On Time-Space Lower Bounds for Finding Short Collisions in Sponge Hash Functions

Akshima[1], Xiaoqi Duan[2], Siyao Guo[3], and Qipeng Liu[4]

[1] NYU Shanghai[‡]
akshima@nyu.edu
[2] ETH Zürich
dogther66@gmail.com
[3] NYU Shanghai[‡]
siyao.guo.41@gmail.com
[4] UC San Diego
qipengliu0@gmail.com

Abstract. Sponge paradigm, used in the design of SHA-3, is an alternative hashing technique to the popular Merkle-Damgård paradigm. We revisit the problem of finding B-block-long collisions in sponge hash functions in the auxiliary-input random permutation model, in which an attacker gets a piece of S-bit advice about the random permutation and makes T (forward or inverse) oracle queries to the random permutation.

Recently, significant progress has been made in the Merkle-Damgård setting and optimal bounds are known for a large range of parameters, including all constant values of B. However, the sponge setting is widely open: there exist significant gaps between known attacks and security bounds even for $B = 1$.

Freitag, Ghoshal and Komargodski (CRYPTO 2022) showed a novel attack for $B = 1$ that takes advantage of the inverse queries and achieves advantage $\tilde{\Omega}(\min(S^2T^2/2^{2c}, (S^2T/2^{2c})^{2/3}) + T^2/2^r)$, where r is bit-rate and c is the capacity of the random permutation. However, they only showed an $\tilde{O}(ST/2^c+T^2/2^r)$ security bound, leaving open an intriguing quadratic gap. For $B = 2$, they beat the general security bound by Coretti, Dodis, Guo (CRYPTO 2018) for arbitrary values of B. However, their highly non-trivial argument is quite laborious, and no better (than the general) bounds are known for $B \geq 3$.

In this work, we study the possibility of proving better security bounds in the sponge setting. To this end,

- For $B = 1$, we prove an improved $\tilde{O}(S^2T^2/2^{2c}+S/2^c+T/2^c+T^2/2^r)$ bound. Our bound strictly improves the bound by Freitag et al., and is optimal for $ST^2 \leq 2^c$.
- For $B = 2$, we give a considerably simpler and more modular proof, recovering the bound obtained by Freitag et al.
- We obtain our bounds by adapting the recent multi-instance technique of Akshima, Guo and Liu (CRYPTO 2022) which bypasses the limitations of prior techniques in the Merkle-Damgård setting. To complement our results, we provably show that the recent multi-instance technique cannot further improve our bounds for $B = 1, 2$, and the general bound by Correti et al., for $B \geq 3$.

Overall, our results yield state-of-the-art security bounds for finding short collisions and fully characterize the power of the multi-instance technique in the sponge setting.

[‡] Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning, NYU Shanghai

Keywords: Collision · hash functions · Sponge · multi-instance · pre-computation · auxiliary input

1 Introduction

Sponge paradigm [BDPA07, BDPA08] is a novel domain extension technique for handling arbitrary long inputs based on a permutation $F : [R] \times [C] \to [R] \times [C]$ (where $C := 2^c$, $R := 2^r$ for bit-rate r and capacity c) with fixed input length. Specifically, a B-block message $\mathbf{m} = (m_1, \cdots, m_B)$ with $m_i \in [R]$ is hashed into $\mathsf{SP}_F(a, \mathbf{m})$ as follows: initialize $(x_0, y_0) = (0, a)$, and compute

$$(x_i, y_i) = F(x_{i-1} \oplus m_i, y_{i-1}) \text{ for } 1 \le i \le B; \text{ finally output } x_B$$

where $a \in [C]$ is the initialization salt[5]. We say $\mathbf{m} \neq \mathbf{m}'$ is a pair of B-block collision with respect to a salt a if they both have at most B blocks and $\mathsf{SP}_F(a, \mathbf{m}) = \mathsf{SP}_F(a, \mathbf{m}')$.

Sponge paradigm is an important alternative hashing technique to the popular Merkle-Damgård (MD) paradigm [Mer89, Dam89]. Notably, it has been used in the most recent hashing standard SHA-3. In this work, we are interested in the collision resistance property of sponge hash functions against preprocessing attackers, which can have an arbitrary (but bounded) precomputed advice about F to help.

Recently, several works have rigorously studied the algorithms for collision finding using preprocessing for Merkle-Damgård hash functions [DGK17, CDGS18, ACDW20, GK22, AGL22, FGK23], and significant progress has been made towards fully determining the optimal security bounds for all values of B [GK22, AGL22]. However, unlike the MD setting, the sponge setting draws much less attention [CDG18, FGK22], and our understanding is quite unsatisfactory. Significant gaps exist between known attacks and security bounds even for $B = 1$.

All of them [CDG18, FGK22] studied this question in the auxiliary-input random permutation model (AI-RPM) proposed by Coretti, Dodis and Guo [CDG18]. In this model, F is treated as a random permutation, and an adversary $\mathcal{A}$ consists of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. In the offline stage, (computationally unbounded) $\mathcal{A}_1$ precomputes S bits of advice about F. In the online stage, $\mathcal{A}_2$ takes this advice, and receive a random challenge salt a as the initialization salt of F. Next, it makes T oracle queries to F or F^{-1} during the attack, and finally output two messages that form the collision. We remark that inverse queries are not allowed in the MD setting, since the hash functions used by MD are usually one-way functions, while in sponge F is a invertible permutation.

Freitag, Ghoshal and Komargodski [FGK22] showed a novel attack for $B = 1$ that takes advantage of the inverse queries and applies the function inversion algorithms by Hellman [Hel80]. This attack achieves advantage $\Omega(\min(S^2T^2/C^2, (S^2T/C^2)^{2/3}) + T^2/R)$.

[5] In some practical sponge applications like SHA-3, this salt is usually set to 0. However, when we study the collision resistance of sponge hash functions in the *auxiliary input* model, such a fixed salt will make finding collisions trivial. [CDG18] identified this need for salting the hash functions for collision resistance in the *auxiliary input* model and so we are interested in the security bounds against a random initialization salt (just like what prior works [CDG18, ACDW20, AGL22, FGK22] did). See more details on the definition of the *auxiliary input* model below in section 2.4.

This is particularly interesting because it suggests that for some range of parameters (e.g., $ST^2 \geq C$), 1-block sponge hashing is less secure than 1-block MD hashing (for which the trivial attack by storing S collisions is known to be optimal [DGK17]). For $B \geq 2$, Freitag et al., based on an analogous attack for MD hashing given by Akshima, Cash, Drucker and Wee [ACDW20], showed an attack with advantage $\tilde{\Omega}(STB/C + T^2/\min(C,R))$ (the notations $\tilde{\Omega}, \tilde{O}$ hides poly-log factors).

In terms of security upper bounds, Coretti, Dodis and Guo [CDG18] proved an $\tilde{O}(ST^2/C + T^2/R)$ bound for any B, showing the optimality of the attack for finding $B \approx T$-length collisions. For other choices of B, only sub-optimal bounds are known for $B \leq 2$, and no better bound than $\tilde{O}(ST^2/C + T^2/R)$ is known for any $B \geq 3$. Specifically, Freitag et al. showed an $\tilde{O}(ST/C + T^2/R)$ bound for $B = 1$ and $\tilde{O}(ST/C + S^2T^4/C^2 + T^2/\min(C,R))$ bound for $B = 2$. Therefore, there is still a quadratic gap between the attack and security upper bound even for $B = 1$. On the contrast, optimal bounds are known for all constant values of B in the MD setting [DGK17, ACDW20, GK22, AGL22].

That motivates us to study the following question in this paper:

What is the optimal bound for $B = 1$?
Is there a better attack or security upper bound?

From the technical level, we are particularly interested in the multi-instance technique used to prove nearly optimal bounds for MD hashing [AGL22]. Specifically, it has recently been observed that the sequential random multi-instance technique by [AGL22] (referred to as multi-instance games technique in [AGL22]) subsumes the popular presampling technique [CDGS18, CDG18] and sequential distinct multi-instance technique [ACDW20] (referred to as multi-instance problem technique in [AGL22]). In the MD setting, it bypasses provable limitations of presampling technique [CDG18] and gives more modular proofs than sequential distinct multi-instance technique [ACDW20]. Moreover, the sequential random multi-instance technique successfully gave optimal bounds for various primitives even in the quantum setting [CGLQ20]. Therefore, we set out to understand the following question,

Can we prove better bounds or provide simpler proofs using multi-instance games?

In this work, we answer both questions.

1.1 Our results

Our first contribution is an improved bound for $B = 1$.

Theorem 1 (Informal). *The advantage of the best adversary with S-bit advice and T queries for finding 1-block collisions in sponge hash functions in the auxiliary-input random permutation model, is*

$$\tilde{O}\left(\frac{S^2T^2}{C^2} + \frac{S}{C} + \frac{T}{C} + \frac{T^2}{R}\right).$$

Our bound strictly improves the $\tilde{O}(ST/C + T^2/R)$ bound, and matches the best known attacks for most ranges of parameters. Note that $S/C, T/C, T^2/R$ terms match trivial or standard attacks, and the S^2T^2/C^2 term matches the attack $\min(S^2T^2/C^2, (S^2T/C^2)^{2/3})$ by [FGK22] as long as $ST^2 \leq C$. Notably, our bound is optimal for $ST^2 \leq C$.

We believe that further bridging the gap between the attack by [FGK22] and our bound is challenging. This is because their attack is obtained via connections with the function inversion problem for which an analog gap exists. Bridging the gap for the function inversion problem is a long standing open problem, and better security bounds would imply new classical circuit lower bounds, as shown by Corrigan-Gibbs and Kogan [CK19].

Our second contribution is a considerably simpler proof for $B = 2$, recovering one of the main results of [FGK22]. The original proof classified the collision structure into over 20 types, while we only need 8 types. This is possible because we do a careful analysis using the MI-games technique from [AGL22].

Theorem 2 (Informal). *The advantage of the best adversary with S-bit advice and T queries for finding 2-block collisions in sponge hash functions in the auxiliary-input random permutation model, is*

$$\tilde{O}\left(\frac{ST}{C} + \frac{S^2T^4}{C^2} + \frac{T^2}{\min(C,R)}\right).$$

We note that the term $ST/C + T^2/\min(C, R)$ matches the best known attack by [FGK22]. Therefore the above bound is optimal when the S^2T^4/C^2 term doesn't dominate the sum, i.e., $ST^3 \leq C$. However, this leaves an intriguing possibility of obtaining a better attack than ST/C for $ST^3 > C$, which will further confirm that sponge hashing is less secure than the MD hashing against preprocessing attackers (this message has been conveyed for $B = 1$ by [FGK22]).

We prove our results using the sequential distinct multi-instance technique (referred to as multi-instance problem technique in [AGL22]), and the sequential random multi-instance technique (referred to as the multi-instance game techniques in [AGL22]). It illustrates the power of the multi-instance technique over prior techniques in the sponge setting. The sequential distinct MI technique bypasses the limitation of the presampling technique (for $B = 1$) and sequential random MI technique gives more modular proofs (for $B = 2$). A comparison of our results with the prior works is summarized in Table 1.

The difference between sequential distinct MI technique and sequential random MI technique is in how the challenge games are defined. As the name suggests, in sequential distinct MI technique the game picks a random set of distinct challenge problems, the adversary is presented with one instance of the challenge problem at a time and has to solve all the instances of the distinct challenge problems to win. Whereas in the sequential random MI technique, the game picks a new randomly chosen instance of challenge problem each time, and the adversary gets that challenge only after solving all the previous challenges. Picking a random instance of the challenge problem allows the sequence of challenges to be independent of each other.

Roughly speaking, the sequential MI technique reduces proving ε security in the auxiliary input model against (S, T)-algorithms to proving $(\varepsilon/2)^S$ security in the (S, T)-multi-instance game. There are S stages in this game, and the adversary need to win all the S stages to win the whole game. In the i^{th} stage, the adversary will first receive a challenge salt a_i, then make T queries to F (or F^{-1} for sponge), and finally output a pair of messages m_A, m_B such that $SP_F(a_i, m_A) = SP_F(a_i, m_B)$. The adversary is allowed to use the queries from previous stages, but is no longer allowed to store advice bits. (See Section 2.4 for relevant definitions.)

Given that the sequential MI technique is used successfully to prove optimal bounds for various problems, such as finding 2-block collision in MD hash functions [AGL22], we wonder why we cannot prove better bounds for $B = 2$ in the sponge setting: is it an issue of our proofs or the technique. Therefore, we set out to understand the limitation of this technique.

Our third contribution is the following theorem, which implies that it is impossible to prove better bounds for any B using the sequential multi-instance technique in the sponge setting.

Theorem 3 (Informal). *Suppose $S, T, R \geq 16$. There are adversaries for finding 1-block collisions with advantage $(\widetilde{\Omega}(S^2T^2/C^2))^S$, and adversaries for finding 2-block collisions with advantage $(\widetilde{\Omega}(S^2T^4/C))^S$, and adversaries for finding 3-block collisions with advantage $(\widetilde{\Omega}(ST^2/C))^S$ when $T^2 < R$, in the (S,T)-multi-instance games of sponge hash functions.*

These lower bounds give limitations on the bound one can prove with multi-instance techniques. In particular, it implies that (using the multi-instance technique) the S^2T^2/C^2 term obtained in Theorem 1 cannot be improved. It also explains why Theorem 2 (also [FGK22]) cannot prove better than S^2T^4/C^2, and why no non-trivial bounds (i.e., better than ST^2/C) can be proved for $B \geq 3$. Together with our new security upper bounds and the general known bound[6] for the multi-instance games (summarized in Table 2), we fully characterize the power of the multi-instance technique in the sponge setting. As the bounds in Theorem 1, Theorem 2 and the general $O(ST^2/C + T^2/R)$ bound are the best one can prove using the multi-instance technique, other novel techniques are required to obtain optimal bounds for collision resistance of sponge in the AI setting. In Section 1.3, we point out potential techniques for future directions.

