Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions

Cody Freitag[1], Ashrujit Ghoshal[2], and Ilan Komargodski[3]

[1] Cornell Tech
cfreitag@cs.cornell.edu
[2] Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, Washington, USA
ashrujit@cs.washington.edu
[3] School of Computer Science and Engineering, Hebrew University of Jerusalem
and NTT Research 91904 Jerusalem, Israel
ilank@cs.huji.ac.il

Abstract. Sponge hashing is a novel alternative to the popular Merkle-Damgård hashing design. The sponge construction has become increasingly popular in various applications, perhaps most notably, it underlies the SHA-3 hashing standard. Sponge hashing is parametrized by two numbers, r and c (bitrate and capacity, respectively), and by a fixed-size permutation on $r + c$ bits. In this work, we study the collision resistance of sponge hashing instantiated with a random permutation by adversaries with arbitrary S-bit auxiliary advice input about the random permutation that make T online queries. Recent work by Coretti et al. (CRYPTO '18) showed that such adversaries can find collisions (with respect to a random c-bit initialization vector) with advantage $\Theta(ST^2/2^c + T^2/2^r)$.

Although the above attack formally breaks collision resistance in some range of parameters, its practical relevance is limited since the resulting collision is very long (on the order of T blocks). Focusing on the task of finding *short* collisions, we study the complexity of finding a B-block collision for a given parameter $B \geq 1$. We give several new attacks and limitations. Most notably, we give a new attack that results in a single-block collision and has advantage

$$\Omega\left(\left(\frac{S^2T}{2^{2c}}\right)^{2/3} + \frac{T^2}{2^r}\right).$$

In certain range of parameters (e.g., $ST^2 > 2^c$), our attack outperforms the previously-known best attack. To the best of our knowledge, this is the first natural application for which sponge hashing is *provably less secure* than the corresponding instance of Merkle-Damgård hashing. Our attack relies on a novel connection between single-block collision finding in sponge hashing and the well-studied function inversion problem. We also give a general attack that works for any $B \geq 2$ and has advantage $\Omega(STB/2^c + T^2/2^{\min\{r,c\}})$, adapting an idea of Akshima et al. (CRYPTO '20).

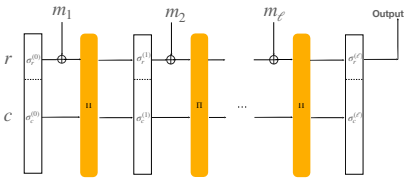
We complement the above attacks with bounds on the best possible attacks. Specifically, we prove that there is a qualitative jump in the advantage of best possible attacks for finding unbounded-length collisions and those for finding very short collisions. Most notably, we prove (via a highly non-trivial compression argument) that the above attack is optimal for $B = 2$ in some range of parameters.

1 Introduction

Due to a series of successful attacks on widely used hash functions such as MD5, SHA-0, and SHA-1, in 2006 the National Institute of Standards and Technology (NIST) organized a competition to create a new hash standard. At that time, the existing hash functions were all based on the well-known Merkle-Damgård hash function construction [24,25,26,14]. The goal of the competition was to find an alternative, dissimilar cryptographic hashing design. It took almost a decade until the winner, a family of cryptographic functions called Keccak, become a hashing standard called SHA-3. The Keccak family is based on the *sponge construction* [8,7] which was a novel alternative to the popular Merkle-Damgård design. By now, the sponge paradigm is used for building collision resistant hash functions, message authentication codes (MACs), pseudorandom functions (PRFs) [9], key derivation functions [19], and more.

A sponge function $\mathsf{Sp}\colon \{0,1\}^* \to \{0,1\}^r$ is defined via three parameters: (1) two natural numbers r (for bitrate) and c (for capacity) so that $n = c + r$, (2) an initial state $\sigma^{(0)} = (\sigma_r^{(0)}, \sigma_c^{(0)}) \in \{0,1\}^r \times \{0,1\}^c$, and (3) a function $\Pi : \{0,1\}^n \to \{0,1\}^n$ which is usually thought of as a (public) pseudorandom permutation. The hashing operation (a.k.a. absorbing) is defined by iterating the state by computing a *round function*. Specifically, given a sequence of r-bit blocks $(m_1, m_2, \ldots, m_\ell)$, $\mathsf{Sp}(m_1, m_2, \ldots, m_\ell)$ is defined as:[4]

1. For $i = 1, \ldots, \ell$, do:
   (a) Compute the round function $\Pi((\sigma_r^{(i-1)} \oplus m_i) \parallel \sigma_c^{(i-1)})$ and let $\sigma^{(i)}$ denote the output.
   (b) Parse $\sigma^{(i)}$ as $(\sigma_r^{(i)}, \sigma_c^{(i)}) \in \{0,1\}^r \times \{0,1\}^c$.
2. Output the first r bits of $\sigma^{(\ell)}$, namely, $\sigma_r^{(\ell)}$.



Typically, $\sigma_r^{(0)}$ is initialized to 0 and $\sigma_c^{(0)}$ is a random initialization vector (IV). If one wants to be explicit, we write $\mathsf{Sp}_{r,c,\Pi,\mathsf{IV}}$ for the sponge function. There are several common instances of r and c used in practice, for example in SHA-3-256 $c = 512$ and $r = 1088$, and in SHA-3-512 $c = 1024$ and $r = 576$. These instance are particularly useful since they were designed to be used as drop-in replacements for the corresponding SHA-2 instances, and as such they were intended to have identical (or better) security properties.

[4] For simplicity, we do not consider padding of the input.

Sponge in the random permutation model. The concrete permutations Π that are used in real-life do not have solid theoretical foundations from the perspective of provable security. Therefore, when coming to analyze the security of the sponge construction, we model the permutation Π as a completely random one. That is, the permutation is randomly chosen, and all parties are given (black-box) access to it and its inverse.[5] This is called the *random permutation model* (RPM). Such bounds are used as an approximation to the best possible security level that can be achieved by the corresponding construction in the real-life implementation. This heuristic has been extensively and successfully used in the past several decades, with exceptions (i.e., examples where the real-life implementation and the ideal world construction are separated) being somewhat contrived and artificial. For "natural" applications it is widely believed that the concrete security proven in the RPM is the right bound even in the real-world, assuming the "best possible" instantiation for the idealized permutation is chosen.

As mentioned, the sponge construction was introduced by Bertoni et al. [8] and its security was analyzed in a follow-up work [7] assuming that the underlying hash function is an invertible random permutation. The latter work showed a strong property called *indifferentiability* from a random oracle, which directly implies many other properties such as collision resistance, pseudorandomness, and more.

For instance, the following is known about Sponge's collision resistance (which is perhaps the most widely used property). For fixed c, r, the collision resistance game is defined as follows: a challenger sends a uniformly random IV to the adversary. The adversary "wins" if it is able to come up with distinct $m, m' \in \{0,1\}^*$ for which $\mathsf{Sp}_{r,c,\Pi,\mathsf{IV}}(m) = \mathsf{Sp}_{r,c,\Pi,\mathsf{IV}}(m')$. There is a well-known attack due to the original works of Bertoni et al. [8,7]: the adversary is given an IV and it merely queries the permutation oracle on inputs of the form $(m\|\text{IV})$, where the m's are chosen uniformly at random. If a collision was observed (i.e., the adversary finds distinct m_1, m_2 such that the first r bits of $\Pi(m_1\|\text{IV})$, $\Pi(m_2\|\text{IV})$ are the same), then the adversary wins. By the well-known birthday bound, the success probability of this event is $\Omega(T^2/2^r)$. Alternatively, if two messages m_1, m_2 such that the query returned a state with the same last c bits (i.e., $\Pi(m_1\|\text{IV}) = a_1\|b$ and $\Pi(m_2\|\text{IV}) = a_2\|b$, then $m_1 \,\|\, a_1$ and $m_2 \,\|\, a_2$ form a collision. The success probability of this event is $\Omega(T^2/2^c)$. Overall, the attacker wins with probability $\Omega(T^2/2^{\min\{c,r\}})$. This is known to be the best possible attack due to the indifferentiability result of [7].

