in the offline phase actually matter after the salt is sampled. A message of at most B blocks can use only the part of the offline transcript that lies on a short path from the salt, or from a state generated by a fresh query after preprocessing. Thus a collision that exists somewhere in the offline transcript is useful only if the online computation can also *reach* it.

The proof therefore separates two questions. First, what structure in the offline transcript can cause two MD evaluations to merge? Second, after the salt is sampled, how likely is a message of at most B blocks to reach that structure? To answer the first question, we view the offline transcript as a directed labelled graph on the MD state space and remove the edges that create either a merge or a directed cycle. The remaining graph has indegree at most one and is acyclic, which makes its short paths easy to count. To answer the second question, we condition on the offline transcript. The salt and all fresh answers revealed afterwards are then independent uniform states. We show

that any successful short collision must either reach one of the removed edges or connect two of these random states by a short path in the remaining graph.

Theorem 4.1 (MD upper bound)**.** *Let $N, M \geq 2$, $B \geq 1$, and $T_1 \geq 0$. For $T_2 \geq B$,*

$$\mathsf{Adv}^{\leq B}_{\mathsf{MD},N,M}(T_1, T_2) \leq O\left(\min\left\{1, \frac{BT_1^2T_2}{N^2} + \frac{BT_1T_2^2}{N^2} + \frac{T_2^2}{N}\right\}\right).$$

The bound holds for every message alphabet size M, including for colliding messages of unequal lengths.

When $T_1 \geq T_2$, the first preprocessing term absorbs the second. We will prove a slightly more precise finite-query bound first; the theorem then follows by substituting the verification budget from Section 3.

The offline transcript as a graph. Fix an adversary, and let $G = (V, E)$ be the directed labelled graph induced by its fresh offline queries. Its vertex set is $V = [N]$. List the fresh offline queries, in the order in which they are made, as $h(x_i, m_i) = y_i$, $1 \leq i \leq t_1$, where $t_1 \leq T_1$. For each such query, add to E the directed labelled edge $x_i \xrightarrow{m_i} y_i$. Thus the tail and head of the edge are respectively the input and output states of the corresponding compression-function query, while the label records the message block. In particular, two edges with the same endpoints but different labels are distinct.

A direct analysis of G is challenging for two reasons. First, several offline edges may enter the same vertex, so two previously disjoint paths can merge. Second, G may contain directed cycles. We isolate these exceptional edges by deleting the i-th edge whenever

$$y_i \in \{y_1, \ldots, y_{i-1}\} \cup \{x_1, \ldots, x_i\}. \tag{7}$$

Let E_{del} be the set of deleted edges and set $H = G \setminus E_{\mathrm{del}}$.

The deletion rule is tailored to the two obstructions above. If y_i is an earlier answer, then retaining the edge could create a second incoming edge at an already reached state. If y_i is one of the tails seen up to time i, then retaining the edge could close a directed cycle; the case $y_i = x_i$ removes self-loops as well.

Every vertex of H has indegree at most one. Indeed, if two retained edges had the same head, then the later of the two would have been deleted because its output had already appeared as an earlier output.

Moreover, H is acyclic. Suppose, for contradiction, that H contains a directed cycle, and let the i-th query be the last queried edge on this cycle. The head y_i of this edge is the tail of another edge on the same cycle. Since every other edge on the cycle was queried no later than the i-th query, this tail belongs to $x_1, \ldots, x_i$. Hence $y_i \in \{x_1, \ldots, x_i\}$, so the i-th edge would have been deleted by (7), a contradiction.

Thus, for any $u, v \in [N]$, there is at most one directed path in H from u to v. Let $d_H(u, v)$ be its length when such a path exists, set $d_H(u, v) = \infty$ otherwise, and set $d_H(u, u) = 0$. The following lemma proves some facts about H that we will need.

Lemma 4.2. *The graph H satisfies*

$$\mathbb{E}[|E_{\mathrm{del}}|] \leq T_1^2/N, \tag{8}$$

$$|\{u : d_H(u, v) \leq B\}| \leq B + 1 \quad \textit{for every } v \in [N], \tag{9}$$

$$|\{(u, v) : d_H(u, v) \leq B\}| \leq N + BT_1. \tag{10}$$

Proof. For (8), for each $i \in [T_1]$, let D_i be the indicator of the event that the adversary makes an i-th fresh offline query and that the corresponding edge is deleted. Then $|E_{\mathrm{del}}| = \sum_{i=1}^{T_1} D_i$.

Consider the i-th fresh query, conditioned on the transcript immediately before this query is answered. Its input (x_i, m_i) may depend arbitrarily on the preceding transcript, but since the query is fresh, its answer y_i is uniform in $[N]$. The edge is deleted only if $y_i \in \{y_1, \ldots, y_{i-1}\} \cup \{x_1, \ldots, x_i\}$, and this set has size at most $2i - 1$. Hence $\Pr[D_i = 1] \le \frac{2i-1}{N}$. By linearity of expectation,

$$\mathbb{E}[|E_{\mathrm{del}}|] = \sum_{i=1}^{T_1} \Pr[D_i = 1] \le \frac{1}{N} \sum_{i=1}^{T_1} (2i - 1) = \frac{T_1^2}{N}.$$

For (9), fix v. Since every vertex of H has indegree at most one, there is at most one vertex at directed distance exactly d from v for each $d \ge 1$. Including v itself at distance zero gives at most $B + 1$ possible starting vertices.

For (10), the N pairs (u, u) account for distance zero. Now fix $d \in \{1, \ldots, B\}$. If $d_H(u, v) = d$, then v is the head of an edge of H, so there are at most $|E(H)| \le T_1$ possible choices for v. Once v and d are fixed, the indegree-one property determines at most one possible u. Thus there are at most T_1 ordered pairs at every positive distance $d \le B$, and summing over d proves the claim. □

Reducing a collision to two structural events. We now condition on the complete offline transcript, so the graphs G and H and the set E_{del} are fixed. Recall from (6) that $Q = T_2 + 2B$ fresh queries after preprocessing suffice to include both the adversary's online queries and any additional queries required to evaluate its two output messages.

Conditioned on the offline transcript, the salt is uniform in $[N]$, and each fresh random-function answer revealed afterwards is again uniform in $[N]$ and independent of the previous fresh answers. We may therefore equivalently sample

$$Z_0, Z_1, \ldots, Z_Q \xleftarrow{\$} [N]$$

independently in advance, use Z_0 as the salt, and use Z_i as the answer to the i-th fresh query after preprocessing. The inputs to those queries may be fully adaptive; only the answers are being sampled in advance. If fewer than Q fresh queries are made, the unused Z_i's are ignored.

After the adversary halts, we evaluate its two output messages and reveal any missing queries, as specified in Section 3. Thus every edge on either message evaluation is now exposed. We call an edge *post-preprocessing* if it is first exposed after the offline phase, including during this final verification. Every edge used by either message is therefore either an offline edge in G or a post-preprocessing edge whose head is one of $Z_1, \ldots, Z_Q$.

The deletion rule separates the offline graph into the retained edges in H and the deleted edges in E_{del}. We now use this decomposition to describe what a successful collision must look like.

Consider one of the two message evaluations. Suppose it uses a deleted edge, and let $u \xrightarrow{m} v$ be the first deleted edge on that evaluation. Immediately before traversing this edge, the evaluation is at u. Since no earlier deleted edge has been used, the part of the evaluation leading to u consists only of edges of H, except possibly for fresh queries made after preprocessing. Starting from the salt, or from the answer to the most recent such fresh query, the evaluation therefore follows a path in H to u. Because the message has length at most B, this path has length at most B.