	Best known attacks	Previous Security bounds	Our Security bounds
$B = 1$	$\min(\frac{S^2T^2}{C^2}, (\frac{S^2T}{C^2})^{2/3}) + \frac{T^2}{R} + \frac{S}{C} + \frac{T}{C}$	$\frac{ST}{C} + \frac{T^2}{R}$ [FGK22]	$\frac{S^2T^2}{C^2} + \frac{S}{C} + \frac{T}{C} + \frac{T^2}{R}$ [Thm 1]
$B = 2$	$\frac{ST}{C} + \frac{T^2}{\min(C,R)}$	$\frac{ST}{C} + \frac{S^2T^4}{C^2} + \frac{T^2}{\min(C,R)}$ [FGK22]	$\frac{ST}{C} + \frac{S^2T^4}{C^2} + \frac{T^2}{\min(C,R)}$ [Thm 2]
$3 \leq B \leq T$	$\frac{STB}{C} + \frac{T^2}{\min(R,C)}$	$\frac{ST^2}{C} + \frac{T^2}{R}$ [CDG18]	-

Table 1: Asymptotic security bounds on the security of finding B-block-long collisions in sponge hash functions constructed from a random permutation $F : [R] \times [C] \mapsto [R] \times [C]$ against (S,T)-algorithms. For simplicity, logarithmic terms and constant factors are omitted and $S, T \geq 1$.

[6] [CDG18] proved an $\widetilde{O}(\frac{ST^2}{C} + \frac{T^2}{R})$ bound using presampling which implies an $(\widetilde{O}(\frac{ST^2}{C} + \frac{T^2}{R}))^S$ multi-instance security.

	Our attacks	Security bounds
$B=1$	$\left(\tilde{\Omega}\left(\frac{S^2T^2}{C^2}\right)\right)^S$ [Thm 10]	$\left(\tilde{O}\left(\frac{S^2T^2}{C^2}+\frac{S}{C}+\frac{T}{C}+\frac{T^2}{R}\right)\right)^S$ [Thm 8]
$B=2$	$\left(\tilde{\Omega}\left(\frac{S^2T^4}{C^2}\right)\right)^S$ [Thm 11]	$\left(\tilde{O}\left(\frac{ST}{C}+\frac{S^2T^4}{C^2}+\frac{T}{\min(C,R)}\right)\right)^S$ [Lemma 2]
$3\le B\le T$	$\left(\tilde{\Omega}\left(\frac{ST^2}{C}\right)\right)^S$ [Thm 12]	$\left(\tilde{O}\left(\frac{ST^2}{C}+\frac{T^2}{R}\right)\right)^S$

Table 2: Asymptotic bounds on the security finding B-block-long collisions in sponge hash functions constructed from a random permutation $F:[R]\times[C]\mapsto[R]\times[C]$ in the (S,T)-multi-instance games. We note that naive attacks can achieve $(\tilde{\Omega}(S/C))^S$, $(\tilde{\Omega}(T/C))^S$ and $(\tilde{\Omega}(T^2/R))^S$ advantage in (S,T)-MI games model.

1.2 Technical Overview

In this section, we present an overview of our proofs using reduction to the the multi-instance game model to analyze security bounds of B-block collision finding for $B=1$ and $B=2$, followed by our attacks for $B=1$, $B=2$ and $B\ge 3$ in the multi-instance game model.

The high level idea is: the multi-instance approach [AGL22, CGLQ20, IK10, ACDW20, GK22, FGK22] reduces proving the security of a problem with S-bit advice to proving the security of S random instances of the problem. If the instances are given at once, then we call it "parallel" multiple instance problem, and if the instances are presented one at a time, we call it"sequential" multi-instance game. [AGL22] showed that if any adversary (with no advice) can solve S random instances of the problem "sequentially" with success probability at most δ^S, then any adversary with S-bit advice can solve one instance of the problem with success probability at most 2δ. We note that security bounds for "parallel" multiple instance problem implies security bounds for corresponding "sequential" multiple instance games. Henceforth, we always mean sequential multi-instance games when we refer to multi-instance games in this paper.

Our proof for $B=1$. We use the compression technique from [DTT10] to analyze our multi-instance games. The compression technique (refer to Theorem 5 for the precise statement) states that for a pair of encoding and decoding algorithms that can compress a random function by at least $\log 1/\varepsilon$ bits, succeeds with probability at most ε. Here, we will design a pair of encoding and decoding algorithm, such that whenever an adversary $\mathcal{A}$ wins the multi-instance game, the encoder can use this adversary $\mathcal{A}$ to compress F. The challenge is to show that the encoder can compress 'enough' bits using this $\mathcal{A}$ to obtain the desired (upper) bound on the success probability of the adversary $\mathcal{A}$.

To get an idea, we first look at the simplest case. Say there is only 1 stage in the game (i.e., $S=1$), and the adversary makes two **forward** queries that collide for the challenge salt a. In other words the adversary queries $F(m_1,a)$ and $F(m_2,a)$ such that their outputs are in $(m,*)$. This means the first part of the outputs for both the queries is the same (which is m in this case). Here we can use 2 pointers, each $\log T$ bits long, to store the positions of the two colliding forward queries among the adversary's forward queries, and remove m from F's mapping table corresponding to the second query (Since we know it equals to the first part of output of the first query). This saves $\log R$ bits. Therefore, we can get an $O(T^2/R)$ upper bound as per theorem 5.

However, for $S > 1$, pointing to the forward colliding queries trivially requires $\log(ST)$-sized pointers (as the adversary makes a total of ST queries). This gives the bound $\varepsilon \leq S^2T^2/C$ which is not good enough. We can do better by storing the colliding queries for all the challenge salts together in an unordered set. The same idea works when the first occurring of the colliding queries is an inverse query and second one is a forward query. Refer to section 3.1.2 for more details about compressing in these cases.

Another possibility is that the adversary always outputs two **inverse** queries, say $F^{-1}(m_i, a_{i_1}) = (m_{i_1}, a_i)$ and $F^{-1}(m_i, a_{i_2}) = (m_{i_2}, a_i)$, as the collision. Then we can compress using that the second part of the output for all these queries will be in $a_1, \ldots, a_S$.

The trickiest case is when the adversary first makes a forward query, say $F(m_{i_1}, a_i) = (m_i, a_{i_1})$, then an inverse query, say $F^{-1}(m_i, a_{i_2}) = (m_{i_2}, a_i)$ as the collision. The trivial thing to do is to compress only the inverse query as above. However, it will only achieve an $O(ST/C)^S$ bound, which is not enough for our results. We use the idea that the output salt of the inverse query is not just in $a_1, \ldots, a_S$ but in fact it is one of the salts that is input to a forward query with output of the form $(m_i, *)$. The issue is the number of salts in $a_1, \ldots, a_S$ meeting this requirement could still be 'large'. When that happens we have to compress the output of the forward queries as well to get enough compression. Refer to section 3.1.2 for more details about this case and how to deal with an adversary that finds different types of collisions for different challenge salts.

Our proof for $B = 2$. For $B = 2$, we will use the proof strategy of Akshima et al. [AGL22] for dealing with $B = 2$ in the MD setting. The main difference is that we have to additionally deal with inverse queries in our analysis. We provide a high level overview of their proof, and describe where our proof differs due to inverse queries.

Recall that, to prove the sequential multi-instance security, it is sufficient to bound the advantage of any adversary that finds a 2-block collision for a fresh salt a, conditioned on it finds 2-block collisions for all the previous random challenge salts $a_1, \cdots, a_S$.

Following the terminology of Akshima et al. [AGL22], we call these ST queries made during the first S rounds as offline queries, and among the T queries made for a, we call the queries that were not made during the first S rounds as online queries. Moreover, we focus on the case that the new salt a has never been queried among the offline queries (because the other case happens with probability at most ST/N). As a result, all queries starting with the challenge salt a have to be online queries.

Akshima et al. [AGL22] studied how can the previous ST queries be helpful for this round of game? The main observation of Akshima et al. [AGL22] is that although the adversary learns about the function from the offline queries, and in the worst case, the offline queries could be very helpful. However, the helpful worst offline queries are not typical and can be tolerated by refining the technique. To do this, they define a bunch of helpful "high knowledge gaining" events among previous ST queries including, 1) there are more than S distinct salts with 1-block collision, 2) there are more than S^2 pairs of queries forming collisions, 3) there are more than S distinct salts with self-loops. They show that these events happen with sufficiently small probability, and conditioned on none of them happens, no online algorithms can find 2-block collisions with advantage better than the desired bound.

Now the question is what changes when inverse queries are allowed? The high knowledge gaining events are essentially the same, however some of these events can easily happen when inverse queries are allowed. In particular, for event 2), it was hard to form collisions (under the first part of output) among ST forward queries $F(x_1, y_1), \ldots, F(x_{ST}, y_{ST})$. However, if we make ST inverse queries with form $F^{-1}(0, y_1), \ldots, F^{-1}(0, y_{ST})$, then we have $\Omega((ST)^2)$ pairs of input pairs such that their evaluations in the forward direction form collision (under the first part of output). Given such a set of offline queries, one can find 2-block collisions for a new salt with probability at least $\Omega(S^2T^4/C^2)$. Fortunately, this is the worst we can get, and we can prove the advantage is at most $\mathcal{O}(S^2T^4/C^2)$ with adjusted high knowledge gaining events.

Our attacks for $B = 1, 2, 3$ in the MI model. We present three simple attacks for finding collisions in the multi-instance model and show their analysis. The main high level idea for all of these attacks is to accumulate relevant high knowledge events in each round to help with the next round.

We briefly illustrate the core idea of our attacks, starting with the attack for $B = 1$. In the ith round, the adversary makes T queries $F^{-1}(0, iT + j)$ for $j = 0, \ldots, T - 1$. The intuition is that via these inverse queries, the expected number of salts for which a collision is found (i.e. For a salt a, there exist two inverse queries $F^{-1}(0, x) = (m_1, a)$ and $F^{-1}(0, y) = (m_2, a)$) is $\Omega((iT)^2/C^2)$ in previous i-rounds. Therefore, once the random challenge salt in the i_{th} round is one of these 'solved' salt, then we are already done. Overall, the probability of finding collisions in each of the S rounds in this manner is at least $(\Omega(S^2T^2/C^2))^S$. We note that this is just the intuition, and we have to carefully deal with the correlations between winning in previous rounds and the expected events happening in previous rounds.

For $B = 2$, the most helpful event is to accumulate a lot of pairs of queries whose first part of output forms a collision. The best way of doing so is to spend an half of the queries in each round to make inverse queries of queries of form $F^{-1}(0, *)$, and spend the other half of the queries trying to hit two of these queries from the current challenge salt a_i. With high probability there will be $\Omega(i^2T^2/C^2)$ such pairs, and one can win the i_{th} stage with probability at least $\Omega(i^2T^4/C^2)$.

For $B = 3$, the most helpful event is to have at least $\Omega(iT)$ salts such that there are 2-block collisions starting from these salts. Specifically, we first try to find 1-block collision collisions starting from a salt y, and then make queries of form $F^{-1}(*, y)$ to generate these $\Omega(iT)$ salts. Then, with $\Omega(iT^2/C)$ probability one can hit one of these salts from the challenge salt and form a 3-block collision. We refer to Figure 7 and Section 5 for the details and analysis of these attacks.

1.3 Discussions and open problems

Is STB-conjecture true for sponge hashing? Akshima et al. [ACDW20] conjectured that the best attack with time T and space S for finding collisions of length $B \geq 2$ in salted MD hash functions built using compression functions with n-bit output achieves advantage $\Theta((STB + T^2)/2^n)$. It is natural to consider a similar STB-conjecture for sponge hash functions, conjecturing the $\Theta(STB/C + T^2/\min(R, C))$ attack by Freitag et al. [FGK22] is optimal for $B \geq 2$. However, this conjecture is only proved for very large $B \approx T$ [CDG18], and sponge hash is provably less secure than MD hash [FGK22] for $B = 1$. It will be extremely interesting to either prove or refute the sponge STB-conjecture. To start with, is the STB-conjecture true for $B = 2$ in sponge?

Better attacks for $B = 2$? The current security upper bound for $B = 2$ suggests that there may exist an attack with advantage $\Omega(S^2T^4/C^2)$ in the auxiliary-input random permutation model. And we show an attack in the multi-instance model with advantage $(\Omega(S^2T^4/C^2))^S$. Can we utilize similar ideas to show a corresponding attack in the auxiliary-input random permutation model?

Better bounds via stateless multi-instance games? Our results characterize the power of the multi-instance technique in the sponge setting by presenting attacks in the model of Akshima et al. [AGL22]. We observe that, a variant of the reduction of Akshima et al. [AGL22] allows one to consider more restricted multiple-instance games, where the adversary is stateless and doesn't remember information from previous rounds. Because our attacks require knowing queries from previous rounds, our attacks don't apply to stateless multi-instance games. We remark that analyzing stateless adversary for multi-instance games is non-trivial because, although the challenges are independent, the same random permutation is reused in multiple rounds. We hope that the study of stateless multi-instance games will shed light on how to obtain optimal bounds for finding collisions in sponge and potentially close the gap for MD other major open problem (such as function inversion) in this area.