Non-uniformity / preprocessing attacks. The above discussion assumes that the adversary is uniform in the sense that it starts off with no knowledge about Π, as if it did not exist before it was invoked. However, this does not capture real-life attack scenarios where an attacker can invest a significant amount

[5] In typical permutation designs, including the permutations underlying the Keccak family, if you have the entire state, you can apply the inverse permutation to go backward to the previous state. This is why we also give free access to the inverse of the permutation as part of the model.

of preprocessing on the public permutation Π to speed up the actual attack whenever the IV is chosen. This is why most works (at least in theoretical cryptography) model attackers as non-uniform machines, where the attacker could obtain arbitrary but bounded-length advice, before attacking the system. The advice generation phase is called *the offline phase* and the "attack" given the advice and the challenge is called *the online phase*. The output size of the offline phase (i.e., the size of the advice) is denoted S and the number of queries allowed in the online phase is denoted T; computation is free of charge in both phases. This model, being an extension of the RPM where the online adversary may know a bounded-length hint about the permutation, is called the auxiliary-input RPM, or AI-RPM in short. This model was first explicitly put forward by Coretti, Dodis, and Guo [10], naturally extending the influential auxiliary-input random oracle model (AI-ROM) from the seminal work of Unruh [31] (which in turn is an explicit version of the model studied by Hellman [23], Yao [33], and Fiat-Naor [17]). Bounds on the power of "auxiliary-input" adversaries are also referred to as "time-space" trade-offs.

Although the sponge paradigm is becoming widespread, very little is known about its formal security guarantees against such attackers that may have a short preprocessed hint about the permutation computed in an offline phase. In fact, there is an attack that utilizes this extra power to achieve advantage $\Omega(ST^2/2^c+T^2/2^r)$ (notice the extra multiplicative S term).[6] The attack is based on a combination of a birthday-style attack, as above, together with a variant of an attack by Hellman [23] which is nowadays referred to as rainbow tables (due to Oechslin [29]). While this attack uses known techniques, we were not able to find an explicit description of it in the literature and so for completeness, we give the attack and its analysis in Section 4.1.[7] [8] Only very recently, in the beautiful work of Coretti et al. [10] (henceforth CDG) it was shown that this attack is optimal; that is, no S-space T-query attackers can find a collision with probability better than $\Omega(ST^2/2^c + T^2/2^r)$.

It turns out that the above attack results in a very long collision. Specifically, for parameters S and T as above, the above attack results in a collision of length $\approx T$. While this formally breaks collision resistance, it is hard to imagine a natural application where such a collision would be helpful in an attack. Say we have a system that uses a sponge-based hash with an output of size 256 bits. Running the above attack with $S = T = 2^{60}$ would result in a collision of several petabytes long, which is likely to be practically useless for any natural attack scenario. Therefore, we ask whether there exist attacks that find shorter collisions and what is their success probability. Specifically, we introduce an

[6] Throughout the introduction, for easy of notation, we supress poly-logarithmic (i.e., $\mathsf{poly}(c, r)$ terms inside the big "O/Ω" notation. The formal theorems state the precise bounds.

[7] More precisely, we give a generalization of this attack which finds collisions of length $B \geq 2$, and this particular attack follows by setting $B = T$.

[8] A related bound is stated in CDG [10, Table 1] but after communication with an author, they confirmed that the attack was never worked out.

additional parameter B (for blocks) and require an attacker, on a random IV, to come up with two $\leq B$-block messages that collide. The main question studied in this work is:

What is the complexity of a preprocessing attacker in finding a B block collision in a Sponge hash function, assuming the underlying permutation is modeled as random?

1.1 Detour: The Case of Merkle-Damgård

Except being a fundamental problem with theoretical and practical importance, another motivation to study the above question comes from the recent work of Akshima et al. [3] (henceforth ACDW), who studied a similar question in the context of Merkle-Damgård hashing (henceforth MD). Recall that sponge hashing was designed to be used as a drop-in replacement for Merkle-Damgård-based hash functions, and as such, it is essential to compare their security guarantees in this natural model that allows attackers to perform preprocessing.

Recall that a Merkle-Damgård hash is defined relative to a *compression* function $h\colon [N] \times [M] \to [N]$. Hashing is performed by breaking the input message into blocks from $[M]$, and processing them one at a time with the compression function, each time combining a block of the input with the output of the previous round, where the 0th round value is the IV.[9] To obtain provable-security guarantees, the analysis models the underlying compression function h as a completely random one. Preprocessing attackers are captured by considering the AI-ROM [23,33,17,31,10,11] which models attackers as two-stage algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. The first algorithm $\mathcal{A}_1$ is unbounded except that it generates an S-bit "advice". The second algorithm $\mathcal{A}_2$ gets the advice and makes T queries to the oracle.

Coretti et al. [11] fully characterize the collision resistance of salted-MD hashing: there exists an attack with advantage $\Omega(ST^2/N+T^2/N)$ (loosely based on the idea of rainbow tables [23,29]), and this is the best possible attack, as shown using the "bit-fixing" technique [31]. As in the case of sponge hashing, this attack results in a very long collision, on the order of T blocks. Motivated by this observation, ACDW [3] ask whether it is strictly harder to find shorter collisions. They have two main results. The first is an extension of the above simple attack to result in B-block collisions with advantage $\Omega(STB/N + T^2/N)$. The second result is an upper bound on the advantage for $B = 2$ of $O(ST/N + T^2/N)$, showing that the above attack is tight. For $B = 1$, the problem is equivalent to finding collisions in a compressing random function, and the advantage is precisely $\Theta(S/N + T^2/N)$ [16].

ACDW [3] could not prove or disprove that their $\Omega(STB/N+T^2/N)$ attack is optimal for any other value of B (except $B = 2$ and $B \in \Omega(T)$). They conjectured that it is optimal and formulated it as *the STB conjecture*. In very recent works, Akshima, Guo, and Liu [4] and Ghoshal and Komargodski [21] proved

[9] All of the results directly extend to the padded version, but we ignore it for simplicity.

new bounds for this problem (almost resolving the conjecture). Can we prove similar bounds for sponge hashing? Should we believe an analogous conjecture?

1.2 Our Results

We initiate the study of time-space tradeoffs for *bounded length collisions in sponge hashing*. First, the known best attack that gives a single-block collision has advantage

$$\Omega\left(\frac{S}{2^c}+\frac{T^2}{2^r}\right). \tag{1}$$

In this attack, the preprocessing is used to "remember" a collision for S different IVs. If the challenge IV is in the set of remembered IVs, then the attack succeeds (this happens with probability $S/2^c$); otherwise, we run a birthday-style attack which succeeds with probability $\Omega(T^2/2^r)$. For MD hashing, the analogous bound for $B = 1$ is known to be tight. Second, there is an attack (loosely based on rainbow tables) that has advantage $\Omega(ST^2/2^c + T^2/2^r)$ and results with a $\Omega(T)$-blocks collision [10].

At this point, if one were to speculate that sponge's security guarantees are at least as good as MD's, one would guess that the above attacks should be tight, at least for $B \in \{1, 2\}$. With some luck and labor, we may even be able to prove it. This is where the situation gets interesting. We show that the above speculation is **false** for $B = 1$ and in some natural settings of parameters, **sponge is strictly less secure than MD** for this task. On the other hand, for $B = 2$ we can only prove tightness for a certain range of parameters.

In what follows, we elaborate on our results. We design two new attacks, one designed for any $B \geq 2$ and the other specifically for $B = 1$. We complement our attacks with "lower bounds", which are actually upper bounds on the best possible advantage. Specifically, we prove that there is a qualitative jump in the advantage of best possible attacks for finding unbounded-length collisions and those for finding very short collisions (i.e., $B \leq 2$).

Attacks

We give two new attacks, one for any $B \geq 2$ and the other is specialized for $B = 1$. The generic attack is the first to result with an arbitrary block length collision while the one specialized to $B = 1$ beats the previously known best attack, at least in some range of parameters. By the latter, to the best of our knowledge, we show the first natural application for which sponge hashing is less secure than MD.

A new attack for $B \geq 2$. The above-mentioned attack on sponge hashing that has advantage $\Omega(ST^2/2^c + T^2/2^r)$ can be modified to result with a B-block collision for $B \geq 2$ and with advantage

$$\Omega\left(\frac{STB}{2^c}+\frac{T^2}{2^{\min\{c,r\}}}\right). \tag{2}$$

The attack follows a similar observation of ACDW [3] regarding MD hashing. Given the upper bound of CDG [10] mentioned earlier, this attack is optimal for $B \in \Omega(T)$. For MD hashing, the analogous bound is known to be tight for $B = 2$ and $B \in \Omega(T)$.