Now suppose that neither message evaluation uses a deleted edge. Then every offline edge used by either evaluation belongs to H. Hence, between the salt or the answer to one fresh query and the next fresh query, each evaluation follows a path entirely inside H. Since the two distinct messages end at the same final state, we will show that this forces two distinct states among $Z_0, \ldots, Z_Q$ to be connected by a path in H of length at most B.

This motivates the following two events.

$$\mathcal{E}_1 = \{\exists i \in \{0, \ldots, Q\},\, u \xrightarrow{m} v \in E_{\text{del}} : d_H(Z_i, u) \leq B\}, \tag{11}$$

$$\mathcal{E}_2 = \{\exists i \neq j \in \{0, \ldots, Q\} : d_H(Z_i, Z_j) \leq B\}. \tag{12}$$

The event $\mathcal{E}_1$ says that the salt or some fresh post-preprocessing answer lies within B surviving offline edges of the tail of a deleted edge. The event $\mathcal{E}_2$ occurs when two distinct entries of the list $Z_0, \ldots, Z_Q$ are connected by a directed path in H of length at most B. Here "distinct" refers to their indices, not necessarily to their values. Thus, if $i \neq j$ but $Z_i = Z_j$, then $d_H(Z_i, Z_j) = 0$, and this case is already included in $\mathcal{E}_2$. We next show that if the adversary succeeds then atleast one of $\mathcal{E}_1, \mathcal{E}_2$ happen.

Lemma 4.3. *Every successful output implies $\mathcal{E}_1 \cup \mathcal{E}_2$.*

Proof. Let $\mathbf{m}$ and $\mathbf{m}'$ be distinct messages of length at most B whose MD evaluations from the common salt Z_0 end at the same state. View their evaluations as two directed labelled walks.

Suppose first that at least one walk uses a deleted edge. On that walk, take the first deleted edge and write it as $u \xrightarrow{m} v$. Look backwards from u to the most recent post-preprocessing edge on the same walk; if there is no such edge, look back to the start of the message. By the choice of the first deleted edge, every offline edge in this segment belongs to H, and by the choice of the most recent post-preprocessing edge, the segment contains no post-preprocessing edge. Its starting vertex is therefore some Z_i: Z_0 if no earlier post-preprocessing edge exists, and otherwise the head of the most recent such edge, and it gives an H-path from Z_i to u of length at most B. Hence $\mathcal{E}_1$ occurs.

We may now assume that neither walk uses a deleted edge. Thus every offline edge appearing on either walk lies in H. Starting from the common final state, delete the longest common suffix of identical labelled edges from the two walks. There are two cases.

One walk is exhausted. Suppose, without loss of generality, that the second walk is exhausted. The remaining part of the first walk is nonempty because the two messages are distinct. Moreover, this remaining walk starts at Z_0 and ends at Z_0: the common suffix that was removed from the first walk is exactly the entire second walk, whose starting state is Z_0. Hence the unmatched prefix is a nonempty closed walk from Z_0 back to Z_0.

This nonempty closed walk cannot consist entirely of edges of H, since H is acyclic. Hence it contains at least one post-preprocessing edge. Traverse the walk in its evaluation order, starting from its initial occurrence of Z_0 and ending at its final occurrence of Z_0, and let e be the last post-preprocessing edge encountered in this traversal. Let the head of e be Z_i. Here $i \in \{1, \ldots, Q\}$ because every post-preprocessing edge arises from a fresh query whose answer is one of $Z_1, \ldots, Z_Q$.

By the choice of e, every edge appearing after e in the walk was already present in the offline transcript. Since we are assuming that neither message uses a deleted edge, all of these edges belong to H. Therefore the portion of the walk after e is a directed path in H from Z_i to Z_0. Its length is at most B, and hence $d_H(Z_i, Z_0) \leq B$. Since $i \neq 0$, the event $\mathcal{E}_2$ occurs.

Both walks remain nonempty. Their last edges are different labelled edges with the same head: they are different because the common suffix was maximal, and they have the same head because both remaining walks end at the starting state of the removed suffix (or at the original common final state if the common suffix is empty). The two edges cannot both belong to H, since every vertex of H has indegree at most one.

If both last edges are post-preprocessing, then they are two different fresh queries, say with answers Z_i and Z_j for $i \neq j$. Their heads are equal, so $Z_i = Z_j$, and therefore $d_H(Z_i, Z_j) = 0$. Thus $\mathcal{E}_2$ occurs.

It remains that exactly one of the two last edges is post-preprocessing. Let its common head be Z_j. On the other walk, move backwards through the maximal final segment consisting only of edges of H. This segment is nonempty and ends at Z_j. Its starting point is either Z_0 or the head Z_i of the most recent post-preprocessing edge preceding the segment. Hence it gives an H-path of length at most B from one sampled state to Z_j. The starting index cannot equal j: otherwise this nonempty path would be a directed cycle in H. Thus two distinct indices are connected by a short H-path, and $\mathcal{E}_2$ occurs. □

□

Upper bounding $\Pr[\mathcal{E}_1]$. Fix a deleted edge with tail u. By (9), at most $B+1$ $z \in [N]$ satisfy $d_H(z,u) \le B$. Taking the union over all deleted edges, the set of states from which some deleted edge can be reached within B surviving offline steps has size at most $(B+1)|E_{\mathrm{del}}|$.

Since every Z_i is uniform in $[N]$, a union bound over $i \in \{0,\dots,Q\}$ gives

$$\Pr[\mathcal{E}_1 \mid \text{offline transcript}] \le \frac{(Q+1)(B+1)|E_{\mathrm{del}}|}{N}.$$

Averaging over the offline transcript and using (8),

$$\Pr[\mathcal{E}_1] \le \frac{(Q+1)(B+1)T_1^2}{N^2}. \tag{13}$$

Remark 4.4. This estimate exhibits the two independent obstacles to exploiting an offline collision: an exceptional edge must first be created, which costs T_1^2/N in expectation, and one of the post-preprocessing random states must then land within B steps of its tail, which contributes the additional factor $(Q+1)(B+1)/N$.

Upper bounding $\Pr[\mathcal{E}_2]$. Fix distinct indices $i \ne j$. Conditional on the offline transcript, (Z_i, Z_j) is uniform in $[N]^2$. By (10), at most $N + BT_1$ ordered pairs of states are joined by an H-path of length at most B. Therefore

$$\Pr[d_H(Z_i,Z_j) \le B \mid \text{offline transcript}] \le \frac{N+BT_1}{N^2} = \frac{1}{N} + \frac{BT_1}{N^2}.$$

There are $Q(Q+1)$ ordered pairs of distinct indices in $\{0,\dots,Q\}$, so

$$\Pr[\mathcal{E}_2] \le Q(Q+1)\left(\frac{1}{N} + \frac{BT_1}{N^2}\right). \tag{14}$$

Here the $1/N$ term is the ordinary birthday contribution $Z_i = Z_j$, while BT_1/N^2 accounts for two distinct sampled states that are connected through a short path already present in the offline transcript.