Other related works. In a recent work [GGPS23], Golovnev et al. presented an algorithm for function inversion which works for any S, T such that $TS^2 \cdot \max\{S, T\} = \tilde{\Theta}(C^3)$ (where C is the size of the range of function) and improves over the Fiat and Noar algorithm when $S < T$. We mention that the time-space tradeoffs of many other cryptographic primitives, such as one-way functions, pseudorandom random generators, discrete discrete logarithm have been studied in various idealized models [DTT10, CHM20, CGK18, CGK19, GGKL21, DGK17, CDG18, CDGS18]. Recently, Ghoshal and Tessaro studied the pre-image resistance and collision-resistance security of preprocessing attacks with bounded offline and online queries for Merkle-Damgård construction in [GT23].

Acknowledgements

We thank TCC reviewers for their constructive comments. Siyao Guo and Akshima are supported by National Natural Science Foundation of China Grant No.62102260, Shanghai Municipal Education Commission (SMEC) Grant No. 0920000169, NYTP Grant No. 20121201 and NYU Shanghai Boost Fund. The work was done while Xiaoqi Duan was a research assistant at Shanghai Qi Zhi Institute and supported by the Shanghai Qi Zhi Institute. Most of the work was done while Qipeng Liu was a Postdoctoral researcher in Simons Institute, supported in part by the Simons Institute for Theory of Computing, through a Quantum Postdoctoral Fellowship and by the DARPA SIEVE-VESPA grant No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the supporting institutions.

2 Preliminaries

2.1 Notations

For any positive integer N, we write $[N]$ to denote the set $\{1, \dots, N\}$. For any non-negative integers N, k, $\binom{[N]}{k}$ is used to denote the collection of all k-sized subsets of $[N]$. For any finite set X, $\mathbf{x} \leftarrow_\$ X$ indicates $\mathbf{x}$ is a uniform random variable in X. We write X^+ to indicate a tuple of one or more elements from X.

2.2 Random Permutation Model

Random Permutation model is an idealized model where a function is modelled as a random permutation sampled uniformly from all possible permutations.

Lazy Sampling One useful property of modelling a function, say F, as a random permutation is that sampling F uniformly at random is equivalent to initializing F with $\perp$ for every input and sampling the responses uniformly at random without replacement as and when the input is queried.

2.3 Sponge Hash Functions

A cryptographic hash function is a function that takes input of arbitrary length and outputs a fixed length string. They are widely used in security applications such as digital signatures, message authentication codes and password hashing. In practice, several hash functions, including SHA-3, are based on the popular Sponge Construction.

A sponge based hash function internally uses a permutation function of fixed length domain. We will treat this permutation as a random permutation for the purpose of analyzing it's security.

We will parameterize our sponge function SP as a function in $[R]^+ \times [C] \to [R]$ such that it uses a random permutation, denoted by F, on $[R] \times [C]$ where $[R]$ corresponds to the set of messages and $[C]$ corresponds to the set of salts. Note that as F is a permutation, its inverse, denoted F^{-1}, is an efficiently computable function. Hence, any entity that can query F can also query F^{-1}.

Say $F(m, a) = (m', a')$ for some $m, m' \in [R]$ and $a, a' \in [C]$, then will use $F(m, a)[1], F(m, a)[2]$ to denote the first and second element from the output tuple. In other words, $F(m, a)[1] = m'$ and $F(m, a)[2] = a'$.

A message m is called a B-block message if it can be written as $m = m_1 || \dots || m_B$ where each $m_i \in [R]$. Then for a B-block message $m = m_1 || m_2 || \dots || m_B$ and some $a \in [C]$, we define the function $\mathsf{SP}_F(m, a)$ as follows:

1. Initialize $(x_0, y_0) = (0, a)$.
2. For the i^{th} block, compute $(x_i, y_i) = F(x_{i-1} \oplus m_i, y_{i-1})$.
3. Return x_B.

Collisions For a given $a \in [C]$, two distinct messages $m, m' \in [R]^+$ are said to form a **collision**, if

$$\mathsf{SP}_F(m, a) = \mathsf{SP}_F(m', a)$$

2.4 Definitions

We establish some definitions in this subsection which will be used throughtout the paper.

Definition 1. *We refer to two queries (m_1, a_1) and (m_2, a_2) as* ***same*** *or not distinct if one of the following holds true:*

1. *when both queries are made to F (or F^{-1}), $a_1 = a_2$ and $m_1 = m_2$*
2. *(m_1, a_1) is made to F (or F^{-1}), (m_2, a_2) is made to F^{-1} (or F) and $F(m_1, a_1) = (m_2, a_2)$ (or $F^{-1}(m_1, a_1) = (m_2, a_2)$)*

If two queries are not same, then they are referred to as ***distinct****.*

Next, we define an AI-adversary against collision resistance in Sponge functions.

Definition 2. *A pair of algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is an (S, T)-AI adversary for SP_F if*

- *$\mathcal{A}_1$ has unbounded access to F (and F^{-1}), and outputs S bits of advice, denoted σ*
- *$\mathcal{A}_2$ takes σ and a challenge salt $a \in [C]$ as input, makes T queries to F or F^{-1}, and outputs m, m'.*

Next, we define the security game for B-block collision-resistance against the (S, T)-AI adversary.

Definition 3. *For any fixed random permutation $F : [R] \times [C] \to [R] \times [C]$, a salt $a \in [C]$ and B which is a function of R, C, we define the game $\mathsf{B\text{-}AICR}$ in fig. 1.*

```
Game $\mathsf{B\text{-}AICR}_{\mathsf{F,a}}(\mathcal{A})$
    $\sigma \leftarrow \mathcal{A}_1^F$
    $m, m' \leftarrow \mathcal{A}_2^{F/F^{-1}}(\sigma, a)$
    If $m$ or $m'$ consists of more than $B$ blocks
        Then Return 0
    If $m \neq m'$ and $\mathsf{SP}_F(m, a) = \mathsf{SP}_F(m', a)$
        Then Return 1
    Else Return 0
```

Fig. 1: Security game $\mathsf{B\text{-}AICR}_{\mathsf{F,a}}(\mathcal{A})$

For any (S, T)-AI adversary $\mathcal{A}$, its advantage is denoted by $\mathsf{Adv}^{\mathsf{AICR}}_{\mathsf{B\text{-}SP}}(\mathcal{A})$ and defined as the probability that for a uniformly random permutation F and a random salt $a \in [C]$, the game $\mathsf{B\text{-}AICR}_{\mathsf{F,a}}(\mathcal{A})$ returns 1. For any functions S, T, B, we define (S, T, B)-auxiliary input collision resistance of Sponge functions, denoted by $\mathsf{Adv}^{\mathsf{AICR}}_{\mathsf{B\text{-}SP}}(\mathsf{S}, \mathsf{T})$, as the maximum advantage taken over all (S, T)-AI adversaries.

It is known from several prior works that security in the AI model is closely related to the security in the multi-instance model. In this work, we will analyze the security in the MI model and use this relation to obtain security bounds in the AI model. To this end, we formally define the multi-instance (MI) adversary and two versions of the security game for collision resistance against the MI adversary next.

Definition 4. *A stateful algorithm $\mathcal{A}$ is an (S,T)-MI adversary against collision resistance in SP_F if for every $i \in [S]$:*

- *$\mathcal{A}$ takes a salt $a_i \in [C]$ as input*
- *$\mathcal{A}$ makes T queries to F or F^{-1}*
- *$\mathcal{A}$ outputs m_i, m_i'.*

We define the security games next.

Definition 5. *For any fixed random permutation $F : [R] \times [C] \to [R] \times [C]$, fixed B and S that are functions of R, C, we define the game $\mathsf{B\text{-}MICR^S}$ in fig. 2.*

```
Game $\mathsf{B\text{-}MICR^S_F}(\mathcal{A})$
    For $i \in [S]$:
        Sample $a_i \leftarrow [C]$ at uniformly random without replacement
        $m_i, m_i' \leftarrow \mathcal{A}^{F/F^{-1}}(a_i)$
        If $m_i$ or $m_i'$ consists of more than $B$ blocks
            Then Return 0
        If $m_i = m_i'$ or $\mathsf{SP}_F(m_i, a_i) \neq \mathsf{SP}_F(m_i', a_i)$
            Then Return 0
    Return 1
```

Fig. 2: Security game $\mathsf{B\text{-}MICR^S_F}(\mathcal{A})$

For any (S,T)-MI adversary $\mathcal{A}$, its advantage is denoted by $\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{B\text{-}SP}}(\mathcal{A})$ and defined as the probability that for a uniformly random permutation F, the game $\mathsf{B\text{-}MICR^S_F}(\mathcal{A})$ returns 1. For any functions S, T, B, we define (S,T,B)-multi-instance collision resistance of Sponge functions, denoted by $\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{B\text{-}SP}}(\mathsf{S},\mathsf{T})$, as the maximum advantage taken over all (S,T)-MI adversaries.

Next, we define another game for the MI adversary and it is differs from the one above only in the way it samples salts uniformly at random with replacement.

Definition 6. *For any fixed random permutation $F : [R] \times [C] \to [R] \times [C]$, fixed B and S that are functions of R, C, we define the game $\mathsf{B\text{-}rand\text{-}MICR^S}$ in fig. 3.*

$\mathsf{Adv}^{\mathsf{rand\text{-}MICR}}_{\mathsf{B\text{-}SP}}(\mathcal{A})$ and $\mathsf{Adv}^{\mathsf{rand\text{-}MICR}}_{\mathsf{B\text{-}SP}}(\mathsf{S},\mathsf{T})$ are analogously defined for (S,T)-MI adversaries as in def. 5.

2.5 Relevant Results

We will use the following theorems in our proofs:

Theorem 4 (Chernoff bounds). *Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_t$ are independent random variables. Let $\mathbf{x} = \sum_{i=1}^{t} \mathbf{x}_i$ and $\mu = \mathbb{E}[\mathbf{x}]$. For any $\delta \geq 0$, it holds that*

$$\Pr[\mathbf{x} \geq (1+\delta)\mu] \leq \exp\left(\frac{-\delta^2\mu}{2+\delta}\right).$$

```
Game $\mathsf{B\text{-}rand\text{-}MICR}_{\mathsf{F}}^{\mathsf{S}}(\mathcal{A})$
    For $i \in [S]$:
        Sample $a_i \leftarrow_{\$} [C]$
        $m_i, m_i' \leftarrow \mathcal{A}^{F/F^{-1}}(a_i)$
        If $m_i$ or $m_i'$ consists of more than $B$ blocks
            Then Return 0
        If $m_i = m_i'$ or $\mathsf{SP}_F(m_i, a_i) \neq \mathsf{SP}_F(m_i', a_i)$
            Then Return 0
    Return 1
```

Fig. 3: Security game $\mathsf{B\text{-}rand\text{-}MICR}_{\mathsf{F}}^{\mathsf{S}}(\mathcal{A})$

Theorem 5 ([DTT10]). *For any pair of encoding and decoding algorithms (Enc, Dec), where $Enc : \{0,1\}^x \to \{0,1\}^y$ and $Dec : \{0,1\}^y \to \{0,1\}^x$, such that $Dec(Enc(z)) = z$ with probability at least ϵ where $z \leftarrow \{0,1\}^x$, then $y \geq x - \log \frac{1}{\epsilon}$.*

This is the compression theorem we mentioned earlier, and we will use it frequently in our proofs.

Theorem 6. *For any S, T, B and $\delta \in [0,1]$, if $\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{rand\text{-}MICR}}(\mathsf{S},\mathsf{T}) \leq \delta^{\mathsf{S}}$, then $\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{AICR}} \leq 2\delta$.*

Such a theorem relating AI-security to MI-security has been used in several prior works. Refer to theorem 3 in [AGL22] for more details and proof.

Theorem 7. *For any S, T, B, C, if $\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{MICR}}(\mathsf{u},\mathsf{T}) \leq \delta^{\mathsf{u}}$ for all $u \in [S]$, then $\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{rand\text{-}MICR}}(\mathsf{S},\mathsf{T}) \leq (\delta + \frac{\mathsf{S}}{\mathsf{C}})^{\mathsf{S}}$.*

Proof. Fix an arbitrary (S,T)-MI adversary $\mathcal{A}$. Let X_i be an indicator that $\mathcal{A}$ wins on the i^{th} instance, i.e., finds B-block collision on a_i (i^{th} challenge salt). Then

$$\begin{aligned}\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{rand\text{-}MICR}}(\mathcal{A}) &= \Pr[X_1 \wedge \cdots \wedge X_S] = \prod_{i=1}^{S} \Pr[X_i | X_{<i}] \\ &\leq \prod_{i=1}^{S} (\Pr[a_i = a_j \text{ for some } j < i] + \Pr[X_i | X_{<i} \wedge a_i \neq a_j \text{ for all } j < i]) \\ &= \prod_{i=1}^{S} \left(\frac{i-1}{C} + \delta \right) \leq \left(\frac{S}{C} + \delta \right)^S\end{aligned}$$

where the third equality follows from the fact that $\mathsf{Adv}_{\mathsf{B\text{-}SP}}^{\mathsf{MICR}}(\mathsf{u},\mathsf{T}) \leq \delta^{\mathsf{u}}$ for every $u \in [S]$. □

This theorem relates the advantage of an MI-adversary that receives random challenge salts (which can possibly repeat) to an MI-adversary that always receives distinct challenge salts.