A new attack for $B = 1$. We design a new attack for sponge hashing that results with a a single-block collision. Specifically, we show that if $ST^2 > 2^c$, then there is an attack with advantage

$$\Omega\left(\left(\frac{S^2T}{2^{2c}}\right)^{2/3} + \frac{T^2}{2^r}\right).$$

To see why this attack is superior to the previously known one (Eq. 1), we give a setting of parameters where it achieves a significantly higher advantage. Consider $r = c$, $S = 2^{4c/5}$, and $T = 2^{2c/5}$. Indeed, $ST^2 > 2^c$ and therefore we can apply the attack. The previously known best attack (Eq. 1) has advantage

$$\Omega\left(\frac{S + T^2}{2^c}\right) = \Omega\left(\frac{1}{2^{c/5}}\right).$$

This attack is the analog of the *provably* best attack for MD. On the other hand, our new attack has strictly better advantage

$$\Omega\left(\left(\frac{S^2T}{2^{2c}}\right)^{2/3} + \frac{T^2}{2^c}\right) = \Omega\left(\left(\frac{2^{8c/5}2^{2c/5}}{2^{2c}}\right)^{2/3} + \frac{1}{2^{c/5}}\right) = \Omega\left(1\right).$$

Thus, at least in this range of parameters, we beat the state-of-the-art attack and show that sponge is *less secure than MD*. In the example above, we chose a setting of parameters where the gap between the attacks is the largest (our attack succeeds with *constant* probability, while the previously known one succeeds with exponentially small probability). However, there are many more concrete settings where our attack is superior, although the gap could be less dramatic. We note that our bounds in this section and the technical overview are simplified for ease of parsing and refer the reader to the technical sections for the exact bounds.

Conceptual novelty: Our attack for $B = 1$ use the famous time-space tradeoffs for function inversion of Hellman [23] and its extension by Fiat-Naor [17]. We leverage the possibility of inverse queries to the underlying permutation Π in the random-permutation model. This is in contrast to Merkle-Damgård construction which is analyzed in the random-oracle model that does not permit inverse queries. At a very high level, we use time-space tradeoffs for function inversion to "invert" the function Π^{-1} on a restricted domain. We view this conceptual connection between time-space tradeoffs for collision resistance of sponge hashing and function inversion as novel and hope that it will lead to better designs and additional attacks in the future.

Lower Bounds

We complement the picture by showing "lower bounds", namely impossibility results for better attacks. (In other words, these are upper bounds on the best possible advantage of any attacker.) We prove two such lower bounds, one for the case where $B = 1$ and the other is for $B = 2$, corresponding to our attacks.

On optimal attacks for $B = 2$. We show that any attack for $B = 2$ must have advantage

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

We note that this bound is tight with the best known attacks for a large range of parameters, but there still may be a gap otherwise. Specifically, if $ST^3 \leq 2^c$, then the above bound simplifies to $O(ST/2^c + T^2/2^{\min\{c,r\}})$ which matches the attack from Eq. (2). Thus, any improvement on the generic attack from Eq. (2) must take advantage of the regime where $ST^3 > 2^c$.

The proof of this result *provably* cannot be obtained via the bit-fixing method. Rather, we obtain the result via a compression argument. In such arguments, an imaginary adversary that is successful too often is used to compress a uniformly random string, a task which is (information-theoretically) impossible. The compression technique has been instrumental in proving lower bounds in computer science (see the survey of Morin et al. [28]). It has become useful in the context of cryptographic constructions and primitives, starting with the work of Gennaro and Trevisan [20]. Unfortunately, one common "feature" of such proofs is that they tend to be extremely technical and involved. Our proof is no different; in fact, it is even much more complicated than the analogous result for $B = 2$ of ACDW [3] since we work in the RPM and need to handle inverse queries.

On optimal attacks for $B = 1$. We show that any attack for $B = 1$ must have an advantage

$$O\left(\frac{ST}{2^c} + \frac{T^2}{2^r}\right).$$

The proof of this result is relatively straightforward by using an optimized version of the remarkable bit-fixing (or presampling) method [31,11,10]. The main point of distinction of our proof from most previous ones is that we need to apply this technique in the RPM context, so our argument needs to handle inverse queries. (This result might have been known before, but we could not find such a statement, so we give it for completeness.)

We summarize our main results as well as the known best bounds in Fig. 1.

1.3 Future Directions

Our work is the first to address the question of characterizing the complexity of a preprocessing attacker in finding a B-block collision in a Sponge hash function.

	Best Attack	Advantage Upper Bound
$B = 1$	$\min\left(\frac{S^2T^2}{2^{2c}}, \left(\frac{S^2T}{2^{2c}}\right)^{2/3}\right) + \frac{S}{2^c} + \frac{T^2}{2^r}$ [Thm 2]	$\frac{ST}{2^c} + \frac{T^2}{2^r}$ [Thm 4]
$B = 2$	$\frac{ST}{2^c} + \frac{T^2}{2^{\min\{c,r\}}}$ [Thm 1]	$\frac{ST}{2^c} + \frac{S^2T^4}{2^{2c}} + \frac{T^2}{2^{\min(c,r)}}$ [Thm 5]
$B \geq 3$	$\frac{STB}{2^c} + \frac{T^2}{2^{\min\{c,r\}}}$ [Thm 1]	$\frac{ST^2}{2^c} + \frac{T^2}{2^r}$ [10]

Fig. 1: A summary of the attacks and advantage upper bounds for finding B-block collisions for the Sponge hash function. All bounds are given ignoring $\mathrm{poly}(c, r)$ terms. We note that the attack for $B = T$ is implicitly claimed in [10] based on [11].

Our results raise many natural open problems on both the attacks side and lower bounds side. Regarding attacks, we have shown, somewhat surprisingly, that there is a non-trivial attack for $B = 1$ that takes advantage of inverse queries in a novel way. We hope that these ideas can be pushed forward to obtain even better attacks for $B = 1$ or beyond. Specifically, is it possible to beat the $ST/2^c$ attack for $B = 2$ in some range of parameters? In ruling out possible attacks, it would be interesting to come up with a tight upper bound on the advantage for $B = 1$ or $B = 2$. Our work suggests that ruling out attacks that use inverse queries may indeed be a complicated task. In fact, for $B = 3$ we are not aware of any upper bound on the advantage that is better than $O(ST^2/2^c + T^2/2^r)$.

1.4 Related Work

Time-space tradeoffs are fundamental to the existence of efficient algorithms. For example, look-up tables (used to avoid "online" recalculations) have been implemented since the very earliest operating systems. In cryptography (or cryptanalysis), they were first used by Hellman [23] in the context of inverting random functions. Hellman's algorithm was subsequently rigorously analyzed by Fiat and Naor [17] where it was also extended to handle *arbitrary* (not necessarily random) functions. Limitations of such algorithms were studied by Yao [33], and by De, Trevisan, and Tulsiani [15] (building on works by Gennaro and Trevisan [20] and Wee [32]). More limitations were proven by Barkan, Biham, and Shamir [5] but for a restricted class of algorithms. Very recently, Corrigan-Gibbs and Kogan [13] showed complexity-theoretic limitations for improving the lower bound of Yao. While these techniques have mostly cryptographic origins, interesting relations were discovered to other classical problems in other fields (e.g., [1,22]). Time-space tradeoffs have been studied for other problems beyond the ones we mentioned (various cryptographic properties of random oracles, function and permutation inversion, and security of common hashing paradigms). For instance, specific modes for block ciphers (e.g., [18] studied the Even-Mansour

cipher), and various assumptions related to cyclic groups, such as discrete logarithms and Diffie-Hellman problems [27,6,12,10].

On the salt. In the theoretical cryptography literature collision resistance is defined with respect to a family of hash functions indexed by a key. This is important to achieve the standard notion of *non-uniform* security. Indeed, no single hash function can be collision-resistant as a non-uniform attacker can just hardwire a collision. In practice, however, a single hash function is considered by fixing an IV. Thus, the relevance of our model could be questioned. However, often in applications, the hash function used is salted by prepending a random salt value to the input, for example in password hashing [30]. Salting essentially brings us back to the random-IV/keyed setting, where our results become relevant.

2 Technical Overview

In this section, we provide a high-level overview of our techniques. We first describe the generic attack for finding B-block collisions for $B \geq 2$. This attack is a variant of an analogous attack for MD, given by ACDW [3]. We also recall the known best attack for $B = 1$. Then, we describe our new attack for finding 1-block collisions. In particular, our attack outperforms the optimal analogous attacks for MD for specific regimes of parameter settings. Lastly, we overview the techniques used to prove limitations on the best possible attacks for finding short collisions.

Sponge notation. A sponge function is a keyed hash function that takes as input an a c-bit initialization vector IV along with an arbitrary size input and outputs an r-bit string: $\mathsf{Sp}\colon \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^r$. The second input is parsed as a sequence of r-bit blocks, denoted $(m_1, m_2, \ldots)$. On such an input $\mathsf{Sp}(\mathsf{IV}, (m_1, m_2, \ldots))$ is defined as follows. The function Sp is defined relative to a permutation $\mathit{\Pi}\colon \{0,1\}^{r+c} \to \{0,1\}^{r+c}$. An input to or an output of this permutation, denoted $\sigma \in \{0,1\}^{r+c}$, contains an r-bit block, denoted $\sigma[1]$, and a c-bit block, denoted $\sigma[2]$. We sometimes use $(\sigma[1], \sigma[2])$ to mean $\sigma[1] \| \sigma[2] = \sigma$.