By Lemma 4.3, every successful output is contained in $\mathcal{E}_1 \cup \mathcal{E}_2$. Combining (13) and (14) therefore gives us

$$\mathsf{Adv}^{\le B}_{\mathsf{MD},N,M}(T_1,T_2) \le \min\left\{1, \frac{(Q+1)(B+1)T_1^2}{N^2} + Q(Q+1)\left(\frac{1}{N} + \frac{BT_1}{N^2}\right)\right\}. \tag{15}$$

Finally, when $T_2 \ge B$, we have $Q = T_2 + 2B \le 3T_2$ and $B+1 \le 2B$. Substituting these inequalities into (15) yields

$$\mathsf{Adv}^{\le B}_{\mathsf{MD},N,M}(T_1,T_2) \le O\left(\min\left\{1, \frac{BT_1^2T_2}{N^2} + \frac{BT_1T_2^2}{N^2} + \frac{T_2^2}{N}\right\}\right),$$

which proves Theorem 4.1.

The upper bound above gives the MD side of the separation. To complete the comparison, it remains to show that Sponge admits a substantially more effective use of preprocessing under the same query budgets. We do this next with an explicit three-block attack.

5 A three-block attack on sponge

The attack stores one collision starting at capacity zero and uses inverse queries to find many capacities from which that collision can be reached. After the salt is revealed, it searches for one of the stored capacities. Every successful search produces two three-block messages. The birthday computation is shared by the entire table, which is the saving over the MD strategy of Appendix A.

Theorem 5.1 (Three-block Sponge attack)**.** *Let $R = 2^r \geq 256$, $C = 2^c \geq 256$, and $B \geq 3$. If*

$$4\sqrt{R} + 4 \leq T_1 \leq \frac{\min\{R, C\}}{8}, \qquad 1 \leq T_2 \leq R/4,$$

then

$$\mathsf{Adv}^{\leq B}_{\mathsf{Sp},R,C}(T_1, T_2) \geq \Omega\left(\min\left\{1, \frac{T_1 T_2}{C}\right\}\right). \tag{16}$$

The attack respects both query budgets in every execution and outputs messages of exactly three blocks whenever it succeeds.

Inverse queries are a known ingredient of preprocessing attacks on sponge [FGK22]. The analysis below accounts for the cost of constructing the table and for the dependence between that table, the precomputed collision, and the online search.

Sharing one collision among many capacities. Suppose preprocessing has found distinct $m_0, m_1 \in \{0,1\}^r$ such that

$$\mathsf{rate}(\Pi(m_0, 0^c)) = \mathsf{rate}(\Pi(m_1, 0^c)). \tag{17}$$

A stored triple (e, b, v) satisfying $\Pi(e, b) = (v, 0^c)$ then permits use of this collision from any state with capacity b. Specifically, if an online query gives $\Pi(x, a) = (u, b)$, the messages

$$\mathbf{m}_0 = (x, u \oplus e, v \oplus m_0), \qquad \mathbf{m}_1 = (x, u \oplus e, v \oplus m_1) \tag{18}$$

collide. For $i \in \{0, 1\}$, their evaluations are

$$(0^r, a) \xrightarrow{x} (u, b) \xrightarrow{u \oplus e} (v, 0^c) \xrightarrow{v \oplus m_i} \Pi(m_i, 0^c), \tag{19}$$

where each arrow absorbs its labelled block and applies the permutation. The final rates agree by (17), and the messages differ in their third blocks.

Every inverse query at an output $(v, 0^c)$ supplies a triple of this form. Different triples can lead to the same capacity b, so the analysis must count distinct capacities. It must also ensure that the online search can still reach those capacities after the offline permutation pairs have been exposed.

Set $k = \left\lfloor \frac{T_1}{2} \right\rfloor$, and $d = \lceil 2\sqrt{R} \rceil + 1$. The offline phase first makes k inverse queries in order to obtain many capacity values from which capacity zero can be reached in one Sponge step. It then makes d forward queries at capacity zero to find a pair satisfying (17). We use this order so that, when analyzing the subsequent d forward queries at inputs $(m, 0^c)$, we may first fix the entire transcript of the preceding inverse queries.

Offline phase. Choose k distinct values $v_1, \ldots, v_k \in \{0,1\}^r$ and query $\Pi^{-1}(v_j, 0^c) = (e_j, b_j)$ for every j. Let $D = \{b_j : b_j \neq 0^c\}$ and retain one triple (e_j, b_j, v_j) for each capacity in D. Next make d fresh forward queries at inputs $(m, 0^c)$, choosing distinct values of m. If the value of $\Pi(m, 0^c)$ has already been determined by one of the preceding inverse queries, choose a different m. Retain any pair $m_0 \neq m_1$ satisfying (17). If no such pair exists, declare failure.

The offline phase makes $k + d$ queries. Since $k \leq T_1/2$ and $d \leq 2\sqrt{R} + 2 \leq \frac{T_1}{2}$, where the last inequality follows from $T_1 \geq 4\sqrt{R} + 4$, we have $k + d \leq T_1$.

Online phase. Given the salt a, make at most T_2 fresh forward queries at inputs (x, a), choosing distinct values of x. If the value of $\Pi(x, a)$ has already been determined by the offline transcript, choose a different x. If an answer (u, b) has $b \in D$, retrieve the stored triple corresponding to b and output (18). Otherwise, declare failure after the last query. No additional query is needed to construct the two messages, since the remaining permutation pairs in (19) were already exposed during preprocessing.

The algorithm described above sometimes requires choosing a new input of a prescribed capacity: during preprocessing we need fresh inputs of the form $(m, 0^c)$, and after the salt is revealed we need fresh inputs of the form (x, a). We now check that such a choice is always possible.

For any fixed capacity value, there are exactly R possible inputs, one for each choice of the rate. Throughout the attack, at most $T_1 + T_2$ permutation pairs are ever exposed. By the assumptions of the theorem,

$$T_1 + T_2 \leq \frac{R}{8} + \frac{R}{4} = \frac{3R}{8} < R.$$

Therefore, even in the worst case where every previously exposed input has the capacity required by the next query, fewer than R such inputs are already known. Hence at least one fresh input with the required capacity remains available.

Success probability analysis. We now analyze the two requirements needed for the attack to succeed. First, the inverse queries should produce many distinct nonzero capacities, so that the online phase has many targets to hit. Second, the subsequent forward queries at capacity zero should produce two outputs with the same rate. The next lemma shows that both events occur simultaneously with constant probability.

Lemma 5.2. *With probability at least* $1/3$*, preprocessing obtains both* $|D| \geq k/2$ *and a pair satisfying* (17)*.*

Proof. We first show that the k inverse queries produce many distinct nonzero capacities. Consider the j-th inverse query $\Pi^{-1}(v_j, 0^c) = (e_j, b_j)$. It fails to add a new capacity to D only if $b_j = 0^c$ or if b_j equals one of $b_1, \ldots, b_{j-1}$. Thus there are at most j capacity values that cause such a failure.

The inverse answer is uniform among the $RC - j + 1$ full states that remain available. Since each capacity occurs in exactly R full states, the conditional probability that the j-th query fails to add a new capacity is at most

$$jR/(RC - j + 1) \leq 2j/C\ .$$

Here we use $j \leq k \leq T_1/2 \leq C/16$, so in particular $RC - j + 1 \geq RC/2$.

It follows by linearity of expectation that

$$\mathbb{E}[k - |D|] \leq \sum_{j=1}^{k} 2j/C = k(k+1)/C\ .$$

Since $k \leq C/16$ and $k \geq 1$, we have $k + 1 \leq 2k \leq C/8$, and hence $\mathbb{E}[k - |D|] \leq k/8$. The random variable $k - |D|$ counts how many of the k inverse queries fail to contribute a new nonzero capacity

to D. We have shown that $\mathbb{E}[k - |D|] \le k/8$. Thus, on average, only a small fraction of the inverse queries are lost.