3 Improved Bound for $B = 1$ (Optimal when $ST^2 \leq C$)

Theorem 8. *For any S, T, R, C such that $S \geq 2^{12}, \max\{\frac{ST}{C}, \frac{T^2}{R}\} < \frac{1}{8e^9}$, then*

$$\mathsf{Adv}^{\mathsf{AICR}}_{\mathsf{1\text{-}SP}}(\mathsf{S}, \mathsf{T}) = \widetilde{\mathsf{O}}\left(\frac{\mathsf{T}^2}{\mathsf{R}} + \frac{\mathsf{S}}{\mathsf{C}} + \frac{\mathsf{T}}{\mathsf{C}} + \left(\frac{\mathsf{ST}}{\mathsf{C}}\right)^2\right)$$

To prove this theorem, we first reduce AI-security to MI-security via Theorem 6 and Theorem 7. Now we only need to look at the advantage bound for any MI-adversary $\mathcal{A}$ in game $\mathsf{B\text{-}MICR}^{\mathsf{S}}_{\mathsf{F}}(\mathcal{A})$. In this game, all the challenge salts $a_1, \ldots, a_S$ are different, and unlike the AI model, here the adversary doesn't have any advice (except the information from previous stages).

In fact, we have the following lemma, which we will prove in the next section:

Lemma 1. *For any S, T, R, C such that $S \geq 2^{12}, \max\{\frac{ST}{C}, \frac{T^2}{R}\} < \frac{1}{8e^9}$, then*

$$\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{1\text{-}SP}} = \left(\widetilde{\mathsf{O}}\left(\frac{\mathsf{T}^2}{\mathsf{R}} + \frac{\mathsf{T}}{\mathsf{C}} + \left(\frac{\mathsf{ST}}{\mathsf{C}}\right)^2\right)\right)^{\mathsf{S}}$$

Now we can prove the main theorem.

Proof of Theorem 8. The Theorem is immediate from Theorem 6, Theorem 7 and Lemma 1. $\square$

3.1 Proof of Lemma 1

We will prove the lemma via compression. In section 3.1.1 and 3.1.2, we will design a set of encoding and decoding algorithms, such that given any MI-adversary $\mathcal{A}$, if $\mathcal{A}$ wins on 'too many' S-sized subsets of $[C]$, we can use fewer bits to describe the function F (i.e. compress it). However, this contradicts with Theorem 5 and thus bounds the number of S-sized subsets of $[C]$ any adversary can succeed on. In section 3.1.3, we will first analyze the number of bits our algorithm can save, and then finally prove Lemma 1 via compression argument. To this end, we first identify the types of 1-block collisions.

3.1.1 Type of Collisions

Due to the existence of F^{-1}, there are 3 possible types of collisions when $B = 1$. A pair of collision (m_1, m_2) on salt a, such that $F(m_1, a) = (m, a_1)$ and $F(m_2, a) = (m, a_2)$, can be classified into cases according to the type and relative position of the corresponding "fresh" queries as made by the adversary.

Here a query $F(m_1, a) = (m, a_1)$ is "fresh" means that neither $F(m_1, a)$ nor $F^{-1}(m, a_1)$ has been queried previously. A query to F is referred to as a forward query and a query to F^{-1} is referred to as an inverse query.

WLOG we assume $F(m_1, a)$ or $F^{-1}(m, a_1)$ is queried for the first time before $F(m_2, a)$ or $F^{-1}(m, a_2)$ is queried for the first time. Then the 3 types of collisions are as follows:

- Type 1: The second fresh query is of the form $F(m_2, a)$, i.e. the first fresh query from the collision can be either a forward query or an inverse query but the second one is a forward query.
- Type 2: The fresh queries are of the form $F^{-1}(m, a_1)$ and $F^{-1}(m, a_2)$, i.e. both fresh queries are inverse queries (that queries F^{-1}).
- Type 3: The fresh queries are of the form $F(m_1, a)$ and $F^{-1}(m, a_2)$ (i.e. the first fresh query is a forward query, and the second fresh query is an inverse query).

See figure 4 for details.

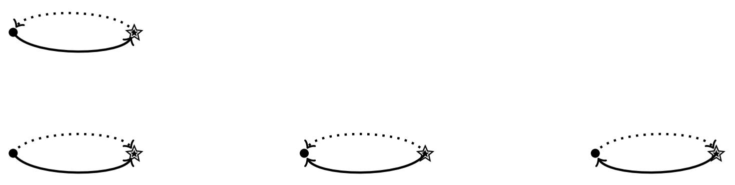

(a) Type 1 **(b)** Type 2 **(c)** Type 3

Fig. 4: All 3 types of collisions. Line directed $\rightarrow$ represents query to F and line directed $\leftarrow$ represents query to F^{-1}. ● represents that (output/input of) the two queries share the salt. ★ represents that (output/input of) the two queries share the message. The dotted line means this query occurs first, and the solid line means this query occurs later than the dotted line query.

3.1.2 Encoding and Decoding Algorithms

Now we state our encoding and decoding algorithms for the random permutation function $F : [R] \times [C] \rightarrow [R] \times [C]$. Let $\mathcal{A}$ be an (S, T)-MI adversary that wins the game, i.e. succeeds on S different challenge salts $a_1, \ldots, a_S$. Generally, in the encoding algorithm, we will store a partial mapping table of F that contains answers to all the queries made by $\mathcal{A}$ in order, except the entries (corresponding to the mappings) we delete from the table, followed by the remaining of the function table (not queried by $\mathcal{A}$) in the lexicographic order. Note that we will store some extra bits apart from the function table that will help recover the deleted entries from the table. In the decoding algorithm, we will restore the removed entries using these extra bits, and thus restore the entire function table.

The encoding algorithm is as follows:

1. Simulate $\mathcal{A}$ on F and $a_1, \ldots, a_S$, and maintain a table that contains the response to each query made by $\mathcal{A}$ in order (e.g. If the first query made by the adversary is $F(m', a') = (m, a)$, then the first entry will be (m, a)).
2. As $\mathcal{A}$ succeeds on $a_1, \ldots, a_S$, it is guaranteed to output S pairs of messages that form collision with each of $a_1, \ldots, a_S$ under SP_F in the simulation. According to the collision types in section 3.1.1, divide the salts into 3 sets $\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3$, respectively. Denote $S_i = |\mathbb{S}_i|$ for $i = 1, 2, 3$, i.e. S_i is the size for set $\mathbb{S}_i$.
3. For salts in set $\mathbb{S}_1$:
   (a) Use $\log S$ bits to store S_1 (i.e. the number of salts in $\mathbb{S}_1$).

   (b) Use $2\log\binom{ST}{S_1}$ bits to store in an unordered fashion the set containing indices of **fresh** queries that form collisions for each salt in $\mathbb{S}_1$. (Since for each salt there are exactly 2 queries that form the collision, and there are ST possible locations for each such query, we need $2\log\binom{ST}{S_1}$ to store this set.)
   (c) For each pair of collision $F(m_{i_1}, a_i) = (m_i, a')$ (or $F^{-1}(m_i, a') = (m_{i_1}, a_i)$) and $F(m_{i_2}, a_i) = (m_i, a'')$ in the order of occurrence, remove m_i in the output of the second query (i.e. the first part of output), but keep the second part. Namely, we remove the the response to $F(m_{i_2}, a_i)$ in the function table, but use extra $\log C$ bits to store the second part, i.e. a''. So after this step, we remove S_1 entries from the function table, and stored the size of $\mathbb{S}_1$ in $\log S$ bits, the set in $2\log\binom{ST}{S_1}$ bits and second part of the S_1 entries in $S_1\log C$ bits.
4. For salts in set $\mathbb{S}_2$:
   (a) Use $\log S$ bits to store S_2.
   (b) Use $2\log\binom{ST}{S_2}$ bits to store the set of indices of **fresh** queries that form collision for each salt in $\mathbb{S}_2$.
   (c) For each stored query $F^{-1}(m, a') = (m', a_i)$, use $\log S$ bits to store the salt index i (since a_i is among $a_1, \dots, a_S$), and remove a_i from the response of the query. (Still, it means that we remove the corresponding entry in the function table, and use $\log R$ extra bits to store m'.)
5. For salts in set $\mathbb{S}_3$, depending on the parameters S, T, R, there are two corresponding strategies of compression:
   (a) When $ST > R$:
      i. Use $\log S$ bits to store S_3.
      ii. Use $2\log\binom{ST}{S_3}$ bits to store the positions of fresh queries which form the collision for each salt $a_i \in \mathbb{S}_3$. (There will be S_3 inverse queries and S_3 forward queries.)
      iii. For each stored inverse query $F^{-1}(m_i, a') = (m_{i_2}, a_i)$, count the number of **stored** forward query $F(m_{j_1}, a_j) = (m_j, a'')$ in previous step, such that $j < i$ and $m_j = m_i$. Denote this number by q_{m_i}. Then we use extra $\log q_{m_i}$ bits to indicate which forward query is exactly the matching query for the current inverse query (Since the matching query must be within these q_{m_i} stored queries). Finally, remove the second part of output of these S_3 inverse queries (i.e. remove a_i).
      iv. For $m \in [R]$, let $Q(m)$ be the set of **stored** forward queries $F(m_{i_1}, a_i) = (m_i, a')$ in step 5(a)ii where $m_i = m$. Denote $Q_m = |Q(m)|$, i.e. the size of $Q(m)$. Notice all the values q_m defined in step 5(a)iii also satisfy $q_m \le Q_m$ (since q_m only counts the previous queries).
      Now, if there exists m such that $Q_m > \log STR$, we will use extra bits to save information about these forward queries (since their first part of output are all m):
         A. Use $\log S$ bits to indicate the number of m such that $Q_m > \log STR$.
         B. For each such m:
            – Use $\log R$ bits to store m.
            – Use $\log S_3$ bits to store Q_m. (Since all the elements in $Q(m)$ are from stored forward queries, the size is at most S_3.)
            – Use $\log\binom{S_3}{Q_m}$ bits to store the positions of queries in $Q(m)$ (among stored forward queries in step 5(a)ii), i.e. the queries with first part of output being m.

  - Remove the first part of output of all these Q_m queries (i.e. remove m in the output).

  For convenience, we denote $Q_{tot} = \sum_{m \in [R]: Q_m > \log STR} Q_m$.

(b) When $ST \le R$:
  i. Use $\log S$ bits to store S_3.
  ii. Use $\log \binom{ST}{S_3}$ bits to store the positions of fresh **inverse** queries which find the collision for each salt $a_i \in \mathbb{S}_3$. (There will be S_3 inverse queries.) Notice we no longer store forward queries.
  iii. For each stored inverse query $F^{-1}(m_i, a') = (m_{i_2}, a_i)$, count the number of **all** forward queries $F(m_{j_1}, a_j) = (m_j, a'')$ such that $j < i$ and $m_j = m_i$. Denote this number be q_{m_i}. Then we use extra $\log q_{m_i}$ bits to indicate which forward query is exactly the matching query for the current inverse query. (Notice q_{m_i} can be as large as ST.) Finally, remove the second part of output of these S_3 inverse queries (i.e. remove a_i).
  iv. For $m \in [R]$, denote $Q(m)$ be the set of **all** queries $F(m_{i_1}, a_i) = (m_i, a')$ (or $F^{-1}(m_i, a') = (m_{i_1}, a_i)$) such that $m_i = m$, and Q_m be the size of $Q(m)$. Further, denote K_m be the number of **stored** inverse queries $F^{-1}(m_i, a'') = (m_{i_2}, a_i)$ in step 5(b)ii such that $m_i = m$. Obviously $K_m \le Q_m$. Further, all the values q_m occurred in step 5(b)iii also satisfy $q_m \le Q_m$ (since q_m only counts the previous queries).

  Now, if there exists m such that $Q_m > \log STR$ and $K_m \ge 1$, we will use extra bits to save information about these forward queries (since their first part of output are all m):
    A. Use $\log S$ bits to indicate the number of m such that $Q_m > \log STR$.
    B. For each such m:
      - Use $\log R$ bits to store m.
      - Use $\log ST$ bits to store Q_m. (Notice now Q_m can be as large as ST.)
      - Use $\log \binom{ST}{Q_m}$ bits to store the indices of the queries whose first part of response is m. (Notice since we don't store the forward queries in step 5(b)ii, here we need $\log \binom{ST}{Q_m}$ instead of $\log \binom{S_3}{Q_m}$.)
      - Remove the first part of the response to all these Q_m queries (i.e. remove m in the output).

  For convenience, we denote $Q_{tot} = \sum_{m \in [R]: Q_m > \log STR, K_m \ge 1} Q_m$, and $K_{tot} = \sum_{m \in [R]: Q_m > \log STR, K_m \ge 1} K_m$. Since $K_m \le Q_m$ for any m, we have $K_{tot} \le Q_{tot}$.

6. The final output of the encoding algorithm will be the extra bits generated in steps 3-5 (except step 5(a)iii and 5(b)iii), followed by the function table (with some entries deleted) that contains responses of the queries, followed by the remaining function table of F that remains unqueried by the adversary, followed by the extra bits in step 5(a)iii and 5(b)iii in the order that the corresponding query is made by the adversary.

Next, we state our decoding algorithm which can fully recover the random permutation function F (for a succeeding adversary).