On input $m_1, m_2, \ldots, m_\ell$ to Sp, it works as follows:

1. Initialize $\sigma^{(0)} = (\sigma^{(0)}[1], \sigma^{(0)}[2]) = (0, \mathsf{IV})$.
2. For $i = 1, \ldots, \ell$, compute $\sigma^{(i)} = \mathit{\Pi}((\sigma^{(i-1)}[1] \oplus m_i) \parallel \sigma^{(i-1)}[2])$.
3. Output $\sigma^{(\ell)}[1]$.

2.1 Attacks

Generic attack for finding length B collisions. In the preprocessing phase the adversary randomly samples $t \approx S$ different IVs $\mathsf{IV}_1, \ldots, \mathsf{IV}_t$ and for $i = 1, \ldots, t$ it does as follows.

1. Compute $\sigma_{i,j}$ for $j \in [B/2 - 1]$ as $\sigma_{i,j} = \mathit{\Pi}(0, \sigma_{i,j-1}[2])$, where $\sigma_{i,0} = (0, \mathsf{IV})$. The sequence $\sigma_{i,0}, \ldots, \sigma_{i,B/2-1}$ forms a "zero-walk" on IV_i.

2. Find m_i, m_i' such that $\Pi(m_i, \sigma_{i,B/2-1}[2])[1] = \Pi(m_i', \sigma_{i,B/2-1}[2])[1]$.

The preprocessing phase outputs $(\sigma_{i,B/2-1}[2], m_i, m_i')_{i=1,\ldots,t}$. In Fig. 2, we depict the preprocessing phase of the attack. In the online phase, the adversary gets

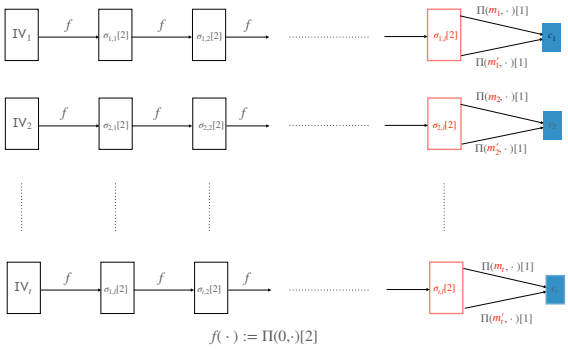


Fig. 2: An illustration of the preprocessing phase of the generic attack. In red, we depict the components that are part of the output of the preprocessing phase. In blue we see the collisions that will be outputted in the online phase if some chain is hit. Notice that we denote $f(\cdot) := \Pi(0, \cdot)[2]$.

a challenge IV as input. For $i = 1, \ldots, T/B$, it computes $\mathsf{IV}_i = \Pi(i, \mathsf{IV})[2]$ (for simplicity, we assume that i is in its bit representation). For each of the IV_i's, it does a zero-walk of length $B-2$. Formally, it sets $\sigma_{i,0} \leftarrow \Pi(i, \mathsf{IV}_i)$ and then for $j = 1, \ldots, B-1$ it does the following.

1. If there is a tuple of the form $(\sigma_{i,j-1}[2], m, m')$ in the preprocessing output, then return

$$(\sigma_{i,0}[1] \,\|\, \ldots \sigma_{i,j-1}[1] \,\|\, m), (\sigma_{i,0}[1] \,\|\, \ldots \sigma_{i,j-1}[1] \,\|\, m').$$

2. Set $\sigma_{i,j} \leftarrow \Pi(0, \sigma_{i,j-1}[2])$.

Correctness is easy to verify. We next discuss the success probability of the adversary. Suppose that the online phase of the adversary computes a $\sigma_{i,j}$ during the first half of any of the T/B zero-walks such $\sigma_{i,j}[2]$ matches the last c bits of one of the $\sigma_{i',j'}$s defined in the preprocessing phase. Then, it is guaranteed to stumble on $\sigma_{i',B/2-1}[2]$ during its zero walk. Hence, in this case, it would output a collision.

Since the adversary encounters roughly $\Omega(SB)$ distinct $\sigma_{i,j}[2]$'s in expectation during the preprocessing phase, this suffices to prove that with probability roughly $\Omega(STB/2^c)$ the online phase will win. The term $\Omega(T^2/2^c + T^2/2^r)$ appears due to birthday-style collisions. We refer the reader to Section 4.1 for details.

Attack for $B = 1$. As described in the introduction, the best attack known so far for $B = 1$ has an advantage of $O(S/2^c + T^2/2^r)$. The analogous attack for MD is provably optimal, as mentioned. However, in contrast to the setting in MD where the ideal object is a random function, here the ideal object is a random permutation, which gives us the additional ability to make inverse queries. This is precisely the leverage that we utilize to get our improved attack. We remark that we are not aware of any prior work that takes advantage of making inverse queries in related contexts.

For $B = 1$, recall that the goal is, given a random IV, to find m, m' such that $\Pi(m, \mathsf{IV})[1] = \Pi(m', \mathsf{IV})[1]$. Our first step is a bit counter-intuitive since we actually aim to solve a *harder* task. Specifically, rather than finding an arbitrary collision, we set out to find a collision on 0, that is, find m and m' such that $\Pi(m, \mathsf{IV})[1] = \Pi(m', \mathsf{IV})[1] = 0$. This step helps us since a natural way to use inverse queries arises, as we argue next.

Main observation: Finding a collision on 0 can be obtained by finding distinct y and y' such that $\Pi^{-1}(0, y)[2] = \Pi^{-1}(0, y')[2] = \mathsf{IV}$.

In other words, it suffices to find two pre-images of IV with respect to the function $f_\Pi \colon \{0,1\}^c \to \{0,1\}^c$ where $f_\Pi(x)$ outputs the last c bits of $\Pi^{-1}(0, x)$. In Fig. 3, we show the partite representations of $\Pi(\cdot)$ and $\Pi^{-1}(0, \cdot)$. Note that while $\Pi(\cdot)$ is a perfect matching, the function $f_\Pi(\cdot) = \Pi^{-1}(0, \cdot)$ has several elements in its co-domain with multiple pre-images.

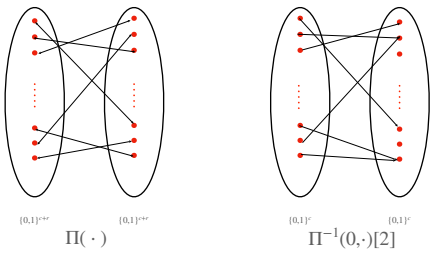


Fig. 3: Partite representation of $\Pi(\cdot)$ and $\Pi^{-1}(0, \cdot)$. Notice that Π is a permutation and thus forms a perfect matching while $\Pi^{-1}(0, \cdot)$ is not a permutation and in expectation a random image will have several pre-images.

At this point, we made some progress: we reduced the problem of collision finding to a function inversion problem (for the function $f_\Pi \colon \{0,1\}^c \to \{0,1\}^c$). Indeed, preprocessing attacks for function inversion have been well studied since the 80's. Hellman [23] described an algorithm that gets S bits of preprocessing on the random function $f \colon \{0,1\}^a \to \{0,1\}^b$ as input and inverts it at a point in its image making T queries to the function. It was later formally analyzed by Fiat and Naor [17] and shown to have advantage $\epsilon(a, b)$ at inverting $y = f(x)$

for a random $x \leftarrow\!\!\$\ \{0,1\}^a$, where

$$\epsilon(a,b) = \Omega\left(\min\left\{1, \frac{ST}{2^{\min(a,b)}}, \left(\frac{S^2T}{2^{2\min(a,b)}}\right)^{1/3}\right\}\right) \tag{3}$$

We are almost done; three technical challenges remain. First, the result of Hellman applies only to random functions. On the other hand, our function is a restriction of a random permutation (which is not a random function). Fiat and Naor [17] showed a clever method to extend Hellman's algorithm to support any function (rather than only random ones), but this improvement is more complicated and comes with a cost in efficiency, which we would like to avoid. To this end, we re-do and adapt the analysis of Hellman to our setting by using the fact that restrictions of permutations are "close enough" to random functions. Our analysis achieves the same parameters as the original one of Hellman, up to constants.