To have $|D| < k/2$, more than half of the k queries must be lost; equivalently, $k - |D| > k/2$. Since $k - |D|$ is nonnegative, Markov's inequality gives

$$\Pr[|D| < k/2] = \Pr[k - |D| > k/2] \le \Pr[k - |D| \ge k/2] \le \frac{\mathbb{E}[k - |D|]}{k/2} \le \frac{k/8}{k/2} = \frac{1}{4}.$$

Therefore

$$\Pr[|D| \ge k/2] \ge \frac{3}{4}.$$

We next analyze the d forward queries at inputs $(m, 0^c)$. Fix an arbitrary transcript of the preceding inverse queries. Suppose that the first $j-1$ forward answers have pairwise distinct rates. For each of these $j-1$ rates, there are C full output states with that rate. One has already been used by the corresponding forward query, and at most one can have been used by the inverse queries, because $v_1, \ldots, v_k$ are distinct. Hence at least $(j-1)(C-2)$ still-unused output states have a rate that has already appeared among the forward answers.

The next fresh forward answer is uniform among the unused output states. Therefore, conditioned on having seen no earlier rate collision, its probability of creating one is at least $(j-1)(C-2)/(RC) \ge (j-1)/(2R)$. Consequently, the probability that all d forward answers have distinct rates is at most

$$\prod_{j=1}^{d} \left(1 - \frac{j-1}{2R}\right) \le \exp\left(-\frac{d(d-1)}{4R}\right) \le e^{-1},$$

where the last inequality follows from $d = \lceil 2\sqrt{R} \rceil + 1$.

Thus, for every fixed inverse-query transcript, the forward queries find a pair satisfying (17) with probability at least $1 - e^{-1}$. In particular, this remains true after conditioning on $|D| \ge k/2$. Combining the two bounds, preprocessing obtains both desired properties with probability at least $(3/4)(1 - e^{-1}) > 1/3$. □

□

Fix a complete offline transcript with these two properties, and fix any salt a. There are $R|D|$ full output states with capacity in D. The outputs specified by the inverse queries all have capacity zero, which was excluded from D. Only the d forward answers can have removed states from this set. Hence at least

$$R|D| - d \ge Rk/2 - d \ge Rk/4 \tag{20}$$

favorable output states remain unused.

Condition further on failure in the first $j-1$ online queries. None of their answers has capacity in D, so none has removed a state counted in (20). The next fresh answer has capacity in D with conditional probability at least $(Rk/4)/(RC) = k/(4C)$. Thus, uniformly in the fixed transcript and salt,

$$\Pr[\text{online success} \mid \text{offline transcript}, a] \ge 1 - \left(1 - \frac{k}{4C}\right)^{T_2} \ge 1 - \exp\left(-\frac{kT_2}{4C}\right).$$

Averaging and applying Lemma 5.2 gives overall success at least $\frac{1}{3}(1 - \exp(-kT_2/(4C)))$. Since $k = \Theta(T_1)$ and $1 - e^{-x} = \Omega(\min\{1, x\})$ for $x \ge 0$, this proves Theorem 5.1.

5.1 Proof of the separation

We can now prove the main theorem, including the gap in the preprocessing needed for constant success.

Proof of Theorem 1.1. Set $T_1 = \lceil N/T_2 \rceil$. Since $T_2 = o(\sqrt{N})$, we have $4\sqrt{N} + 4 \le T_1 \le N/8$ and $B \le T_2 \le T_1$. Also $T_1 T_2 / N = \Theta(1)$. By Corollary 6.2, the Sponge attack succeeds with probability at least an absolute constant $\delta > 0$ and uses exactly three blocks.

For MD, substituting $T_1 = \Theta(N/T_2)$ into Theorem 4.1 gives

$$\mathsf{Adv}^{\le B}_{\widetilde{\mathsf{MD}},N,N}(T_1, T_2) \le O\left(\frac{B}{T_2} + \frac{BT_2}{N} + \frac{T_2^2}{N}\right) \le O\left(\frac{B}{T_2} + \frac{T_2^2}{N}\right) = o(1),$$

where $B \le T_2$ and the theorem's asymptotic assumptions justify the last two steps.

Finally, consider an MD adversary with the same online budget and at most $q_1 \le \eta N/\sqrt{BT_2}$ offline queries. Its success probability is bounded by

$$O\left(\eta^2 + \eta \frac{\sqrt{B}\, T_2^{3/2}}{N} + \frac{T_2^2}{N}\right) \le O(\eta^2) + o(1).$$

For every fixed $\varepsilon > 0$, choosing a sufficiently small constant $\eta > 0$ makes this less than ε for all large N. Thus constant MD success requires $\Omega_\varepsilon(N/\sqrt{BT_2})$ offline queries. □

The block restriction enters through the MD upper bound; the Sponge attack already uses three blocks. In particular, its extension to every allowed bound $B \ge 3$ requires no common suffix. Such a suffix would not in general preserve equality of the final Sponge rates.

6 An upper bound for short sponge collisions

For a sponge evaluation, absorbing a message block changes only the rate part of the current state; the capacity part is unchanged. Thus, once the output of one permutation call is fixed, the capacity of the next permutation input is fixed as well, while the message block determines its rate. This suggests representing the revealed permutation pairs only through their capacity values: for every revealed pair $\Pi(X) = Y$, we place an edge between $\mathsf{cap}(X)$ and $\mathsf{cap}(Y)$.

The proof studies the connected component of this graph that contains the salt. We show that, unless certain capacity values repeat, this component is a tree rooted at the salt: every newly revealed permutation pair that joins the component introduces a new capacity value. A sponge evaluation starting from the salt must then follow a directed path in this tree. We further show that two distinct messages cannot follow the same path. Hence their final permutation calls are different, and a collision is possible only if the rate parts of the two corresponding permutation outputs coincide.

Theorem 6.1 (Sponge upper bound)**.** *Let $R = 2^r$ and $C = 2^c$, where $r, c \ge 1$, and let $B \ge 1$ and $T_1 \ge 0$. For $T_2 \ge B$,*

$$\mathsf{Adv}^{\le B}_{\widetilde{\mathsf{Sp}},R,C}(T_1, T_2) \le O\left(\min\left\{1, \frac{T_1 T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right\}\right). \tag{21}$$

If $T_1 \ge T_2$, the first term absorbs the second.

The theorem allows the adversary to make both forward and inverse permutation queries, both before and after the salt is revealed. The terms involving $1/C$ arise from repetitions among capacity

values, while the term involving $1/R$ arises from a repetition among the rate parts of two fresh permutation outputs. We now make this precise.

THE PERMUTATION TRANSCRIPT. Fix an adversary, and let $Q = T_2 + 2B$. Thus at most Q fresh permutation pairs are exposed after preprocessing, including any additional pairs needed to evaluate the two messages output by the adversary.

Let the fresh pairs exposed during preprocessing as

$$\Pi(U_i) = V_i, \qquad 1 \le i \le t_1,$$

where $t_1 \le T_1$. Let

$$V_{\text{off}} = \{\mathsf{cap}(U_i), \mathsf{cap}(V_i) : 1 \le i \le t_1\}.$$

Thus V_{off} contains every capacity value appearing in a fresh permutation pair exposed before the salt is sampled, and $|V_{\text{off}}| \le 2T_1$.

Let $a \in \{0,1\}^c$ be the salt. We next describe the fresh permutation pairs exposed after preprocessing. Consider the j-th such pair. One full permutation state is specified by the query, while the other is sampled uniformly from the states that are still available on the corresponding side of the permutation. Let A_j denote the state specified by the query and let W_j denote the state sampled in response. Thus a forward query reveals $\Pi(A_j) = W_j$, whereas an inverse query reveals $\Pi(W_j) = A_j$.