1. Read out the first part of generated extra bits according to encoding step 3-5. For example, for step 3, we first read S_1, then according to S_1 we know the value of $2\log \binom{ST}{S_1}$,

so we can continue reading out the indices of the stored queries, etc. After this step we will know the total number of removed queries in the table, and thus we know the starting point of the second part of extra bits.

2. Now simulate the adversary $\mathcal{A}$ on the given salts $a_1, \ldots, a_S$. During the process, $\mathcal{A}$ will make several oracle queries to either F or F^{-1}, and we know from the extra bits whether we have removed the corresponding entry in the (partial) function table or not.
   – If the query is not removed from the function table, we simply read the next entry from the table and answer it directly.
   – Otherwise, it must have been deleted in one of the encoding steps 3-5. Formally, we must have stored information about this query in either step 3b, 4b, 5(a)ii, 5(a)ivB, 5(b)ii or 5(b)ivB.
     • It is stored in encoding step 3b. Thus it must be a forward query with form $F(m_0, a) = (?, a')$ (we know a' since we have stored it when encoding, but we don't know the first part). Then we look at the whole stored query set in step 3b. Since all challenge salts are different, we can uniquely determine its corresponding first occurring query $F(m_1, a) = (m, a')$ or $F^{-1}(m, a') = (m_1, a)$ according to a . Hence, we know $F(m_0, a) = (m, a')$.
     • It is stored in encoding step 4b. Thus it must be an inverse query of the form $F^{-1}(m, a) = (m', ?)$. From the stored index i we immediately know $F^{-1}(m, a) = (m', a_i)$.
     • It is stored in encoding step 5(a)ii or 5(b)ii. Thus it must be an inverse query of the form $F^{-1}(m, a_0) = (m', ?)$. Further, we also know a $\log q_m$ pointer to its corresponding forward query $F(m_1, a) = (m, a_1)$ or $F^{-1}(m, a_1) = (m_1, a)$. (Since we already know all the previous queries, we can easily recover the set $Q(m)$ and know the value q_m, and then we can read the next $\log q_m$ bits from the second part of extra bits, and determine the specific query.) Then we know $F^{-1}(m, a_0) = (m', a)$.
     • It is stored in encoding step 5(a)ivB or 5(b)ivB. Then it must be a forward query of the form $F(m', a') = (?, a'')$. Since we have stored the corresponding m for this query, we know $F(m', a') = (m, a'')$.
3. Continue reading out the function table of F that are not queried by the adversary.

Therefore, as long as the adversary $\mathcal{A}$ can successfully find all the S collisions, we can correctly restore the the function F.

3.1.3 Number of Bits Saved

In this section, we will analyze the number of bits we can compress using our encoding algorithm, and then prove Lemma 1.

In our algorithm, the compression comes from deleting several entries of the function table of F and storing some extra (lesser number of) bits for learning those deleted entries. The following claim shows the compression from removing these entries:

Claim 1. *Suppose in the encoding algorithm we removed y entries from the function table, and stored x extra bits for information. Then we can save at least $y(\log RC - 1) - x$ bits via the compression.*

Proof. In order to count the saving, we first need to know how many bits are needed to store these rows originally. Note that the adversary $\mathcal{A}$ only queries at most ST values of F, and touches at most $2ST$ (message,salt) pairs (either in the input of the query, or from output of the query), so there will be at least $RC - 2ST \geq \frac{RC}{2}$ untouched pairs. Therefore, for each query, there are at least $\frac{RC}{2}$ different possible outputs (since the output can be any of the untouched pairs), and we need at least $\log RC - 1$ bits to store these entries. Since it's true for any query, we originally need at least $y(\log RC - 1)$ to store all these removed entries, so we can save at least $y(\log RC - 1) - x$ bits. □

Next, we analyze the savings according to the types of collisions:

– In step 3, we stored $\log S + 2\log\binom{ST}{S_1} + S_1 \log C$ extra bits. Besides, we deleted S_1 entries from the function table. Therefore, according to the above claim, we can save at least

$$\begin{aligned} L_1 :=& S_1(\log RC - 1) - \log S - 2\log\binom{ST}{S_1} - S_1\log C \\ =& \, S_1(\log R - 1) - \log S - 2\log\binom{ST}{S_1} \end{aligned}$$

bits.

– In step 4, we stored $\log S + 2\log\binom{ST}{S_2} + 2S_2\log S + 2S_2\log R$ bits, and deleted $2S_2$ entries from the function table. Then we can save at least

$$L_2 := 2S_2(\log C - 1) - \log S - 2S_2\log S - 2\log\binom{ST}{S_2}$$

bits.

– In step 5:
  - When $ST > R$, we first stored $\log S + 2\log\binom{ST}{S_3} + \log S$ bits, and for each query we stored an $\log q_{m_i}$ bit pointer. Further, for each large Q_i, we stored $\log R + \log S_3 + \log\binom{S_3}{Q_i}$ bits. What we saved is the second part of output of S_3 inverse queries, which is at least $S_3(\log C - 1)$ bits, and the first part of the response to the queries in Q_i, which is at least $Q_i(\log R - 1)$ bits. Therefore, we can save at least

$$\begin{aligned} L_3 :=& S_3(\log C - 1) + \sum_{i:Q_i > \log STR} Q_i(\log R - 1) - 2\log S \\ & - 2\log\binom{ST}{S_3} - \sum_{i:Q_i>\log STR}\left(\log S_3 + \log\binom{S_3}{Q_i} + \log R\right) - \sum_i Q_i \log Q_i \end{aligned}$$

  bits.
  - When $ST \leq R$, similarly we can save at least

$$\begin{aligned} L_3 :=& S_3(\log C - 1) + \sum_{i:Q_i > \log STR, K_i \geq 1} Q_i(\log R - 1) - 2\log S \\ & - \log\binom{ST}{S_3} - \sum_{i:Q_i>\log STR, K_i\geq 1}\left(\log ST + \log\binom{ST}{Q_i} + \log R\right) - \sum_i K_i \log Q_i \end{aligned}$$

  bits.

Based on the above analysis, we have the following claims:

Claim 2. *Suppose $S \geq 2^{12}, \max\{\frac{ST}{C}, \frac{T^2}{R}\} < \frac{1}{8e^9}$. When $ST > R$:*

$$2^{L_1+L_2+L_3} = \left(\widetilde{O}\left(\frac{T^2}{R} + \left(\frac{ST}{C}\right)^2\right)\right)^S$$

Claim 3. *Suppose $S \geq 2^{12}, \max\{\frac{ST}{C}, \frac{T^2}{R}\} < \frac{1}{8e^9}$. When $ST \leq R$:*

$$2^{L_1+L_2+L_3} = \left(\widetilde{O}\left(\frac{T^2}{R} + \left(\frac{ST}{C}\right)^2 + \frac{T}{C}\right)\right)^S$$

The proof for these claims have been deferred to the appendix A.

Proof of Lemma 1. For any adversary $\mathcal{A}$, denote $\epsilon = \mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{1\text{-}SP}}(\mathcal{A})$. According to Theorem 5 , Claim 2 and Claim 3:

$$\begin{aligned}\log \epsilon &\leq L_1 + L_2 + L_3\\ \epsilon &\leq 2^{L_1+L_2+L_3}\\ &\leq \left(\widetilde{O}\left(\frac{T^2}{R} + \left(\frac{ST}{C}\right)^2 + \frac{T}{C}\right)\right)^S\end{aligned}$$

Since it holds for any $\mathcal{A}$, according to definition of $\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{1\text{-}SP}}$, this completes the proof. □

4 A simpler proof for $B = 2$

In this section we analyze the lower bound for 2-block collisions in Sponge hash functions in the AI model via reduction to MI model. Our bound in the MI model matches the attack in the MI model (refer to section 5.2).

Theorem 9. *For any S, T, C and R,*

$$\mathsf{Adv}^{\mathsf{AICR}}_{\mathsf{2\text{-}SP}}(\mathsf{S},\mathsf{T}) = \widetilde{\mathsf{O}}\left(\left(\frac{\mathsf{ST}^2}{\mathsf{C}}\right)^2 + \frac{\mathsf{ST}}{\mathsf{C}} + \frac{\mathsf{T}^2}{\mathsf{C}} + \frac{\mathsf{T}^2}{\mathsf{R}}\right).$$

From theorems 6 and 7, we know it suffices to prove the following lemma in order to prove theorem 9.

Lemma 2. *For any S, T, C and R,*

$$\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{2\text{-}SP}}(\mathsf{S},\mathsf{T}) = \left(\widetilde{\mathsf{O}}\left(\left(\frac{\mathsf{ST}^2}{\mathsf{C}}\right)^2 + \frac{\mathsf{ST}}{\mathsf{C}} + \frac{\mathsf{T}^2}{\mathsf{C}} + \frac{\mathsf{T}^2}{\mathsf{R}}\right)\right)^{\mathsf{S}}.$$

Proof of Lemma 2. We assume F is a random permutation from $[R] \times [C] \to [R] \times [C]$ which is lazily sampled. Fix an arbitrary (S, T)-MI adversary $\mathcal{A}$.

Let X_i be the indicator that $\mathcal{A}$ wins on the i^{th} instance, i.e., finds 2-block collision on a_i. Then we make the following claim:

Claim 4.

$$\Pr[X_1 \wedge \cdots \wedge X_S] \leq \prod_{i=1}^{S} \left(\Pr[X_i | X_{<i} \wedge \overline{E^i}] + \frac{\Pr[E^i]}{\Pr[X_{<i}]} \right)$$

for any event E^i.

Proof of Claim 4.

$$\begin{aligned}
\Pr[X_1 \wedge \cdots \wedge X_S] &= \prod_{i=1}^{S} \Pr[X_i | X_{<i}] = \prod_{i=1}^{S} \frac{\Pr[X_{\leq i}]}{\Pr[X_{<i}]} \\
&= \prod_{i=1}^{S} \left(\frac{\Pr[X_{\leq i} \wedge \overline{E^i}]}{\Pr[X_{<i}]} + \frac{\Pr[X_{\leq i} \wedge E^i]}{\Pr[X_{<i}]} \right) \\
&= \prod_{i=1}^{S} \left(\frac{\Pr[X_i | X_{<i} \wedge \overline{E^i}] \cdot \Pr[X_{<i} \wedge \overline{E^i}]}{\Pr[X_{<i}]} + \frac{\Pr[X_{\leq i} \wedge E^i]}{\Pr[X_{<i}]} \right) \\
&\leq \prod_{i=1}^{S} \left(\Pr[X_i | X_{<i} \wedge \overline{E^i}] + \frac{\Pr[E^i]}{\Pr[X_{<i}]} \right)
\end{aligned}$$

□

Say we want to bound $\Pr[X_1 \wedge \cdots \wedge X_S]$ by δ^S, then it is sufficient to bound $\Pr[X_i | X_{<i}]$ by δ for an arbitrary $i \in [S]$. As in [AGL22], when analyzing $\Pr[X_i | X_{<i}]$, we will refer to the stage before receiving the i^{th} challenge salt a_i as the offline stage and the stage after receiving a_i as online stage.

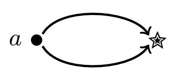
Definition 7. ***Database*** *is defined as the set of sampled distinct queries on F/F^{-1} and their responses.*
The set of distinct queries made in the offline stage (i.e., before receiving the challenge salt as input) are referred to as ***offline queries****. The set of distinct queries made in the online stage (i.e., after receiving the challenge salt as input) that had not been made in the offline phase are referred to as the* ***online queries****.*

Following the claim 4, our high-level strategy would be to define the 'good' events and then bound the two terms separately to obtain our results. It is worth noting that we can assume $\Pr[X_{<i}] \geq \delta^i$. Otherwise $\Pr[X_{<i+1}] \leq \delta^i$ holds trivially.

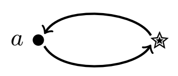
To this end we first define the 'good' events. For $j \in [4]$, let's define E^i_j to be the event that there exists at least $10i \log R$ of Type j structures (shown in fig. 5) from distinct a in the offline queries of i^{th} instance. Let's define the event $E^i := E^i_1 \vee E^i_2 \vee E^i_3 \vee E^i_4$.

Next, we analyze the probability of the events E^i_j for all $j \in [4]$.

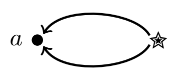
Claim 5. *For any $i \in [S]$ and $T^2 \leq R/2$, $\Pr[E^i_1] \leq R^{-10i}$.*



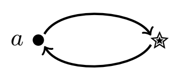
(a) Type 1



(b) Type 2



(c) Type 3



(d) Type 4

Fig. 5: The top line represents the query that occurs first, and the bottom line represents the query that occurs later. $\rightarrow$ represents a query to F and $\leftarrow$ represents a query to F^{-1}. ● represents that (output/input of) the two queries share the salt. ✯ represents that (output/input of) the two queries share the message.

Proof. For simplicity, we prove the claim via compression. Before we present the encoding algorithm, recall that for event E_1^i to happen, there should exist at least $10i \log R$ pair of offline queries where queries in each pair have the same output message and the same input salt. Also no two pairs share the same input salt. So, if the encoder stores the indices of the $20i \log R$ queries as an unordered pair, it is possible for the decoder to identify the pairs via the shared input salt which is unique to the pair. Our encoding can compress the output message of the second query in every pair.

We give a formal description of the encoding algorithm next.

- Store the $20i \log R$ offline queries that form the $10i \log R$ Type 1 structures as an unordered set, say W. This would require $\log \binom{iT}{20i \log R}$ bits.
- Delete the output message of the second occurring queries from each pair, each at least $(\log R - 1)$ bits long, in the unordered set W from the table of sampled queries on F/F^{-1}.