The second problem is that we want to find a pre-image of $\mathsf{IV} \leftarrow\!\!\$\ \{0,1\}^c$ under f_Π, but IV may not even have any pre-images under f_Π, let alone two which are required for our attack. Fortunately, as f_Π is at least "close" to a random function, we can show via a balls-into-bins analysis that a constant fraction of the co-domain will have at least two distinct pre-images. Still, could it be the case that Hellman's attack somehow fails on this fraction of the co-domain? Via a closer analysis of Hellman's attack, we show that for any function $f\colon \{0,1\}^a \to \{0,1\}^b$ and fixed element $y \in \{0,1\}^b$, the attack succeeds at finding a pre-image $x' \in f^{-1}(y)$ with probability $\epsilon(a,b)$ where

$$\epsilon(a,b) = \Omega\left(\frac{1}{b}\min\left(1, \frac{ST \cdot |f^{-1}(y)|}{2^a}, \left(\frac{S^2T \cdot |f^{-1}(y)|^2}{2^{2a}}\right)^{1/3}\right)\right) \tag{4}$$

The last problem we face is that we need to find two *distinct* pre-images for IV. However, applying an inversion algorithm in a black box fashion does not guarantee that distinct inverses will be found. Thus, we also prove that Hellman's inversion algorithm finds a uniform pre-image among all possible pre-images for a given element in the co-domain.

After resolving the above technical challenges, we show that if we run Hellman's attack twice independently for the function f_Π on the image IV, if IV has at least two pre-images (which it does with constant probability), then we will find two distinct pre-images with at least $1/2$ probability times the probability that both attacks succeed. Thus, our overall success probability is roughly $\Omega(\epsilon(c,c)^2)$ where ϵ is defined in (4). We refer the reader to section 4.2 for the details.

2.2 Impossibility Results for Best Attacks

When giving new attacks for finding short collisions, the natural question is how far we can go. In other words, what are the best possible attacks? For $B = 1, 2$ in the case of MD, optimal attacks are known. Our goal here is to prove an

upper bound on the advantage for the best-possible adversary that has S bits of preprocessing as input and can make T queries to Π, Π^{-1} in finding collisions of length 1 and 2 for the sponge construction.

Impossibility result for $B = 1$. We use the pre-sampling technique proposed by [31] and later optimized and adapted to the AI-RPM by [10] to get an advantage upper bound of roughly $O(ST/2^c + T^2/2^r)$. However, we note that this bound does not match the best $B = 1$ attacks, so it is open which side can be improved. Ideally, one could use a compression-based technique as done in [16] to get a tight bound for the $B = 1$ case for MD, but it is not clear how to adapt this argument to handle inverse queries in the AI-RPM model, as we shall see below.

Impossibility result for $B = 2$. The presampling technique of [31,10] *provably* cannot give an advantage upper bound better than $O(ST^2/2^c + T^2/2^r)$ for $B = 2$. Since we can prove this advantage upper bound even for unbounded length collisions, it is natural to ask whether we can prove that 2-block collisions are, in fact, harder to find than collisions of arbitrary length. Aside from presampling techniques, the main technique used to rule out attacks is via a compression argument [20,32], which we turn to for our impossibility result. As a warm up, we first give an overview for the $B = 1$ compression argument for MD from [16] to highlight the key challenges in our setting.

Overview of $B = 1$ compression argument for MD. In a compression argument, the main idea is to use an adversary $\mathcal{A}$ that succeeds at some task involving a random object $\mathcal{O}$, to compress $\mathcal{O}$ beyond what is information theoretically possible. This clearly establishes a contradiction, which gives an upper bound in the success probability of $\mathcal{A}$.

Let $h\colon [N] \times [M] \to [N]$ be a hash function that is modeled as a random oracle, and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an (S, T) adversary that tries to find a 1-block collision in h. $\mathcal{A}_1$ gets h as input and can output S bits of advice σ. $\mathcal{A}_2$ gets σ along with a random salt $a \in [N]$, can make T queries to h, and tries to output $m, m' \in [M]$ such that $h(a, m) = h(a, m')$ and $m \neq m'$. We show that if $\mathcal{A}$ succeeds at this task for many salts a, then we can describe h with fewer bits than possible.

To encode h, we first compute $\sigma \leftarrow \mathcal{A}_1(h)$. We let $G \subseteq [N]$ be the set of elements for which $\mathcal{A}_2$ succeeds on inputs (σ, a) for all $a \in G$. We run $\mathcal{A}_2$ on (σ, a) for all $a \in G$ in lexicographic order. The hope is that whenever $\mathcal{A}_2$ succeeds in finding a collision, we can use the corresponding queries for the collision it makes to compress the function h.

For example, if $\mathcal{A}_2(\sigma, a)$ outputs a collision (m, m'), then we can assume that $\mathcal{A}_2$ must have queried $h(a, m)$ and $h(a, m')$ at some point (we assume without loss of generality that (a, m) was queried before (a, m')). So whenever it queries $h(a, m')$, rather than encoding the output of h, we write down information to indicate that it's the same output as the query for $h(a, m)$. It is easy to see by a counting argument that at least half of the $a \in G$ cannot be queried by $\mathcal{A}_2$ more than $2T$ times. For all such a, we can refer back to the previous query $h(a, m)$

using $\log 2T$ bits, and we use another $\log 2T$ bits to identify the query $h(a, m')$. Thus, we save $\log N - 2\log 2T$ bits per each of these a values in G, which gives a non-trivial compression of h if $T^2 < N$.

The problem with inverse queries. As we stated before, the first major roadblock we encounter when adapting this framework to the AI-RPM is the existence of inverse queries. Let's try to adapt the argument above to the setting of Sponge with 1-block collision. Now the preprocessing adversary $\mathcal{A}_1$ is given a random permutation Π and outputs some state σ with $|\sigma| \leq S$. The online adversary $\mathcal{A}_2$ receives σ and a random IV, and tries to find m, m' such that $\Pi(m, \mathsf{IV})[1] = \Pi(m', \mathsf{IV})[1]$.

Now suppose that $\mathcal{A}_2$ outputs a collision (m, m') with respect to the sponge construction. We can no longer even assume that $\mathcal{A}_2$ queries both $\Pi(m, \mathsf{IV})$ and $\Pi(m', \mathsf{IV})$! For example, it may have first queried $\Pi(m, \mathsf{IV}) = (y, u_1)$ and then queried $\Pi^{-1}(y, u_2) = (m', \mathsf{IV})$. At first glance, this doesn't seem like a problem, we can again note that part of the output of query (y, u_2) is the same as the input to the query for (m, IV). So maybe we can use the same trick as before and instead of storing the whole answer of $\Pi^{-1}(y, u')$, store information indicating that the last c bits of the answer is the same as the input of the query $\Pi(m, \mathsf{IV})$. This intuition is misleading. We can no longer do a counting argument to show that this information is short. It is not clear how to identify the query $\Pi^{-1}(y, u')$ with few bits to be able to point back to the $\Pi(m, \mathsf{IV})$ query. For example, the adversary may just query $\Pi^{-1}(y, *)$ many times and hope to hit IV twice. We hope that this example sheds light on why, at a minimum, inverse queries significantly complicate the situation and deserve extra attention.

Compression for $B = 2$ via multi-instance games. The above compression approach is not known to generalize to the case of $B \geq 2$ collisions for MD. To overcome this limitation, Akshima et al. [3] propose a beautiful framework that gives non-trivial bounds $B \geq 2$ for the case of MD. Their framework reduces the problem to a related "multi-instance" game. In a multi-instance game, the adversary has an arbitrary size string σ of S-bits hard-coded, and its goal is to find a 2-block collision for a set of $u \approx S$ uniformly random a's. The adversary $\mathcal{A}_2$ can make T queries to h when running on each of the u IVs. The key distinctions in this multi-instance game is that (1) the advice sigma that $\mathcal{A}_2$ receives is *independent* of h, and (2) we only need to analyze $\mathcal{A}_2$'s success probability for a random set of u IVs. The core of the proof is a compression argument to upper bound the advantage of this adversary. This framework unfortunately is not strong enough to deal with $B = 1$, as at best it gives the same bound as bit-fixing. However, we adapt this framework to the setting of random permutations to give a non-trivial bound for $B = 2$.

In our case, we need to build a compression argument to compress Π and a set of u random IVs, $\mathsf{IV}_1, \ldots, \mathsf{IV}_u$, (for $u \approx S$) using an adversary $\mathcal{A}_2$ which has some fixed hard-coded advice. $\mathcal{A}_2$ runs on the IVs one by one and succeeds in finding 2-block collisions for all of them. The encoding avoids storing some of

the values of Π explicitly, and instead stores information about the queries of $\mathcal{A}_2$ to Π and Π^{-1} which help during the decoding procedure to recover these particular values of Π.