Before W_j is sampled, define

$$\begin{aligned} D_j &= V_{\text{off}} \cup \{a\} \cup \{\mathsf{cap}(A_i), \mathsf{cap}(W_i) : i < j\} \cup \{\mathsf{cap}(A_j)\}, \\ E_j &= \{\mathsf{rate}(W_i) : i < j\}. \end{aligned}$$

The set D_j contains every capacity value that has already appeared, together with the salt and the capacity of the state A_j specified by the current query. Hence

$$|D_j| \le 2T_1 + 2j.$$

The set E_j contains the rate parts of the random states sampled in the previous fresh queries after preprocessing, and therefore $|E_j| \le j - 1$.

We consider the following three events:

$$\mathcal{F}_0 = \{a \in V_{\text{off}}\}, \tag{22}$$

$$\mathcal{F}_C = \{\exists j \le q : \mathsf{cap}(W_j) \in D_j\}, \tag{23}$$

$$\mathcal{F}_R = \{\exists j \le q : \mathsf{rate}(W_j) \in E_j\}, \tag{24}$$

where $q \le Q$ is the number of fresh pairs that are actually exposed after preprocessing.

The event $\mathcal{F}_0$ occurs if the salt has a capacity value that already appeared in the offline transcript. The event $\mathcal{F}_C$ occurs if the capacity of a newly sampled state repeats a capacity that has already appeared, equals the salt, or equals the capacity specified by the current query. Finally, $\mathcal{F}_R$ occurs if the rate of a newly sampled state agrees with the rate of an earlier sampled state.

We next show that if none of these events occurs, then two distinct messages cannot collide.

THE PERMUTATION PAIRS REACHABLE FROM THE SALT. We associate a graph with the revealed permutation pairs. Its vertices are the C possible capacity values. Each revealed pair $\Pi(X) = Y$ contributes an edge from $\mathsf{cap}(X)$ to $\mathsf{cap}(Y)$. We also remember which full permutation pair $\Pi(X) = Y$ produced each edge. When we speak about connected components or trees, we ignore the direction of the edges.

Assume first that neither $\mathcal{F}_0$ nor $\mathcal{F}_C$ occurs. Since $\mathcal{F}_0$ does not occur, the salt a is not the capacity of either endpoint of any offline pair. Thus, immediately after preprocessing, the vertex a is isolated from all offline edges.

Now consider a fresh pair exposed after preprocessing. Because $\mathcal{F}_C$ does not occur, the capacity $\mathsf{cap}(W_j)$ of the randomly sampled state has not appeared earlier, is different from a, and is different from $\mathsf{cap}(A_j)$. Therefore the edge corresponding to this pair always introduces a new capacity vertex, namely $\mathsf{cap}(W_j)$.

Adding an edge whose one endpoint is new cannot create a cycle and cannot join two components that already existed. It follows inductively that the connected component containing a is a tree. Moreover, this component never acquires an offline edge: it starts disjoint from all offline edges, and no subsequent edge can join it to another already existing component.

Root this tree at a. Consider an edge that is added to the component containing a. Its A_j-capacity is already in the component, whereas $\mathsf{cap}(W_j)$ is new. Thus $\mathsf{cap}(A_j)$ is the parent of $\mathsf{cap}(W_j)$ in the rooted tree.

The direction of the edge depends on whether the pair was exposed by a forward or an inverse query. For a forward query, $\Pi(A_j) = W_j$, so the edge is directed from the parent $\mathsf{cap}(A_j)$ to the child $\mathsf{cap}(W_j)$. For an inverse query, $\Pi(W_j) = A_j$, so the edge is directed from the child $\mathsf{cap}(W_j)$ to the parent $\mathsf{cap}(A_j)$.

We now consider an actual sponge evaluation starting from the salt. Suppose one permutation call outputs the state Y, and the next message block is absorbed before the following call. Absorption changes only the rate part, so the next permutation input X satisfies $\mathsf{cap}(X) = \mathsf{cap}(Y)$. Consequently, the sequence of permutation pairs used by the sponge evaluation gives a directed walk in the graph of capacity values, starting at a.

Such a walk cannot use an edge that was first exposed by an inverse query. Indeed, an inverse-query edge is directed from a child to its parent. Before the evaluation could traverse this edge, it would first have to reach the child. But the unique undirected path from the root a to that child contains the same edge, traversed from the parent to the child, whereas the edge is directed in the opposite direction. Hence the child is not reachable from a by a directed walk.

It follows that every permutation pair used by a sponge evaluation from the salt was first exposed by a forward query after preprocessing. Moreover, each such edge is directed from a parent to a child. Therefore every sponge evaluation follows a simple path in the rooted tree, starting at a and moving away from the root.

We next show that two distinct messages cannot follow the same path. Suppose a path uses the full permutation pairs $\Pi(X_1) = Y_1, \ldots, \Pi(X_\ell) = Y_\ell$.. These pairs uniquely determine the message blocks. The first block is $\mathsf{rate}(X_1)$, since the initial rate is zero. For every $k \geq 2$, the k-th block is $\mathsf{rate}(Y_{k-1}) \oplus \mathsf{rate}(X_k)$. Thus a fixed sequence of permutation pairs determines at most one message. Consequently, two distinct messages follow two distinct paths from the root.

Because the underlying graph is a tree, two distinct paths starting at the root end with different edges. Hence the final permutation calls of the two message evaluations correspond to two different fresh forward queries, say with outputs W_i and W_j for $i \neq j$.

Finally, assume that $\mathcal{F}_R$ also does not occur. Then the rate parts of all the sampled states $W_1, \ldots, W_q$ are pairwise distinct. In particular, $\mathsf{rate}(W_i) \neq \mathsf{rate}(W_j)$. The two message evaluations therefore produce different sponge outputs. We have proved

$$\{\text{successful output}\} \subseteq \mathcal{F}_0 \cup \mathcal{F}_C \cup \mathcal{F}_R. \tag{25}$$

Note that this argument does not require the two messages to have the same length.

Upper bounding the probability of the three events. We first assume that $T_1 + Q \le \frac{RC}{2}$.

Before any fresh query after preprocessing is answered, at most $T_1 + Q$ permutation pairs can have been fixed. Hence the random state W_j is sampled from a set of at least $RC/2$ available states.

Consider the event $\mathcal{F}_C$. Conditional on the entire execution before W_j is sampled, the set D_j is fixed and has size at most $2T_1 + 2j$. By the permutation-sampling bound from Equation (5),

$$\Pr[\mathsf{cap}(W_j) \in D_j] \le \frac{2|D_j|}{C} \le \frac{4T_1 + 4j}{C}.$$

This bound holds whenever the j-th fresh query is actually made, regardless of how the adversary chose that query from the preceding transcript. Therefore, summing over the at most Q possible query indices,

$$\Pr[\mathcal{F}_C] \le \sum_{j=1}^{Q} \frac{4T_1 + 4j}{C} = O\left(\frac{T_1 Q + Q^2}{C}\right).$$

Similarly, before W_j is sampled, the set E_j is fixed and has size at most $j - 1$. Again using (5),

$$\Pr[\mathsf{rate}(W_j) \in E_j] \le \frac{2|E_j|}{R} \le \frac{2(j-1)}{R}.$$

Hence

$$\Pr[\mathcal{F}_R] \le \sum_{j=1}^{Q} \frac{2(j-1)}{R} = O\left(\frac{Q^2}{R}\right).$$

The salt is sampled uniformly and independently of the offline transcript. Since $|V_{\text{off}}| \le 2T_1$,

$$\Pr[\mathcal{F}_0] \le \frac{2T_1}{C}.$$

Combining these three estimates with (25) gives

$$\mathsf{Adv}^{\le B}_{\widetilde{\mathsf{Sp}},R,C}(T_1, T_2) \le O\left(\min\left\{1, \frac{T_1(Q+1) + Q^2}{C} + \frac{Q^2}{R}\right\}\right). \tag{26}$$

It remains to remove the assumption $T_1 + Q \le RC/2$. If $T_1 + Q > RC/2$, then

$$\frac{T_1 Q + Q^2}{C} = \frac{Q(T_1 + Q)}{C} > \frac{RQ}{2}.$$

Since $R \ge 2$ and $Q = T_2 + 2B \ge 2$, the right-hand side is at least one. Thus the quantity inside the minimum in (26) is already at least a constant, and the same bound follows from the trivial upper bound $\mathsf{Adv}^{\le B}_{\widetilde{\mathsf{Sp}},R,C}(T_1, T_2) \le 1$.