Then

$$\begin{aligned}\Pr[E_1^i] &= \Pr[\exists \geq 10i \log R \text{ of Type 1 structures from distinct } a] \\ &\leq \Pr[\exists \text{ exactly } 10i \log R \text{ of Type 1 structures from distinct } a] \\ &\leq \frac{\binom{iT}{20i \log R}}{(R/2)^{10i \log R}} \leq \left(\frac{2e^2 i^2 T^2}{400 i^2 \log^2 R \cdot R}\right)^{10i \log R} \leq 2^{-10i \log R}\end{aligned}$$

where the last inequality holds from the assumption that $T^2 \leq R/2$. □

Claim 6. *For any $i \in [S]$ and $T^2 \leq R/2$, $\Pr[E_2^i] \leq R^{-10i}$.*

Proof for Claim 6 can be obtained in a similar fashion to that of Claim 5.

Claim 7. *For any $i \in [S]$ and $ST^2 \leq C/2$, $\Pr[E_3^i] \leq R^{-10i}$.*

Proof. Again we will prove the claim via compression. Before we present the encoding algorithm, we note that each pair of queries forming a Type 3 structure share the input message and the output salt. Even though the output salt is unique for every pair, the input message could be same for two pairs. Thus, the decoder may no longer be able to identify the pairs from an unordered set. Thus, our encoder stores the first occurring query from each pair in an unordered set and the later query from each pair in an ordered set (ordered by the corresponding query in the first set).

We give a formal description of the encoding algorithm next.

- Store the first occurring offline query from each of the $10i \log R$ pairs that form the Type 3 structure as an unordered set, say U. This would require $\log \binom{iT}{10i \log R}$ bits.
- Store the later occurring offline query from each of the $10i \log R$ pairs that form the Type 3 structure as an ordered set (ordered by the corresponding query from the pair in U), say V. This would require at most $10i \log R \cdot \log(iT)$ bits.
- Delete the output salt, each at least $(\log C - 1)$ bits long, from each query in the ordered set V from the table of sampled queries on F/F^{-1}.

Then

$$\Pr[E_3^i] \leq \frac{\binom{iT}{10i \log R}}{(C/2)^{10i \log R}} \cdot (iT)^{10i \log R} \leq \left(\frac{2ei^2T^2}{10i \log R \cdot C}\right)^{10i \log R} \leq 2^{-10i \log R}$$

where the last inequality holds from the assumption that $ST^2 \leq C/2$ which implies $iT^2 \leq C/2$ as $i \leq S$. □

Claim 8. *For any $i \in [S]$ and $ST^2 \leq C/2$, $\Pr[E_4^i] \leq R^{-10i}$.*

Proof for Claim 8 can be obtained in a similar fashion to that of Claim 7.

Claims 5-8 are sufficient to show that $\frac{\Pr[E^i]}{\Pr[X_{<i}]} \leq \sum_{j=1}^{4} \frac{\Pr[E_j^i]}{\Pr[X_{<i}]}$ is small enough. Now we need to bound the term $\Pr[X_i | X_{<i} \wedge \overline{E^i}]$. To this end, we will have to analyze all the types of two block collisions that can be found in Sponge hash functions. First, we give some definitions and then identify all the types of 2-block collisions in the next claim.

Claim 9. *To find a 2-block collision on a_i for any $i \in [S]$, the queries in the database should satisfy at least one of the following conditions:*

1. *There exists an offline query that takes $(*, a_i)$ as input or outputs $(*, a_i)$.*
2. *There exists an online query such that it's output is $(*, a_i)$.*
3. *There exist two online queries with corresponding outputs (m', a') and (m'', a'') such that either $m' = m''$ or $a' = a''$.*
4. *There exist an online query, say it's output is denoted by (m, a), and an offline query, say it's input is denoted by (m', a') and output is denoted by (m'', a''), such that $a \in \{a', a''\}$ and $m \in \{m', m''\}$.*
5. *There exist two offline queries and one online query where:*
   - *input and output of the first (occurring) offline query is denoted by (m', a') and (m'', a'') respectively*

- *the second offline query is to F and it's input and output is denoted by (ℓ', b') and (ℓ'', b'') respectively*
- *the output of the online query is denoted by (m, a)*

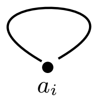
Then either $a = b' = a'$ and $\ell'' = m''$ or $a = b' = a''$ and $\ell'' = m'$. In other words, the offline queries form either Type 1 or Type 2 structure and the output salt of the online is the same as the input salt of the second offline query.

6. *There exist two offline queries and one online query where:*
   - *input and output of the first (occurring) offline query is denoted by (m', a') and (m'', a'') respectively*
   - *the second offline query is to F^{-1} and it's input and output is denoted by (ℓ', b') and (ℓ'', b'') respectively*
   - *the output of the online query is denoted by (m, a)*

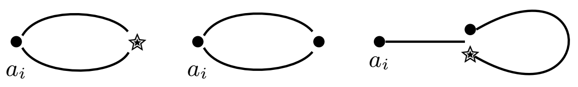
   Then either $a = b'' = a'$ and $\ell' = m''$ or $a = b'' = a''$ and $\ell' = m'$. In other words, the offline queries form either Type 3 or Type 4 structure and the output salt of the online is the same as the output salt of the second offline query.
7. *There exist an offline query and two online queries where:*
   - *input and output of the offline query us denoted by (m', a') and (m'', a'') respectively*
   - *output of the two online queries is denoted by (m, a) and (ℓ, b)*

   Then $a \in \{a', a''\}$ and either $\ell \in \{m', m''\}$ or $b \in \{a', a''\}$.
8. *There exist two offline queries and two online queries where:*
   - *input and output of one offline query is denoted by $(*, a')$ and $(*, a'')$ respectively*
   - *input and output of the other offline query is denoted by $(*, b')$ and $(*, b'')$ respectively*
   - *output of the two online queries is denoted by $(*, a)$ and $(*, b)$*

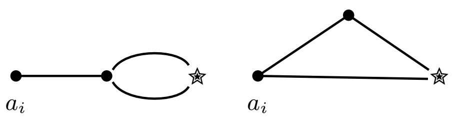
   Then $a \in \{a', a''\}$ and $b \in \{b', b''\}$.



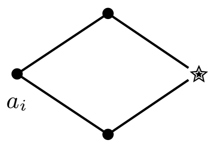
(a) 1 Query collision



(b) 2 Query collisions



(c) 3 Query collisions



(d) 4 Query collision

Fig. 6: Line represents query to F/F^{-1}. Each line could be directed $\rightarrow$ for query to F or $\leftarrow$ for query to F^{-1}. ● represents that (output/input of) the two queries share the salt. ☆ represents that (output/input of) the two queries share the message.

Proof. First, we identify all the types of 2-block collisions for Sponge hash functions in fig. 6. Note that each line shown in the figure could be depicting an offline/online query and a query to F/F^{-1}. We next show that all the possible types of collisions satisfy at least one of the cases in the claim.

First let's consider when none of the colliding queries (i.e., queries involved in collision) is an offline query. Then for the type of collision depicted in fig. 6a will satisfy case 2 in the claim. And all the other types of collisions (depicted in fig. 6b,6c,6d) will satisfy case 3 in the claim when all the colliding queries are online queries.

Next, let's consider when exactly one of the colliding queries is an offline query. For the collision types depicted in fig. 6a and the first two in fig. 6b, this one offline query will necessary have the node a_i at one end (in other words, the offline query will have input/output of the form $(*, a_i)$) and will satisfy case 1 in the claim. Similarly for the remaining collision types, if the offline query is the one with node a_i at one end, the colliding queries will satisfy case 1 in the claim. For the third collision type in fig. 6b, when the offline query is the one depicted by the curved line, it will satisfy case 4 in the claim. All the remaining collision types will satisfy case 7 in the claim when the one offline query is depicted by a line that does not have the node a_i at one of its ends.

Next, we consider when two or more of the colliding queries are offline queries. Again if even one of these offline queries is depicted by a line that has the node a_i at one end, they will satisfy case 1 in the claim. Otherwise, the first collision type depicted in fig. 6c will satisfy one of case 5 or 6 in the claim. The collision type depicted in fig. 6d will satisfy case 8 in the claim. □

Finally, we analyze each case in Claim 9 and show that it's advantage is bounded by $\widetilde{O}((ST^2/C)^2 + ST/C + T^2/C + T^2/R)$ for any $i \in [S]$ when $X_{<i} \wedge \overline{E^i}$.

Case 1: As F/F^{-1} are lazily sampled and there are at most iT offline queries, the probability is bounded by $2iT/C$.

Case 2: For each of the T online queries, the probability it's output is $(*, a)$ is $1/C$. Therefore, the probability is bounded by T/C.

Case 3: By birthday bound, T online queries implies the probability of finding collision is bounded by $T^2/C + T^2/R$.

Case 4: Fix an offline query and the probability that output of at least one of the T online queries can be completely determined by the input and output of the fixed offline query is $4/RC$. Thus, the probability is bounded by $4ST^2/RC$.

Case 5: Conditioned on $\overline{E^i_1} \wedge \overline{E^i_2}$, there exists at most $10i \log R$ pair of offline queries each that form Type 1 and Type 2 structure. Thus, the probability that the output salt of each of the T online queries hitting either Type 1 or Type 2 structure is $20i \log R/C$. Thus, the probability is bounded by $20iT \log R/C$.

Case 6: Conditioned on $\overline{E^i_3} \wedge \overline{E^i_4}$, there exists at most $10i \log R$ pair of offline queries each that form Type 3 and Type 4 structure. Thus, the probability that the output salt of each of the T online queries hitting either Type 3 or Type 4 structure is $20i \log R/C$. Thus, the probability is bounded by $20iT \log R/C$.

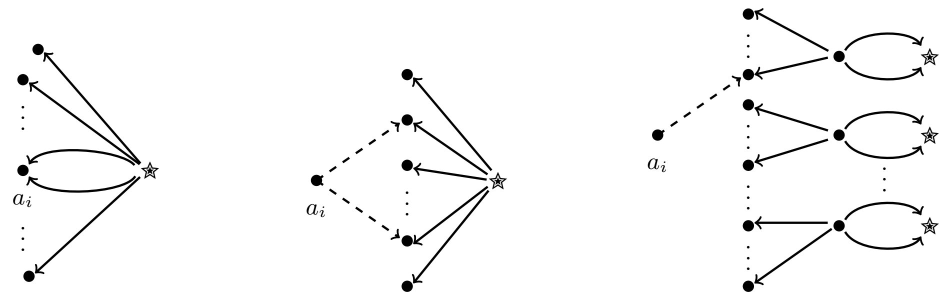
Case 7: Fix an offline query. Then the probability that output salt of one online query is input/output salt of the fixed offline query and output salt or message of one online query is input/output salt or message respectively of the fixed offline query is $2/C \cdot (2/R + 2/C)$. Thus, the probability is bounded by $4ST \cdot T^2/C \cdot (1/R + 1/C)$.

Case 8: Fix two offline queries. Then the probability that output of two online queries is input/output salt of one fixed offline query each is $2/C \cdot 2/C$. Then the probability is bounded by $4ST^2/C \cdot ST^2/C$.

□

5 Limitations for the Multi-Instance model

In this section, we will show that finding collisions is easy in the multi-instance model. For sponge, we will present 3 different attacks, one for each of parameter range $B = 1, B = 2$ and $B \geq 3$. It's worth noticing that the advantage of our attack matches the security bound given by Theorem 8 and [FGK22] in all cases. Therefore, we can not hope to prove better bounds unless we find a better model.



(a) 1-Block collision attack   **(b)** 2-Block collision attack   **(c)** 3-Block collision attack

Fig. 7: Attacks for sponge in multi-instance model. Dashed line represent online query. Solid line represents offline/online query. Line directed $\rightarrow$ represents query to F and line directed $\leftarrow$ represents query to F^{-1}. ● represents that (output/input of) the two queries share the salt. ✯ represents that (output/input of) the two queries share the message.

5.1 Attacks for Sponge in Multi-Instance Model When $B = 1$

Theorem 10. *Suppose $S, T \geq 8$ and $\frac{ST}{C} \leq \frac{1}{2}$. There exists an (S,T)-MI adversary $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{1\text{-}SP}}(\mathcal{A}) = \left(\widetilde{\Omega}\left(\frac{\mathsf{S}^2\mathsf{T}^2}{\mathsf{C}^2}\right)\right)^{\mathsf{S}}.$$

Proof. For the i^{th} stage of an MI game, let the challenge salt be denoted by a_i. In this stage, we will use the following strategy:

- We make T queries $F^{-1}(0,(i-1)\cdot T+j)$ where $j\in[T]$.
- Among all the (online and offline) queries, if there are two different queries $F^{-1}(0,j_1)=(m_1,a_{j_1})$ and $F^{-1}(0,j_2)=(m_2,a_{j_2})$, such that

$$a_{j_1}=a_{j_2}=a_i$$

Then we output the message pair $(m_1),(m_2)$ as the collision.

See figure 7a for reference. The correctness of the attack is immediate.