For $B = 2$ collisions, there are possibly 4 "crucial" queries that the adversary might make that correspond to a 2-block collision (two for each message). We possibly need to consider all combinations of ways that the queries could have been made in either the forward or the reverse direction. For the case of this overview, we zoom in on a single case where inverse queries complicate the situation, and explain how we overcome this.

Suppose on an input IV_j (where $\mathcal{A}_2$ had previously been run on inputs $\mathsf{IV}_1, \ldots, \mathsf{IV}_{j-1}$), $\mathcal{A}_2$ arrives at a collision by making the crucial queries q_1, q_2, q_3, q_4 (not necessarily in that order) such that

1. q_1 was a query to Π on (m_1, IV_j) and returned (x_1, IV'_1)
2. q_2 was a query to Π on (m_2, IV_j) and returned (x_2, IV'_2)
3. q_3 was a query to Π on $(m_3 \oplus x_1, \mathsf{IV}'_1)$ and returned (y, IV'_3)
4. q_4 was a query to Π on $(m_4 \oplus x_2, \mathsf{IV}'_2)$ and returned (y, IV'_4)

Clearly, (m_1, m_3) and (m_2, m_4) hash to the same output and hence are a collision. Now suppose queries q_1 and q_2 were first made during $\mathcal{A}_2(\mathsf{IV}_j)$,[10] while queries q_3 and q_4 were each made previously while running $\mathcal{A}_2$ on an earlier IV_i value. Now, the strategy to compress on the lines of [3] is not to include the last c bits of the answers of q_1, q_2 and the last r bits of the answer of q_4 in the encoding. Instead, we can store the index of the queries q_3, q_4 among all queries (these indices will be in $[uT]$ since there are u IVs and T queries for each of them) and store the indices of the queries q_1, q_2 among the queries made while running $\mathcal{A}_2$ on IV_j (these indices will be in $[T]$). This leads to a saving of roughly $2c + r - 2\log T - 2\log uT$ bits. For reasonable parameters of S, T, c, r, this implies a compression of at least $c - \log uT$ bits, so if $uT \approx ST < 2^c$, this gives non-trivial compression. This implies an upper bound of $ST/2^c$ on the advantage for this case.

However, with inverse queries allowed, things get more complicated. Suppose instead that queries q_3 and q_4 were made in the reverse direction, so $\mathcal{A}_2$ queries $\Pi^{-1}(y, \mathsf{IV}'_3)$ and $\Pi^{-1}(y, \mathsf{IV}'_4)$ prior to running $\mathcal{A}_2(\mathsf{IV}_j)$. In this case, we can still save the c bits from the answers of q_1, q_2. But there is no clear way to save in storing the answer to query q_4 since its answer $(m_4 \oplus x_2, \mathsf{IV}'_2)$ has seemingly no relation to either the answer or input of q_3. So, in this case, we are only able to save $2c - 2\log T - 2\log uT$ bits, which leads to non-trivial compression only if $u^2T^4 \approx S^2T^4 < 2^c$. This implies an upper bound of $S^2T^4/2^{2c}$ on the attacker's advantage for this case. Note that this is actually *better* than the $ST/2^c$ bound we got when considering only forward queries whenever $ST^3 < 2^c$. However, it is important to note that we still need to consider all possible ways in which the attacker may find a collision. We need to show even in the worst case, we can compress Π in order to get an upper bound on the advantage.

[10] Note that this assumption is easy to remove as otherwise we can achieve compression by not including IV_j in the encoding and recovering it from $\mathcal{A}_2$'s queries during decoding

The above highlights just one of the several subtleties that inverse queries introduce in the proof. The ability of the adversary to make queries in two directions makes the encoding and decoding procedures significantly more complicated and lengthy. See Section 5 for full details.

3 Preliminaries

We let $[N] = \{1, 2, \ldots, N\}$ for $N \in \mathbb{N}$ and for $k \in \mathbb{N}$ such that $k \leq N$, let $\binom{S}{k}$ denote the set of k-sized subsets of S. We use $|X|$ to denote the size of a set X and use X^+ to denote one or more elements of X. The set of all permutations on D is denoted by $\mathsf{Perm}(D)$. We let $*$ denote a wildcard element. For example $(*, z) \in L$ is true if there is an ordered pair in L where z is the second element (the type of the wildcard element shall be clear from the context). For a random variable X we use $\mathbb{E}[X]$ to denote its expected value.

We use $x \xleftarrow{\$} \mathcal{D}$ to denote sampling x according to the distribution $\mathcal{D}$. If D is a set, we overload notation and let $x \xleftarrow{\$} D$ denote uniformly sampling from the elements of D. For a bit-string s we use $|s|$ to denote the number of bits in s.

All logarithms in this paper are for base 2 unless otherwise specified.

Sponge-based hashing. For $c, r \in \mathbb{N}$, let $\Pi : \{0,1\}^{c+r} \to \{0,1\}^{c+r}$ be a permutation. We define sponge-based hashing $\mathsf{Sp}_\Pi : \{0,1\}^c \times (\{0,1\}^r)^+ \to \{0,1\}^r$ as follows. For $s \in \{0,1\}^{r+c}$ we use $s[1]$ to denote its first r bits and $s[2]$ to denote its last c bits.

$\mathsf{Sp}_\Pi(\mathsf{IV}, m = (m_1, \ldots, m_B))$
$s_0 \leftarrow 0^r \,\|\, \mathsf{IV}$
For $i = 1, \ldots, B$
  $s_i[1] \,\|\, s_i[2] \leftarrow \Pi\left((m_i \oplus s_{i-1}[1]) \,\|\, s_{i-1}[2]\right)$
Return $s_B[1]$

The elements of $\{0,1\}^r$ shall be referred to as *blocks* and IV refers to the *initialization vector* (also referred to as *salt* in the literature). This is the same abstraction of sponge-based hashing as the one used in [10].

Auxiliary-input Random Permutation Model (AI-RPM). We use the Auxiliary-Input Random Permutation Model (AI-RPM) introduced by Coretti, Dodis and Guo [10] to study non-uniform adversaries in the Random Permutation Model (this was a natural extension of the AI-ROM model proposed by Unruh in [31]). This model is parameterized by two non-negative integers S and T and an adversary $\mathcal{A}$ is divided into two stages $(\mathcal{A}_1, \mathcal{A}_2)$. Adversary $\mathcal{A}_1$, referred to as the preprocessing phase of $\mathcal{A}$ has unbounded access to the random permutation Π and it outputs an S-bit auxiliary input σ. Adversary $\mathcal{A}_2$, referred to as the online phase, gets σ as input and can make a total of T queries to Π, Π^{-1}, and attempts to accomplish some goal involving Π. Formally, we say that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is an (S, T)-AI adversary if $\mathcal{A}_1$ outputs S bits and $\mathcal{A}_2$ issues T queries to its oracles. We next formalize the collision resistance of sponge-based hash functions in AI-RPM.

Game $\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}(\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2))$	Subroutine $\mathsf{AI\text{-}CR}_{\Pi,\mathsf{IV}}(\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2))$
1. $\Pi \leftarrow\!\!\$\ \mathsf{Perm}(\{0,1\}^{c+r})$ 2. $\mathsf{IV} \leftarrow\!\!\$\ \{0,1\}^c$ 3. Return $\mathsf{AI\text{-}CR}_{\Pi,\mathsf{IV}}(\mathcal{A})$	1. $\sigma \leftarrow\!\!\$\ \mathcal{A}_1(\Pi)$ 2. $(\alpha,\alpha') \leftarrow\!\!\$\ \mathcal{A}_2^{\Pi,\Pi^{-1}}(\sigma,\mathsf{IV})$ 3. Return true if: (a) $\alpha \neq \alpha'$, (b) $\lvert\alpha\rvert, \lvert\alpha'\rvert$ are at most B blocks long and (c) $\mathsf{Sp}_\Pi(\mathsf{IV},\alpha) = \mathsf{Sp}_\Pi(\mathsf{IV},\alpha')$ 4. Else, return false

Fig. 4: The bounded-length collision resistance game of salted sponge based hash in the AI-RPM, denoted $\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}$.

Short collision resistance of sponge-based hashing in AI-RPM. We formalize the hardness of bounded-length collision resistance of sponge-based hash functions in the AI-RPM. The game is parameterized by c, r. The game first samples a permutation Π uniformly at random from $\mathsf{Perm}(\{0,1\}^{c+r})$ and IV uniformly at random from $\{0,1\}^c$. Then, $\mathcal{A}_1$ is given unbounded access to Π, and it outputs σ. At this time, $\mathcal{A}_2$ gets σ and IV as input and has oracle access to Π, Π^{-1}. It needs to find $\alpha \neq \alpha'$ such that (1) $\mathsf{Sp}_\Pi(\mathsf{IV},\alpha) = \mathsf{Sp}_\Pi(\mathsf{IV},\alpha')$ and (2) α, α' consist of $\leq B$ blocks from $\{0,1\}^r$. This game, denoted $\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}$, is explicitly written in Fig. 4. In Fig. 4, we write the adversary's execution in its own subroutine only for syntactical purposes (as we shall use it later).