Finally, if $T_2 \ge B$, then $Q = T_2 + 2B \le 3T_2$. Substituting this into (26) gives

$$\mathsf{Adv}^{\le B}_{\widetilde{\mathsf{Sp}},R,C}(T_1, T_2) \le O\left(\min\left\{1, \frac{T_1 T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right\}\right),$$

which proves Theorem 6.1.

6.1 Tightness of the sponge upper bound

The upper bound of Theorem 6.1 has three terms. The three-block attack of Theorem 5.1 achieves the preprocessing term T_1T_2/C. The other two terms are achieved by attacks that use no preprocessing.

Given the salt a, query $\Pi(m, a)$ for t distinct values of m. If two answers have the same rate, the corresponding one-block messages collide. If two answers have the same capacity, say $\Pi(m, a) = (u, b)$ and $\Pi(m', a) = (u', b)$, then the two-block messages (m, u) and (m', u') collide, since both second permutation inputs are $(0^r, b)$.

For $2 \le t \le R$, a repeated rate occurs with probability $\Omega(\min\{1, t^2/R\})$, and a repeated capacity occurs with probability $\Omega(\min\{1, t^2/C\})$. Hence, by choosing the better of the two attacks, we obtain

$$\Omega\left(\min\left\{1, \frac{t^2}{R} + \frac{t^2}{C}\right\}\right).$$

Together with Theorem 5.1, this matches the full upper bound.

Corollary 6.2 (Tight sponge advantage)**.** *Let $R = 2^r \ge 256$, $C = 2^c \ge 256$, and $B \ge 3$. Suppose*

$$4\sqrt{R} + 4 \le T_1 \le \frac{\min\{R, C\}}{8}, \qquad B \le T_2 \le \frac{R}{4}.$$

Then

$$\mathsf{Adv}^{\le B}_{\mathsf{Sp},R,C}(T_1, T_2) = \Theta\left(\min\left\{1, \frac{T_1T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right\}\right). \tag{27}$$

The lower bound is achieved by attacks using messages of at most three blocks.

Proof. The upper bound is exactly Theorem 6.1.

For the lower bound, put

$$A = \frac{T_1T_2}{C}, \qquad D = \frac{T_2^2}{C} + \frac{T_2^2}{R}.$$

By Theorem 5.1, there is a three-block attack with advantage $\Omega(\min\{1, A\})$. The attacks above give advantage $\Omega(\min\{1, D\})$ without preprocessing.

The adversary can choose whichever of these attacks has larger success probability. Since, for all $A, D \ge 0$,

$$\max\{\min\{1, A\}, \min\{1, D\}\} \ge \frac{1}{2}\min\{1, A + D\},$$

we obtain

$$\mathsf{Adv}^{\le B}_{\mathsf{Sp},R,C}(T_1, T_2) = \Omega\left(\min\left\{1, \frac{T_1T_2}{C} + \frac{T_2^2}{C} + \frac{T_2^2}{R}\right\}\right).$$

Together with the upper bound, this proves the claim. □

In the balanced setting $R = C = N$ with $T_2 \le T_1$, the two birthday terms are dominated by the preprocessing term, and (27) simplifies to

$$\mathsf{Adv}^{\le B}_{\mathsf{Sp},R,C}(T_1, T_2) = \Theta\left(\min\left\{1, \frac{T_1T_2}{N}\right\}\right).$$

Acknowledgements

This work was supported by an ANRF Early Career Research Grant (Grant No ANRF/ECRG/2025/000556/ENS), a New Faculty Initiation Grant at IIT Madras, the Arvind Raghunathan Center for Theoretical CS (aRtCS), the Mphasis F1 Foundation, the National Quantum Mission initiative of Dept of Science & Technology, India and CyStar, IIT Madras. The author thanks Ilan Komargodski for early discussions and feedback on an earlier draft, and Siddharth Iyer and Elaine Shi for their comments on that draft.

Statement on Generative AI Usage

We used ChatGPT 5.6 Sol and GPT-6 Astra exclusively for editorial assistance with the presentation and organization of the paper. The TikZ code for Figures 1 to 3 was generated using ChatGPT. All ideas, results, and proofs were human-developed.

References

[ABG+24] Akshima, Tyler Besselman, Siyao Guo, Zhiye Xie, and Yuping Ye. Tight time-space tradeoffs for the decisional diffie-hellman problem. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1739–1749, 2024.

[ACDW20] Akshima, David Cash, Andrew Drucker, and Hoeteck Wee. Time-space tradeoffs and short collisions in Merkle-Damgård hash functions. In *Advances in Cryptology - CRYPTO*, pages 157–186, 2020.

[ADGL23] Akshima, Xiaoqi Duan, Siyao Guo, and Qipeng Liu. On time-space lower bounds for finding short collisions in sponge hash functions. In *Theory of Cryptography Conference*, pages 237–270. Springer, 2023.

[AGL22] Akshima, Siyao Guo, and Qipeng Liu. Time-space lower bounds for finding collisions in Merkle-Damgård hash functions. In *Advances in Cryptology - CRYPTO*, pages 192–221, 2022.

[BDPVA07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *Advances in Cryptology - CRYPTO*, pages 693–721, 2018.

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In *Advances in Cryptology - EUROCRYPT*, pages 227–258, 2018.

[CGK18] Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–447, 2018.

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 673–684. IEEE, 2020.

[CK19] Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In *Theory of Cryptography - TCC*, pages 393–421, 2019.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - EUROCRYPT*, pages 203–216, 1987.

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *Advances in Cryptology - CRYPTO*, pages 157–186, 2010.

[FGK22] Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Time-space tradeoffs for sponge hashing: Attacks and limitations for short collisions. In *Annual International Cryptology Conference*, pages 131–160. Springer, 2022.

[FGK23] Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Optimal security for keyed hash functions: avoiding time-space tradeoffs for finding collisions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 440–469. Springer, 2023.

[GK22] Ashrujit Ghoshal and Ilan Komargodski. On time-space tradeoffs for bounded-length collisions in Merkle-Damgård hashing. In *Advances in Cryptology - CRYPTO*, pages 161–191, 2022.

[GLLZ21] Siyao Guo, Qian Li, Qipeng Liu, and Jiapeng Zhang. Unifying presampling via concentration bounds. In *Theory of Cryptography Conference*, pages 177–208. Springer, 2021.

[GT23] Ashrujit Ghoshal and Stefano Tessaro. The query-complexity of preprocessing attacks. In *Annual International Cryptology Conference*, pages 482–513. Springer, 2023.