Next we analyze the lower bound on the success probability of the attack stated above. Define event E_i: There are **exactly** 2 queries (among the iT queries in the first i stages) hitting a_i. If E_i happens, then we will definitely win the i^{th} stage. Consider the event $E_1\wedge E_2\wedge\ldots\wedge E_S$:

$$\begin{aligned}\Pr[E_1\wedge E_2\wedge\ldots\wedge E_S]&=\left(\prod_{i=1}^{S}\binom{iT-2(i-1)}{2}\frac{1}{C^2}\right)\left(1-\frac{S}{C}\right)^{ST-2S}\\&\geq\prod_{i=1}^{S}\left(\frac{iT}{2}\cdot\frac{iT}{4}\cdot\frac{1}{2C^2}\left(1-\frac{S}{C}\right)^{T-2}\right)\\&\geq\prod_{i=1}^{S}\left(\frac{i^2T^2}{16C^2}\left(1-\frac{S(T-2)}{C}\right)\right)\\&\geq\left(\frac{T^2}{16C^2}\cdot\frac{1}{2}\right)^S\prod_{i=1}^{S}i^2\\&\geq\left(\frac{S^2T^2}{16C^2\log^2 S}\right)^S\end{aligned}$$

where the first inequality holds using $i(T-2)\geq\frac{iT}{2}$ (as $T\geq 8$), second to last inequality holds using $\frac{ST}{C}\leq\frac{1}{2}$, and the last inequality holds using the fact $S!\geq\left(\frac{S}{e}\right)^S$ and $e<\log S$.

This event gives a lower bound of success probability of our attack, which is already $\left(\widetilde{\Omega}\left(\frac{S^2T^2}{C^2}\right)\right)^S$. □

5.2 Attacks for Sponge in Multi-Instance Model When $B=2$

Theorem 11. *Suppose $S,T\geq 8$ and $\frac{ST^2}{C}\leq\frac{1}{2}$. There exists an (S,T)-MI adversary $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{2\text{-}SP}}(\mathcal{A})=\left(\widetilde{\Omega}\left(\frac{\mathsf{S}^2\mathsf{T}^4}{\mathsf{C}^2}\right)\right)^{\mathsf{S}}.$$

Proof. Again we denote the challenge salt for the i^{th} stage of the MI game by a_i. We maintain a counter x (initially set to 0) through the whole game. The strategy in the i^{th} stage of the game for any $i\in[S]$ will be as follows:

1. First, we make $\frac{T}{2}$ queries in the following way:
   (a) If the query $F^{-1}(0, x)$ is **fresh**, we make this query. (Recall that a query $F^{-1}(m, a) = (m', a')$ is fresh if neither $F^{-1}(m, a)$ nor $F(m', a')$ is queried before.)
   (b) Set $x \leftarrow x + 1$.
   We will repeat until $\frac{T}{2}$ fresh queries have been made in step (a).
2. Next, we make $\frac{T}{2}$ queries $F(j, a_i)$ where $j \in [\frac{T}{2}]$.
3. If there are at least 2 queries in step 2 that are **not** fresh, they must have the form $F(m_1, a_i) = (0, j_1)$ and $F(m_2, a_i) = (0, j_2)$ (Since all the challenge salts a_i are different, their first occurrence must be in step 1 of some previous stages, which must have form $F^{-1}(0, *) = (*, a_i)$). Then we output the message pair $(m_1), (m_2)$ as the collision.
4. Otherwise, if there exist two different online queries $F(j_1, a_i) = (m_1, a_{i_1}), F(j_2, a_i) = (m_2, a_{i_2})$ from step 2, and two different (online or offline) queries $F^{-1}(0, j_3) = (m_3, a_{i_3}), F^{-1}(0, j_4) = (m_4, a_{i_4})$ from step 1 such that
$$a_{i_1} = a_{i_3}$$
$$a_{i_2} = a_{i_4}$$
   Then we output the message pair $(j_1, m_1 \oplus m_3), (j_2, m_2 \oplus m_4)$ as the collision.

See figure 7b for reference. One may wonder that the adversary may not find enough queries in step 1(a). However, if it happens, since all the C queries $F^{-1}(0, i)$ (where $i \in [0, C)$) are different, it means at least $C - \frac{iT}{2} > 3ST$ queries are not fresh (since $ST < \frac{C}{2}$), which is impossible since there are at most iT different queries till now. Hence, as long as we can find such queries in step 3 or 4, we will succeed in this stage.

Next, we analyze the success probability of this attack. Suppose we are in stage i. Let Q_i be the indicator variable whether the adversary can find such queries in step 3 or 4. Similarly, let $Q_{<i}$ be the indicator variable that the adversary succeeds in finding such queries in each of the stages $1, \ldots, i-1$.

We define another indicator variable E_i for the event: number of new salts a' appeared in the output of queries $F(j', a_i) = (j'', a')$ in step 1 for stage i is no less than $\frac{T}{4}$ (here "new salt" means that this salt has not been the output of any queries in step 1, including the previous stages). Analogous to $Q_{<i}$, we define $E_{<i}$ to be the indicator variable that $E_j = 1$ for all $j < i$. We will show that despite the extra requirement of these events happening, our attack still achieves the desired advantage.

To solve the problem, we analyze the conditional probability $\Pr[Q_i = 1 \wedge E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1]$ for each stage i. To begin with, given that $E_{<i} = 1 \wedge Q_{<i} = 1$, we know that at least $\frac{(i-1)T}{4}$ distinct salts appeared in step 1 from previous stages. Therefore, for $E_i = 0$ to happen $\frac{T}{2}$ queries in step 1 will generate less than $\frac{T}{4}$ new salts (i.e. more than $\frac{T}{4}$ queries generate old salts). However, there are at most $\frac{iT}{2}$ salts visited in step 1 so far. Hence, we

have

$$\Pr[E_i = 0 | Q_{<i} = 1 \wedge E_{<i} = 1] \leq \binom{\frac{T}{2}}{\frac{T}{4}} \left(\frac{iT}{2C}\right)^{\frac{T}{4}} \leq (2e)^{\frac{T}{4}} \left(\frac{iT}{2C}\right)^{\frac{T}{4}} = \left(\frac{eiT}{C}\right)^{\frac{T}{4}}$$

Therefore,

$$\Pr[E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \geq 1 - \left(\frac{eiT}{C}\right)^{\frac{T}{4}} \geq 1 - \left(\frac{eiT^2}{4C}\right) \geq \frac{1}{2}$$

Since $\frac{ST^2}{C} < \frac{1}{2}$.

Next, given $E_i = 1$, we compute a lower bound of the probability for $Q_i = 1$. Since $E_i = 1$ and $E_{<i} = 1$, we know there are at least $\frac{iT}{4}$ salts visited in step 1. Besides, if the adversary fails in step 3, then there are at least $\frac{T}{2} - 1 \geq \frac{T}{4}$ fresh queries in step 2. Now we only consider a special case that the adversary can succeed (which gives a lower bound on probability of event $Q_i = 1$): The adversary fails in step 3, and among the **first** $\frac{T}{4}$ fresh online queries in step 2, **exactly** 2 of them hits two of the **first** $\frac{iT}{4}$ new salts in step 1:

$$\begin{aligned}
\Pr[Q_i = 1 | E_i = 1 \wedge Q_{<i} = 1 \wedge E_{<i} = 1] &\geq \binom{\frac{T}{4}}{2} \left(\frac{iT}{4C}\right)^2 \left(1 - \frac{iT}{4C}\right)^{\frac{T}{4}-2} \\
&\geq \frac{T^2}{64} \left(\frac{iT}{4C}\right)^2 \left(1 - \frac{iT^2}{16C}\right) \\
&\geq \frac{i^2T^4}{2^{11}C}
\end{aligned}$$

Since $\frac{ST^2}{C} \leq \frac{1}{2}$.

Hence,

$$\begin{aligned}
&\Pr[Q_i = 1 \wedge E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \\
&= \Pr[E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \cdot \Pr[Q_i = 1 | E_i = 1 \wedge Q_{<i} = 1 \wedge E_{<i} = 1] \\
&\geq \frac{1}{2} \cdot \frac{i^2T^4}{2^{11}C} = \frac{i^2T^4}{2^{12}C^2}
\end{aligned}$$

Finally, the advantage of our attack is at least:

$$\begin{aligned}
&\Pr[Q_1 = 1 \wedge Q_2 = 1 \ldots \wedge Q_S = 1] \\
&\geq \Pr[Q_1 = 1 \wedge Q_2 = 1 \ldots \wedge Q_S = 1 \wedge E_1 = 1 \wedge E_2 = 1 \ldots \wedge E_S = 1] \\
&= \prod_{i=1}^{S} \Pr[Q_i = 1 \wedge E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \\
&\geq \prod_{i=1}^{S} \left(\frac{i^2T^4}{2^{12}C^2}\right) \\
&\geq \left(\frac{S^2T^4}{2^{12}C^2 \log^2 S}\right)^S
\end{aligned}$$

which proves the theorem. □

5.3 Attacks for Sponge in Multi-Instance Model When $B \geq 3$

Theorem 12. *Suppose $R, S, T \geq 16$, $\frac{ST^2}{C} \leq \frac{1}{2}$ and $\frac{T^2}{R} \leq 1$. For any $B \geq 3$, there exists an (S,T)-MI adversary $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{MICR}}_{\mathsf{B\text{-}SP}}(\mathcal{A}) = \left(\widetilde{\Omega}\left(\frac{\mathsf{ST}^2}{\mathsf{C}}\right)\right)^{\mathsf{S}}.$$

The assumption $\frac{T^2}{R} \leq 1$ is required, otherwise the trivial birthday attack will already have $\Omega(1)$ advantage even in the Auxiliary-Input setting.

Proof. We will propose an attack that only uses 3-block messages, which is valid for any $B \geq 3$. Our attack is as follows:

```
– Initially set $x \leftarrow 0, y \leftarrow 0$.
– For stage $i$ of the MI game (let the challenge salt in this stage be $a_i$):
    1. First, we make $\frac{T}{2}$ queries in the following fashion:
        (a) If $F^{-1}(x,y)$ is **fresh**, we make this query.
        (b) Set $x \leftarrow x+1$.
        (c) If $x \geq R$, we set $y \leftarrow y+1$ and $x \leftarrow 0$.
        We repeat until $T/2$ new fresh queries are made in (a).
    2. Next, we make $T/2-2$ queries of the form $F(j,a_i)$ where $j \in [\frac{T}{2}]$.
    3. If any query in step 2 is **not** fresh, it must have form $F(m_1,a_i) = (j_1,y')$
       where $0 \leq y' \leq y$ (Similarly, this is since that the first occurrence of this
       query must be in step 1(a), which has form $F^{-1}(*,y') = (*,a_i)$). Then we
       make two extra queries $F(0,y')$ and $F(1,y')$. If $F(0,y')[1] = F(1,y')[1]$, we
       output the message pair $(m_1,j_1),(m_1,j_1 \oplus 1)$ as the collision.
    4. Otherwise, if there exists one online query $F(j_1,a_i) = (m_1,a_{i_1})$ and one
       (online or offline) query $F^{-1}(j_2,y') = (m_2,a_{i_2})$ such that $a_{i_1} = a_{i_2}$, then
       we make two extra queries $F(0,y')$ and $F(1,y')$. If $F(0,y')[1] = F(1,y')[1]$,
       we output the message pair $(j_1,m_1 \oplus m_2,j_2),(j_1,m_1 \oplus m_2,j_2 \oplus 1)$ as the
       collision.
```

See figure 7c for reference. Recall that $F(0,y')[1]$ means the first part of output (i.e. message) of query $F(0,y')$. Our attack will need to form $y+1$ of the structures shown in the figure. Notice that since we introduced y, the adversary may eventually check all possible $F^{-1}(x,y)$, and thus will always find enough queries in step 1(a) (since $\frac{ST}{RC} < \frac{1}{2}$).

First, we claim that $y \leq \frac{ST}{R}$ after all S stages of execution. This is because if any query in step 1(a) is not fresh, then it has occurred before in step 2-4 (of a previous stage). However, there are at most $\frac{ST}{2}$ such queries in total. Therefore, the number of queries that are fresh in step 1(a) is at least $yR - \frac{ST}{2}$ (There can be more fresh queries if any of $F^{-1}(i,y)$ is fresh for some $0 \leq i < x$). Further, we know that there are exactly $\frac{ST}{2}$ fresh queries in step 1(a) after S stages, so we have $yR - \frac{ST}{2} \leq \frac{ST}{2}$, which means $y \leq \frac{ST}{R}$.

Let us define an indicator variable W_i as whether $F(0,i)[1] = F(1,i)[1]$, and variable $W = W_0 \wedge W_1 \wedge \ldots \wedge W_y$. If $W = 1$ and that we find such queries in step 3 or 4, it's not hard to check that our output forms a valid collision and we win this stage.

First, we claim that the event $W = 1$ will happen with large enough probability via lazy-sampling. For $i \in [y]$, $F(0,i)$ and $F(1,i)$ are sampled in order without replacement. For any $i \in [y]$ and for some m, the probability $F(1,i)[1] = m$ depends on the number of previous samples that had output $(m,*)$. Note that there can be at most $2i+1$ such samples. Therefore,

$$\begin{aligned}\Pr[W=1] &= \prod_{i=0}^{y} \Pr[W_i = 1 | W_{<i} = 1] = \prod_{i=0}^{y} \Pr[F(1,i)[1] = F(0,i)[1] | W_{<i} = 1] \\ &\geq \prod_{i=0}^{y} \frac{C-(2i+1)}{RC-(2i+1)} \geq \left(\frac{C-(2y+1)}{RC}\right)^{y+1} \geq \left(\frac{C/2}{RC}\right)^{y+1} = \left(\frac{1}{2R}\right)^{y+1}\end{aligned}$$

where the last inequality uses that $2y+1 \leq 4ST/R \leq C/2$.