Definition 1 (AI-CR Advantage). *For parameters $c, r, B \in \mathbb{N}$, the advantage of an adversary $\mathcal{A}$ against the bounded-length collision resistance of sponge in the AI-RPM is*

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,B}(\mathcal{A}) = \Pr\left[\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}(\mathcal{A}) = \mathsf{true}\right]$$

For parameters $S, T \in \mathbb{N}$, we overload notation and denote

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,B}(S,T) = \max_{\mathcal{A}} \left\{ \mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,B}(\mathcal{A}) \right\},$$

where the maximum is over all (S,T)-AI adversaries.

The compression lemma. Our proof of the impossibility result for $B = 2$ uses the well-known technique of finding an "impossible compression". The main idea, formalized in the following proposition, is that it is impossible to compress a random element in set $\mathcal{X}$ to a string shorter than $\log|\mathcal{X}|$ bits long, even relative to a random string.

Proposition 1 (E.g., [15]). *Let* Encode *be a randomized map from $\mathcal{X}$ to $\mathcal{Y}$ and let* Decode *be a randomized map from $\mathcal{Y}$ to $\mathcal{X}$ such that*

$$\Pr_{x \leftarrow\!\!\$\ \mathcal{X}}\left[\mathsf{Decode}(\mathsf{Encode}(x)) = x\right] \geq \epsilon.$$

Then, $\log|\mathcal{Y}| \geq \log|\mathcal{X}| - \log(1/\epsilon)$.

4 Attacks

In this section, we first provide the generic attack for finding B-block collisions inspired by the analogous attack for MD in [3]. We then provide our new AI-RPM attack for finding 1-block collision (Section 4.2). Additionally, in Section 4.3, we prove the key lemma for our attack. The key lemma is a preprocessing attack for inverting a function f which is a restricted random permutation. The attack is closely related to that of Hellman [23], but we provide rigorous analysis for our specific application for completeness.

4.1 Generic Attack for B-Block Collisions

We give a (S,T) adversary $\mathcal{A}$ that has advantage $O(STB/2^c + T^2/2^c + T^2/2^r)$ against $\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}$. The main idea for this attack is similar to the zero-walk attack for finding B-block collisions in the Merkle-Damgård construction introduced in [3] which was in turn inspired by an attack in [11].

High level idea. In the preprocessing phase the adversary randomly samples $t \approx S$ different IVs $\mathsf{IV}_1, \ldots, \mathsf{IV}_t$ and for each of them computes $\sigma_{i,j}$ for $j \in [B/2-1]$ as $\sigma_{i,j} = \mathit{\Pi}(0, \sigma_{i,j-1}[2])$, where $\sigma_{i,0} = (0, \mathsf{IV})$. The sequence $\sigma_{i,0}, \ldots, \sigma_{i,B/2-1}$ forms a "zero-walk" on IV_i. It then finds m_i, m_i' such that $\mathit{\Pi}(m_i, \sigma_{i,B/2-1}[2])[1] = \mathit{\Pi}(m_i', \sigma_{i,B/2-1}[2])[1]$ for $i = 1, \ldots, t$. It outputs

$$(\sigma_{i,B/2-1}[2], m_i, m_i')_{i=1,\ldots,t} \ .$$

In the online phase, the adversary gets a challenge IV as input. For $i = 1, \ldots, T/B$, it computes $\mathsf{IV}_i = \mathit{\Pi}(i, \mathsf{IV})[2]$. For each of the IV_i's, it does a zero-walk of length $B-2$. If on any of the walks it hits an IV that the preprocessing phase output then it outputs a collision. The reason this attack achieves an advantage of $\Omega(STB/2^c)$ is because in the preprocessing phase the adversary roughly hits $\Omega(SB)$ distinct IVs and in the online phase if it hits any of these IV's in the first half of its T/B (i.e., in roughly $T/2$ of the queries) walks it finds a collision.

We formally state our result below.

Theorem 1. *Let $S, T, B, c, r \in \mathbb{N}$ such that $SB \leq 2^{c-1}$, $T \leq \min\{(2^{c-1}, 2^{r-1})\}$, $T \geq 2B$. There exists an (S,T) adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that*

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,B}(\mathcal{A}) \geq \left\lfloor \frac{S}{c+2r} \right\rfloor \left\lfloor \frac{B}{2} - 1 \right\rfloor \frac{T}{2^{c+3}} + \frac{(T-B)(T-B-1)}{2^{c+1}} + \frac{3(T-B)(T-B-1)}{2^{r+3}} - \frac{S}{e^{(2^r-1)}} \ .$$

We defer the proof of this theorem to the full version.

4.2 Preprocessing Attack for $B = 1$

We give a new AI-RPM attack for finding 1-block collisions in the Sponge construction. The key ingredient in our attack is an (S, T) adversary for a function f finds two distinct pre-images of a random element of the co-domain under f. We construct this adversary in Lemma 1 based on the adversary from Lemma 2 that finds a single pre-image of a random element of the co-domain under f.

Theorem 2. *Let $c, r \in \mathbb{N}$. For any $S, T \in \mathbb{N}$ such that $S \geq 24c$, $2^c \geq 24S$, and $2^c \geq (S/(T-2)) \cdot 24^3$, there exists an (S, T) attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that on input $\{0,1\}^c$ outputs a valid 1-block collision with probability ϵ, where*

$$\epsilon \geq \left(\frac{1}{20 \cdot 288^2 \cdot c^2}\right) \cdot \min\left(1, \frac{S^2(T-2)^2}{2^{2c+2}}, \left(\frac{S^2(T-2)}{2^{2c+1}}\right)^{2/3}\right).$$

Proof. Let $\Pi \colon \{0,1\}^{c+r} \to \{0,1\}^{c+r}$ be a random permutation. Define the function $f_\Pi \colon \{0,1\}^c \to \{0,1\}^c$ as $f_\Pi(x) = \Pi^{-1}(0^r \,\|\, x)[2]$. Note that f_Π is equivalent to the function that outputs the first c bits of the permutation $\Pi'(x \,\|\, 0^r)$, where $\Pi'(x \,\|\, y)$ for $x \in \{0,1\}^c, y \in \{0,1\}^r$ computes $\Pi^{-1}(y \,\|\, x)$ and shifts the first r bits of the output to the end of its output. Thus, we can invoke Lemma 1 for the function f_Π, which implies an $(S, T-2)$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for finding two distinct pre-images of a random $y \leftarrow\!\!\$\, \{0,1\}^c$. The attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is defined as follows.

- $\mathcal{A}_1(\Pi)$:
  1. Output $\sigma \leftarrow \mathcal{B}_1(f_\Pi)$.
- $\mathcal{A}_2(\mathsf{IV}, \sigma)$:
  1. Compute $(x_1, x_2) \leftarrow \mathcal{B}_1(\mathsf{IV}, \sigma)$.
  2. If $f_\Pi(x_1) = f_\Pi(x_2) = \mathsf{IV}$ and $x_1 \neq x_2$, compute $m_1 \,\|\, \mathsf{IV} = \Pi^{-1}(0^r \,\|\, x_1)$ and $m_2 \,\|\, \mathsf{IV} = \Pi^{-1}(0^r \,\|\, x_2)$ and output (m_1, m_2).
  3. Otherwise, output $\perp$.

For correctness, we note that if $\mathcal{A}_2$ outputs a non-$\perp$ value, then $\mathcal{A}_2$ succeeds in finding a 1-block collision. Recall that $\mathsf{Sp}_\Pi(\mathsf{IV}, m_1) = \Pi(m_1 \,\|\, \mathsf{IV})[1]$ and $\mathsf{Sp}_\Pi(\mathsf{IV}, m_2) = \Pi(m_2 \,\|\, \mathsf{IV})[1]$. By construction, $f_\Pi(x_1) = f_\Pi(x_2) = \mathsf{IV}$ implies $m_1 \,\|\, \mathsf{IV} = \Pi^{-1}(0^r \,\|\, x_1)$ and $m_2 \,\|\, \mathsf{IV} = \Pi^{-1}(0^r \,\|\, x_2)$ for some $m_1, m_2 \in \{0,1\}^r$. But this in turn implies that $\Pi(m_1 \,\|\, \mathsf{IV})[1] = \Pi(m_2 \,\|\, \mathsf{IV})[1] = 0^r$. Since Π is a permutation and $x_1 \neq x_2$, it must be the case that $m_1 \neq m_2$, so (m_1, m_2) is a valid 1-block collision, as required.