[Hel80] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO*, pages 369–378, 1987.

[Nat15a] National Institute of Standards and Technology. Secure hash standard (SHS). Technical Report FIPS PUB 180-4, National Institute of Standards and Technology, August 2015.

[Nat15b] National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. Technical Report FIPS PUB 202, National Institute of Standards and Technology, August 2015.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In *Advances in Cryptology - CRYPTO*, pages 205–223, 2007.

A Preprocessing attacks on MD

For completeness, we give explicit attacks on MD in the (T_1, T_2) model as in the main text. These attacks are not used in the separation, and they do not match the upper bound of Theorem 4.1 in every parameter regime. Their purpose is to record the strongest MD attacks that we obtain from preprocessing.

Recall that $h : [N] \times [M] \to [N]$. For a fixed block bound B, the clean bounds below hold whenever

$$M \geq \alpha_B N^{1-1/(2B)}$$

for a sufficiently large constant α_B depending only on B. This assumption is only used to ensure that the required number of distinct message blocks is available. In particular, it is satisfied in the balanced setting $M = N$ used for the main separation. For smaller M, the same constructions can be truncated, but the resulting bounds acquire an additional dependence on M.

The simplest MD preprocessing attack finds one-block collisions independently at several starting values. When more blocks are available, one can do better. Suppose preprocessing already contains a short collision starting from each value in some set $H \subseteq [N]$. If a query $h(x, m)$ lands in H, then prepending m to the corresponding stored collision gives a collision starting from x. Thus a collection of shorter collisions can be reused to produce collisions from many additional starting values.

Preparing collisions for many starting values. Fix an offline transcript τ. For $b \geq 1$, let $\mathcal{C}_b(\tau) \subseteq [N]$ be the set of values x for which the transcript contains two distinct messages $\mathbf{m}, \mathbf{m}' \in [M]^{\leq b}$ satisfying $\mathsf{MD}^h_x(\mathbf{m}) = \mathsf{MD}^h_x(\mathbf{m}')$, together with all compression-function queries needed to evaluate the two messages. Thus, once preprocessing has produced τ, a collision of length at most b is already known for every $x \in \mathcal{C}_b(\tau)$.

We first describe how a set of such collisions can be extended by one block.

Lemma A.1. *Let $b \geq 2$. Let $H \subseteq [N]$ have size L, and suppose a collision of length at most $b-1$ is already known from every value in H. Let $1 \leq K \leq N/4$. Suppose there is a set $\mathcal{M}_{\mathrm{new}} \subseteq [M]$ of message blocks such that the current transcript contains no query whose second input lies in $\mathcal{M}_{\mathrm{new}}$, and $|\mathcal{M}_{\mathrm{new}}| \geq \alpha N/L$ for a sufficiently large absolute constant α. Then using $O(KN/L)$ additional queries, preprocessing can, with constant probability, find K distinct values x for which a collision of length at most b is known.*

Proof. Choose $2K$ distinct candidate values x. For each candidate, query $h(x, m)$ on $q = \lceil \alpha N/L \rceil$ distinct message blocks from $\mathcal{M}_{\mathrm{new}}$. By assumption, none of these query coordinates appears in the current transcript, so all of these queries are fresh.

For a fixed candidate x, the q answers are independent and uniform in $[N]$. Hence the probability that none lies in H is at most $(1 - L/N)^q \leq e^{-qL/N} \leq e^{-\alpha}$. Choosing α sufficiently large makes the probability that a candidate reaches H larger than, say, $3/4$.

The fresh query coordinates used for different candidates are disjoint, so the corresponding answers are independent. A standard Chernoff bound therefore shows that, with probability bounded below by an absolute constant, at least K of the $2K$ candidates reach H.

For every such candidate x, retain one block m_x for which $h(x, m_x) = z \in H$. If $\mathbf{m}_z$ and $\mathbf{m}'_z$ are the stored colliding messages from z, then $(m_x, \mathbf{m}_z)$ and $(m_x, \mathbf{m}'_z)$ are distinct messages of length at most b that collide from x. The total number of queries is $2Kq = O(KN/L)$. □

We can now build many short collisions without finding each of them independently.

Lemma A.2. *Fix $b \geq 1$. There are constants $\alpha_b, \delta_b > 0$, depending only on b, with the following property. Suppose $M \geq \alpha_b N^{1-1/(2b)}$. Then, for every $1 \leq K \leq N/4$, there is an offline algorithm using*

$$O_b\left(K^{1/b} N^{1-1/(2b)}\right) \tag{28}$$

queries such that, with probability at least δ_b, its transcript τ satisfies $|\mathcal{C}_b(\tau)| \geq K$.

Proof. We proceed by induction on b.

For $b = 1$, choose $4K$ distinct candidate values x. At each candidate, query $h(x, m)$ on $\alpha_1\sqrt{N}$ distinct message blocks. For sufficiently large α_1, the birthday bound implies that a one-block collision is found at a fixed candidate with probability at least $3/4$. Since the answers belonging to different first inputs are independent, with constant probability at least K candidates contain a one-block collision. The query cost is $O(K\sqrt{N})$, as required.

Now let $b \geq 2$ and assume the claim for $b-1$. If $K \leq \sqrt{N}$, use the base-case strategy above on $4K$ candidate values. With constant probability this prepares one-block collisions from at least K starting values, using $O(K\sqrt{N})$ queries, and $K\sqrt{N} \leq K^{1/b} N^{1-1/(2b)}$ because $K \leq \sqrt{N}$. Thus it remains to consider $K > \sqrt{N}$.

Partition the message alphabet into two disjoint sets $\mathcal{M}_{<b}$ and $\mathcal{M}_b$, each of size at least $\lfloor M/2 \rfloor$. We use only blocks in $\mathcal{M}_{<b}$ for the inductive construction and reserve $\mathcal{M}_b$ for the final extension. Choose $L = \left\lceil K^{(b-1)/b} N^{1/(2b)} \right\rceil$. Since $K > \sqrt{N}$, we have $L \leq K \leq N/4$. We may therefore apply the induction hypothesis, after relabeling the blocks in $\mathcal{M}_{<b}$, to obtain a set H of L values from which collisions of length at most $b-1$ are already known. This costs $O_{b-1}(L^{1/(b-1)} N^{1-1/(2(b-1))})$ queries. By construction, the resulting transcript contains no query whose second input lies in $\mathcal{M}_b$.

Next apply Lemma A.1 with $\mathcal{M}_{\text{new}} = \mathcal{M}_b$ to obtain K starting values that reach H in one query. This costs $O(KN/L)$ queries. For a sufficiently large choice of α_b, the assumption on M is sufficient for both steps. In particular, $N/L \leq N^{1-1/(2b)}$, so $\mathcal{M}_b$ contains enough reserved blocks for the extension, while $\mathcal{M}_{<b}$ is large enough for the induction hypothesis.

Substituting the chosen value of L, both query costs are $O_b(K^{1/b} N^{1-1/(2b)})$. The two stages each succeed with probability bounded below by a positive constant depending only on b, even conditioned on success of the preceding stage, so their joint success probability is also bounded below by such a constant. □

The point of Lemma A.2 is that the cost is no longer linear in the number K of starting values. Finding a fresh one-block collision independently at every starting value costs $\Theta(K\sqrt{N})$ queries. For $b > 1$, shorter collisions found earlier can be reused, reducing the dependence on K to $K^{1/b}$.