Next, we focus on the probability of finding such queries. We define $Q_i, Q_{<i}, E_i, E_{<i}$ the same way as in section 5.2. With the same analysis, we have

$$\Pr[E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \geq \frac{1}{2}$$

Now, given $E_i = 1$, we need to lower bound the probability for $Q_i = 1$ again. Here, we consider the case that the adversary fails in step 3, and that there is **exactly** 1 online query (Since the adversary fails in step 3, this query must be fresh) that hits the **first** $\frac{iT}{4}$ new salts (from step 1):

$$\begin{aligned}\Pr[Q_i = 1 | E_i = 1 \wedge Q_{<i} = 1 \wedge E_{<i} = 1] &\geq \left(\frac{T}{2} - 2\right) \cdot \frac{iT}{4C}\left(1 - \frac{iT}{4C}\right)^{\frac{T}{2}-3} \\ &\geq \frac{iT^2}{16C}\left(1 - \frac{iT^2}{8C}\right) \geq \frac{iT^2}{32C}\end{aligned}$$

Hence

$$\begin{aligned}&\Pr[Q_i = 1 \wedge E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \\ &= \Pr[E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \cdot \Pr[Q_i = 1 | E_i = 1 \wedge Q_{<i} = 1 \wedge E_{<i} = 1] \\ &\geq \frac{iT^2}{64C}\end{aligned}$$

Finally, the success probability of our attack will be at least

$$
\begin{aligned}
&\Pr[W = 1 \wedge Q_1 = 1 \wedge Q_2 = 1 \ldots \wedge Q_S = 1] \\
&\geq \Pr[W \wedge Q_1 = 1 \wedge Q_2 = 1 \ldots \wedge Q_S = 1 \wedge E_1 = 1 \wedge E_2 = 1 \ldots \wedge E_S = 1] \\
&= \Pr[W = 1] \cdot \prod_{i=1}^{S} \Pr[Q_i = 1 \wedge E_i = 1 | Q_{<i} = 1 \wedge E_{<i} = 1] \\
&\geq \left(\frac{1}{R}\right)^{y+1} \prod_{i=1}^{S} \left(\frac{iT^2}{64C}\right) \\
&\geq \left(\frac{T^2}{64C}\left(\frac{1}{R}\right)^{\frac{2T}{R}}\right)^{S} \prod_{i=1}^{S} i \\
&\geq \left(\frac{T^2}{64C}\left(\frac{1}{2}\right)^{\frac{2T\log R}{R}}\right)^{S} \left(\frac{S}{\log S}\right)^{S} \\
&\geq \left(\frac{ST^2}{256C\log S}\right)^{S}
\end{aligned}
$$

where third to last inequality holds as $y + 1 \leq \frac{2ST}{R}$ (as explained above), while the last inequality holds since $\frac{T}{\sqrt{R}} \leq 1$ and $\frac{\log R}{\sqrt{R}} \leq 1$. □

References

ACDW20. Akshima, David Cash, Andrew Drucker, and Hoeteck Wee. Time-space tradeoffs and short collisions in merkle-damgård hash functions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 157–186. Springer, 2020.

AGL22. Akshima, Siyao Guo, and Qipeng Liu. Time-space lower bounds for finding collisions in merkle-damgård hash functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 192–221. Springer, 2022.

BDPA07. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.

BDPA08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

CDG18. Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra

Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, 2018.

CDGS18. Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 227–258. Springer, 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. Springer, 2018.

CGK19. Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In *Theory of Cryptography Conference*, pages 393–421. Springer, 2019.

CGLQ20. Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 673–684. IEEE, 2020.

CHM20. Dror Chawin, Iftach Haitner, and Noam Mazor. Lower bounds on the time/memory tradeoff of function inversion. In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, pages 305–334, 2020.

CK19. Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 393–421. Springer, 2019.

Dam89. Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.

DGK17. Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495, 2017.

DTT10. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *Annual Cryptology Conference*, pages 649–665. Springer, 2010.

FGK22. Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Time-space tradeoffs for sponge hashing: Attacks and limitations for short collisions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, 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 Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 440–469. Springer, 2023.

GGKL21. Nick Gravin, Siyao Guo, Tsz Chiu Kwok, and Pinyan Lu. Concentration bounds for almost k-wise independence with applications to non-uniform security. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2404–2423, 2021.

GGPS23. Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. Revisiting time-space tradeoffs for function inversion. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 453–481. Springer, 2023.

GK22. Ashrujit Ghoshal and Ilan Komargodski. On time-space tradeoffs for bounded-length collisions in merkle-damgård hashing. In *Annual International Cryptology Conference*. Springer, 2022.

GT23. Ashrujit Ghoshal and Stefano Tessaro. The query-complexity of preprocessing attacks. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 482–513. Springer, 2023.

Hel80. M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theor.*, 26(4):401–406, July 1980.

IK10. Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 617–631, 2010.

Mer89. Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 218–238, 1989.

A Proof Details of Lemma 1

A.1 Proof of Claim 2

In this case, as $ST < C$, we must have $R < C$. Now, according to our analysis in section 3.1.3, we have

$$\begin{aligned} L_1 + L_2 + L_3 \le 4\log S &+ S_1(2\log eST - 2\log S_1 - \log R + 1) \\ &+ S_2(2\log eST - 2\log S_2 + 2\log S + 2 - 2\log C) \\ &+ \sum_{\substack{i\in[R]:\\ Q_i > \log STR}} (\log S_3 + Q_i \log eS_3 - (Q_i - 1)(\log R - 1)) \\ &+ \sum_{\substack{i\in[R]:\\ Q_i \le \log STR}} (Q_i \log\log STR) \\ &+ S_3(2\log eST - 2\log S_3 - \log C + 1) \end{aligned} \tag{1}$$

Therefore, we have

$$2^{L_1+L_2+L_3}$$

$$
\begin{aligned}
&\leq S^4 \left(\frac{2e^2S^2T^2}{S_1^2R}\right)^{S_1} \left(\frac{4e^2S^4T^2}{S_2^2C^2}\right)^{S_2} \left(\frac{2e^2S^2T^2}{S_3^2C}\right)^{S_3} \prod_{\substack{i\in[R]:\\ Q_i>\log STR}} \left(\frac{S_3(2eS_3)^{Q_i}}{R^{Q_i-1}}\right) \prod_{\substack{i\in[R]:\\ Q_i\leq\log STR}} (\log STR)^{Q_i} \\
&\leq \left(\frac{S}{S_1}\right)^{2S_1} \left(\frac{2e^2T^2}{R}\right)^{S_1} \left(\frac{S}{S_2}\right)^{2S_2} \left(\frac{4e^2S^2T^2}{C^2}\right)^{S_2} \left(\frac{S}{S_3}\right)^{2S_3} \left(\frac{2e^2T^2}{C}\right)^{S_3} \\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR}} \left\{\left(\frac{4eS_3}{R}\right)^{Q_i} \cdot (\log STR)^{S_3-Q_{tot}}\right\} \qquad (2) \\
&\leq e^{6S} \left(\frac{2e^2T^2}{R}\right)^{S_1} \left(\frac{4e^2S^2T^2}{C^2}\right)^{S_2} \left(\frac{2e^2T^2\log STR}{C}\right)^{S_3-Q_{tot}} \left(\frac{2e^2T^2}{C}\right)^{Q_{tot}} \\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR}} \left(\frac{4eS_3}{R}\right)^{Q_i} \qquad (3) \\
&\leq \left(\frac{2e^8T^2}{R}\right)^{S_1} \left(\frac{4e^8S^2T^2}{C^2}\right)^{S_2} \left(\frac{2e^8T^2\log STR}{C}\right)^{S_3-Q_{tot}} \left(\frac{2e^8T^2}{C}\right)^{Q_{tot}} \\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR}} \left(\frac{4eS_3}{R}\right)^{Q_i} \\
&\leq \left(2e^8\left(\frac{T^2}{R}+\frac{2S^2T^2}{C^2}\right)\right)^{S_1+S_2} \left(\frac{2e^8T^2\log STR}{C}\right)^{S_3-Q_{tot}} \left(\frac{8e^9S_3T^2}{RC}\right)^{Q_{tot}} \\
&\leq \left(8e^9\left(\frac{T^2}{R}+\frac{S^2T^2}{C^2}+\frac{T^2\log STR}{C}+\frac{S_3}{C}\cdot\frac{T^2}{R}\right)\right)^{S_1+S_2+S_3} \\
&= \tilde{O}\left(\frac{T^2}{R}+\left(\frac{ST}{C}\right)^2+\frac{T^2}{C}\right)^S \qquad (4) \\
&= \tilde{O}\left(\frac{T^2}{R}+\left(\frac{ST}{C}\right)^2\right)^S \qquad (5)
\end{aligned}
$$

where (1) holds using $\log\binom{n}{m} \leq m\log en - m\log m$, (2) holds as $2^S > S^4$ (since $S > 32$), $STR \leq 2^{Q_i}$ and $S_1+S_2+S_3 = S$, (3) holds as $\left(\frac{S}{x}\right)^x \leq e^S$, (4) holds since $\frac{S_3}{C} \leq \frac{ST}{C} \leq \frac{1}{8e^9}$, and (5) is using $R < C$.

A.2 Proof of Claim 3

The analysis is very similar, except for type 3. For completeness, we still write down the whole calculation:

$$
\begin{aligned}
L_1 + L_2 + L_3 \leq\ & 4\log S + S_1(2\log eST - 2\log S_1 - \log R + 1) \\
& + S_2(2\log eST - 2\log S_2 + 2\log S + 2 - 2\log C)
\end{aligned}
$$

$$+ \sum_{\substack{i\in[R]:\\ Q_i>\log STR\\ K_i\geq 1}} (\log ST + Q_i \log eST - (Q_i-1)(\log R - 1) - (Q_i - K_i)\log Q_i)$$
$$+ \sum_{\substack{i\in[R]:\\ Q_i\leq\log STR\\ K_i\geq 1}} (K_i \log\log C) + S_3(\log eST - \log S_3 - \log C + 1) \tag{6}$$

Therefore, we have

$$\begin{aligned}
&2^{L_1+L_2+L_3}\\
&\leq S^4 \left(\frac{2e^2S^2T^2}{S_1^2R}\right)^{S_1} \left(\frac{4e^2S^4T^2}{S_2^2C^2}\right)^{S_2} \left(\frac{2eST}{S_3C}\right)^{S_3}\\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR\\ K_i\geq 1}} \left(\frac{ST\cdot(2eST)^{Q_i}}{R^{Q_i-1}Q_i^{Q_i-K_i}}\right) \cdot \prod_{\substack{i\in[R]:\\ Q_i\leq\log STR\\ K_i\geq 1}} (\log STR)^{K_i}\\
&\leq \left(\frac{S}{S_1}\right)^{2S_1} \left(\frac{2e^2T^2}{R}\right)^{S_1} \left(\frac{S}{S_2}\right)^{2S_2} \left(\frac{4e^2S^2T^2}{C^2}\right)^{S_2} \left(\frac{S}{S_3}\right)^{S_3} \left(\frac{2eT}{C}\right)^{S_3}\\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR\\ K_i\geq 1}} \left(\left(\frac{4eST}{R}\right)^{Q_i} \cdot \frac{1}{Q_i^{Q_i-K_i}}\right) \cdot (\log STR)^{S_3-K_{tot}} && (7)\\
&\leq e^{5S} \left(\frac{2e^2T^2}{R}\right)^{S_1} \left(\frac{4e^2S^2T^2}{C^2}\right)^{S_2} \left(\frac{2eT\log STR}{C}\right)^{S_3-K_{tot}} \left(\frac{2eT}{C}\right)^{K_{tot}}\\
&\qquad \cdot \prod_{\substack{i\in[R]:\\ Q_i>\log STR\\ K_i\geq 1}} \left(\left(\frac{4eST}{R\log STR}\right)^{Q_i-K_i} \cdot \left(\frac{4eST}{R}\right)^{K_i}\right) && (8)\\
&\leq \left(\frac{2e^7T^2}{R} + \frac{4e^7S^2T^2}{C^2}\right)^{S_1+S_2} \left(\frac{2e^6T\log STR}{C}\right)^{S_3-K_{tot}} \left(\frac{8e^7ST^2}{RC}\right)^{K_{tot}} \left(\frac{4eST}{R\log STR}\right)^{Q_{tot}-K_{tot}} && (9)\\
&\leq \left(8e^7\left(\frac{T^2}{R} + \frac{S^2T^2}{C^2} + \frac{T\log STR}{C}\right)\right)^{S_1+S_2+S_3} && (10)\\
&= \tilde{O}\left(\frac{T^2}{R} + \left(\frac{ST}{C}\right)^2 + \frac{T}{C}\right)^S
\end{aligned}$$

where (6) holds because $\log\binom{n}{m} \leq m\log en - m\log m$, (7) holds as $2^S > S^4$ (since $S > 32$), $STR \leq 2^{Q_i}$ and $S_1+S_2+S_3 = S$, (8) holds using $\left(\frac{S}{x}\right)^x \leq e^S$ and $Q_i > \log STR$, (9) holds because $\sum_i K_i = S_3$ and $K_i \leq Q_i$, and (10) holds since $ST \leq R, \log S > 4e, \frac{T}{C} < \frac{1}{8e^7}$ and $\frac{S}{C} < \frac{1}{8e^7}$. □