What a preprocessing budget can achieve. For $b \geq 1$, define

$$\Phi_b(T_1) = \begin{cases} 0, & T_1 = 0, \\ \dfrac{T_1^2}{N^2}, & 1 \leq T_1 \leq \sqrt{N}, \\ \dfrac{T_1}{N^{3/2}}, & \sqrt{N} \leq T_1 \leq N, \\ \dfrac{T_1^b}{N^{b+1/2}}, & N \leq T_1 \leq N^{1+1/(2b)}, \\ 1, & T_1 \geq N^{1+1/(2b)}. \end{cases} \tag{29}$$

These four ranges correspond to three simple preprocessing strategies.

When $T_1 \le \sqrt{N}$, choose one fixed value x and spend all T_1 queries on distinct blocks at x. A one-block collision is found with probability $\Omega(T_1^2/N)$. Since the random salt equals x with probability $1/N$, this gives advantage $\Omega(T_1^2/N^2)$.

When $\sqrt{N} \le T_1 \le N$, spend $\Theta(\sqrt{N})$ queries at each of $\Theta(T_1/\sqrt{N})$ distinct starting values. With constant probability, a constant fraction of these values acquire one-block collisions. A uniform salt belongs to this set with probability $\Omega(T_1/N^{3/2})$.

Finally, suppose $T_1 \ge N$. Let $c_b > 0$ be a sufficiently small constant and set

$$K = \left\lfloor c_b \min\left\{ \frac{T_1^b}{N^{b-1/2}},\, N \right\} \right\rfloor .$$

Choosing c_b small enough ensures both $K \le N/4$ and that the query cost in Lemma A.2 is at most T_1. Thus preprocessing can prepare collisions of length at most b from K distinct starting values with constant probability. A uniform salt belongs to this set with probability $K/N = \Omega_b(\Phi_b(T_1))$, giving the last two ranges in (29).

Consequently, whenever $M \ge \alpha_b N^{1-1/(2b)}$,

$$\mathsf{Adv}^{\le b}_{\mathsf{MD},N,M}(T_1, 0) \ge \Omega_b(\Phi_b(T_1)).$$

Using the online queries. Preprocessing need not prepare a collision for the salt itself. It can instead prepare collisions from many possible intermediate values and use the online phase to try to reach one of them.

Suppose preprocessing has stored colliding messages $\mathbf{m}_z \ne \mathbf{m}'_z$ of length at most $B-1$ for every z in some set H. After receiving the salt a, query $h(a, m)$ on several distinct first blocks. If one answer is $z \in H$, then $(m, \mathbf{m}_z)$ and $(m, \mathbf{m}'_z)$ are distinct messages of length at most B and collide from a.

Because the message alphabet contains only M blocks, at most $\min\{T_2, M\}$ distinct first-block queries can be useful. Put

$$t = \min\{T_2, M\}.$$

When preprocessing and online queries are both used, we reserve disjoint constant-fraction subsets of the message alphabet for the two phases. Thus the online phase still has $\Omega(t)$ distinct blocks available whose coordinates were never queried during preprocessing. The constant loss is absorbed in the $\Omega_B(\cdot)$ notation.

Theorem A.3 (MD attack)**.** *Fix $B \ge 2$. There is a constant $\alpha_B > 0$, depending only on B, such that if $M \ge \alpha_B N^{1-1/(2B)}$, then*

$$\mathsf{Adv}^{\le B}_{\mathsf{MD},N,M}(T_1, T_2) \ge \Omega_B\left(\min\left\{ 1,\, \frac{t^2}{N} + t\Phi_{B-1}(T_1) + \Phi_B(T_1) \right\} \right), \qquad t = \min\{T_2, M\}. \tag{30}$$

All three attacks respect the query budgets on every execution and output messages of at most B blocks.

Proof. We describe three attacks and then choose the best of them.

The first ignores preprocessing. After receiving the salt, query $h(a, m)$ on t distinct blocks. The usual birthday attack succeeds with probability $\Omega(\min\{1, t^2/N\})$.

The second uses preprocessing alone. Applying the construction described above with $b = B$ gives success probability $\Omega_B(\Phi_B(T_1))$.

For the third attack, preprocessing prepares shorter collisions and the online phase tries to reach one of them. Partition the message alphabet into disjoint sets $\mathcal{M}_{\text{off}}$ and $\mathcal{M}_{\text{on}}$, each of

constant-fraction size. Preprocessing uses only blocks in $\mathcal{M}_{\text{off}}$, while the online phase uses only blocks in $\mathcal{M}_{\text{on}}$. Hence every online query coordinate is fresh, regardless of the salt. Put

$$t' = \min\{T_2, |\mathcal{M}_{\text{on}}|\}.$$

Since $|\mathcal{M}_{\text{on}}| = \Omega(M)$, we have $t' = \Omega(t)$. We consider the same three preprocessing ranges as above.

If $T_1 \le \sqrt{N}$, use all T_1 offline queries at one fixed value z, using blocks from $\mathcal{M}_{\text{off}}$. A one-block collision is found there with probability $\Omega(T_1^2/N)$. Conditional on finding it, the t' fresh queries from the salt reach z with probability $\Omega(\min\{1, t'/N\})$. In the range before saturation, their product is $\Omega(tT_1^2/N^2) = \Omega(t\Phi_{B-1}(T_1))$.

If $\sqrt{N} \le T_1 \le N$, preprocessing, using only $\mathcal{M}_{\text{off}}$, produces one-block collisions from $\Theta(T_1/\sqrt{N})$ distinct starting values with constant probability. The set therefore occupies a fraction $\Theta(T_1/N^{3/2})$ of $[N]$. The t' online first-block queries hit this set with probability $\Omega(\min\{1, t'T_1/N^{3/2}\})$, which is $\Omega(\min\{1, t\Phi_{B-1}(T_1)\})$.

Finally, suppose $T_1 \ge N$. Apply Lemma A.2 with $b = B - 1$, using only blocks in $\mathcal{M}_{\text{off}}$. As above, choose the hidden constant in the number of prepared starting values small enough that the lemma's requirement $K \le N/4$ and the budget T_1 are both respected. Preprocessing obtains

$$K = \Omega_B\left(\min\left\{\frac{T_1^{B-1}}{N^{B-3/2}}, N\right\}\right)$$

starting values from which collisions of length at most $B - 1$ are known. Each of the t' online first-block answers is uniform in $[N]$ conditioned on the preprocessing transcript, so these queries reach the stored set with probability $\Omega(\min\{1, t'K/N\})$. This is $\Omega_B(\min\{1, t\Phi_{B-1}(T_1)\})$.

Thus the three available attacks have advantages $\Omega(\min\{1, t^2/N\})$, $\Omega_B(\min\{1, t\Phi_{B-1}(T_1)\})$, and $\Omega_B(\min\{1, \Phi_B(T_1)\})$. Choosing the best of the three loses only a constant factor compared with their sum: for nonnegative x, y, z,

$$\max\{\min\{1, x\}, \min\{1, y\}, \min\{1, z\}\} \ge \frac{1}{3}\min\{1, x + y + z\}.$$

This proves (30). □

For $T_1 \ge N$, before the corresponding terms saturate, Theorem A.3 gives the preprocessing-dependent advantage

$$\Omega_B\left(\frac{tT_1^{B-1}}{N^{B-1/2}} + \frac{T_1^B}{N^{B+1/2}}\right).$$

For $B = 2$, the mixed term is $tT_1/N^{3/2}$, while the offline-only term is $T_1^2/N^{5/2}$. For larger fixed B, preprocessing can reuse shorter collisions across progressively more starting values, which improves both terms once the offline budget is sufficiently large.