Whenever $\mathcal{B}$ succeeds, $\mathcal{A}$ succeeds, so the success probability follows immediately from Lemma 1. ∎

Lemma 1. *Let $n \geq 1$, $a \leq n-3$ and Π be a random permutation over $\{0,1\}^n$. Let $f \colon \{0,1\}^a \to \{0,1\}^a$ such that $f(x)$ consists of the first a bits output by $\Pi(x \,\|\, 0^{n-a})$. For any $S, T \in \mathbb{N}$ such that $S \geq 24a$, $2^a \geq 24S$, and $2^a \geq (S/T) \cdot 24^3$, there exists an (S, T) attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that on input $y \leftarrow\!\!\$\, \{0,1\}^a$*

outputs x_1, x_2 *such that* $f(x_1) = f(x_2) = y$ *and* $x_1 \neq x_2$ *with probability* ϵ*, where*

$$\epsilon \geq \left(\frac{1}{20 \cdot 288^2 \cdot a^2}\right) \cdot \min\left(1, \frac{S^2T^2}{2^{2a+2}}, \left(\frac{S^2T}{2^{2a+1}}\right)^{2/3}\right).$$

Proof. Let $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ be an $(S/2, T/2)$ adversary from Lemma 2. In the offline phase, $\mathcal{A}_1$ on input the function f runs $\mathcal{B}_1(f)$ twice and gets σ_1, σ_2. $\mathcal{A}_1$ outputs $\sigma = (\sigma_1, \sigma_2)$. In the online phase, $\mathcal{A}_2$ on input σ and $y = f(x)$ for $x \leftarrow\!\!\$\ \{0,1\}^a$, computes $x_1 = \mathcal{B}_2(y, \sigma_1)$ and $x_2 = \mathcal{B}_2(y, \sigma_2)$. If $f(x_1) = f(x_2) = y$ and $x_1 \neq x_2$, $\mathcal{A}_2$ outputs (x_1, x_2) and otherwise outputs $\perp$. It directly follows that $\mathcal{A}$ uses space $|\sigma| = |\sigma_1| + |\sigma_2| \leq S$ and makes at most $2 \cdot (T/2) = T$ queries. So it remains to analyze the advantage of $\mathcal{A}$.

We define the following events that are relevant to the analysis. Let $\mathsf{Success}_1$, $\mathsf{Success}_2$ be the events that $\mathcal{B}_1(f, \sigma_1)$ and $\mathcal{B}_1(f, \sigma_2)$ output a valid pre-image, respectively. Let $\mathsf{Inverse}$ be the event that $|f^{-1}(y)| \geq 2$ for the challenge $y \leftarrow\!\!\$\ \{0,1\}^a$. Let $\mathsf{Distinct}$ be the event that the outputs x_1 and x_2 are distinct. Thus, the probability of success is given by

$$\epsilon = \Pr[\mathsf{Success}_1 \wedge \mathsf{Success}_2 \wedge \mathsf{Inverse} \wedge \mathsf{Distinct}].$$

Note that $\mathsf{Success}_1$ and $\mathsf{Success}_2$ are identical and independently distributed given a fixed value for y. Thus, we can rewrite the success probability as

$$\begin{aligned}\epsilon &= \Pr[\mathsf{Inverse}] \cdot \Pr[\mathsf{Success}_1 \mid \mathsf{Inverse}] \cdot \Pr[\mathsf{Success}_2 \mid \mathsf{Inverse}] \\ &\quad \cdot \Pr[\mathsf{Distinct} \mid \mathsf{Success}_1 \wedge \mathsf{Success}_2 \wedge \mathsf{Inverse}]. \\ &= \Pr[\mathsf{Inverse}] \cdot \Pr[\mathsf{Success}_1 \mid \mathsf{Inverse}]^2 \\ &\quad \cdot \Pr[\mathsf{Distinct} \mid \mathsf{Success}_1 \wedge \mathsf{Success}_2 \wedge \mathsf{Inverse}]\end{aligned}$$

We analyze each of these terms separately.

In Claim 3, we show that $\Pr[\mathsf{Inverse}] \geq 1/10$ as long as $a \leq n-3$. $\Pr[\mathsf{Success}_1 \mid \mathsf{Inverse}]$ is given in Lemma 2 using $S' = S/2$ and $T' = T/2$. As $|f^{-1}(y)| \geq 2$ by assumption, it holds that

$$\Pr[\mathsf{Success}_1 \mid \mathsf{Inverse}] \geq \left(\frac{1}{288 \cdot a}\right) \cdot \min\left(1, \frac{ST}{2^{a+1}}, \left(\frac{S^2T}{2^{2a+1}}\right)^{1/3}\right).$$

For the event $\mathsf{Distinct}$, note that in the worst case $|f^{-1}(y)| = 2$. In this case, it is equally as likely that $x_1 = x_2$ compared to $x_1 \neq x_2$ since there are only two equally likely values for x_1, x_2. Thus,

$$\Pr[\mathsf{Distinct} \mid \mathsf{Success}_1 \wedge \mathsf{Success}_2 \wedge \mathsf{Inverse}] = 1/2.$$

Combining the above, we conclude that the attackers probability of success is at least

$$\epsilon \geq \frac{1}{2} \cdot \frac{1}{10} \cdot \left(\frac{1}{288^2 \cdot a^2}\right) \cdot \min\left(1, \frac{S^2T^2}{2^{2a+2}}, \left(\frac{S^2T}{2^{2a+1}}\right)^{2/3}\right),$$

as required. ∎

Claim 3 *Let $n \geq 1$, $a \leq n-3$ and Π be a random permutation over $\{0,1\}^n$. Let $f\colon \{0,1\}^a \to \{0,1\}^a$ such that $f(x)$ consists of the first a bits output by $\Pi(x \,\|\, 0^{n-a})$. Then, $\Pr[y \leftarrow\!\!\$\, \{0,1\}^a : |f^{-1}(y)| \geq 2] \geq 1/10$.*

We provide the proof of this claim in the full versiondue to lack of space.

4.3 Time-Space Tradeoffs for Inverting a Restricted Permutation

In this section, we prove a time-space tradeoff for inverting a restricted permutation. Let $n \in \mathbb{N}$, $a, b < n$, and let $\Pi \leftarrow \mathsf{Perm}\,(n)$ be a randomly chosen permutation. Consider the function $f\colon \{0,1\}^a \to \{0,1\}^b$ defined such that $f(x)$ outputs the first b bits of $\Pi(x \,\|\, 0^{n-a})$. We show that there exists an (S,T) adversary $\mathcal{A}$ that inverts f with advantage roughly $\Omega(\min(1, ST/2^{\min(a,b)}, (S^2T/2^{2\min(a,b)})^{1/3}))$. Additionally, we show that on input $y = f(x)$ for a random $x \leftarrow\!\!\$\, \{0,1\}^a$, $\mathcal{A}$ outputs a uniformly random pre-image $x' \in f^{-1}(y)$ if it succeeds.

We note that our attack closely follows the approach of Hellman [23] and its extension from Fiat and Naor [17]. We provide the full details of the attack and analysis for completeness. We emphasize that our analysis differs from Hellman's analysis since our function f is not quite a random function. Still, we do not need the full generality of the result of Fiat and Naor that works for arbitrary functions. We also note that we show how to instantiate and analyze the "g" functions (see the proof for full details) used in Hellman's attack using only pairwise independence, whereas Fiat and Naor's result for arbitrary functions required k-wise independence for $k \approx T$.

Lemma 2. *Let $n \geq 1$, $a, b \leq n$ and Π be a random permutation over $\{0,1\}^n$. Let $f\colon \{0,1\}^a \to \{0,1\}^b$ such that $f(x)$ consists of the first b bits output by $\Pi(x \,\|\, 0^{n-a})$. For any $S, T \in \mathbb{N}$ such that $S \geq 24\max(a,b)$, $2^{\min(a,b)} \geq 24S$, and $2^{\min(a,b)} \geq (S/T)\cdot 24^3$, there exists an (S,T) attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that succeeds in inverting f on input $y = f(x)$ for $x \leftarrow\!\!\$\, \{0,1\}^a$ with probability ϵ, where*

$$\epsilon \geq \left(\frac{1}{288 \cdot b}\right) \cdot \min\left(1, \frac{ST}{2^{\min(a,b)}}, \left(\frac{S^2T}{2^{2\min(a,b)}}\right)^{1/3}\right).$$

Additionally, the following hold:

- *If the attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ succeeds at inverting $y = f(x)$, it outputs a uniform pre-image $x' \in f^{-1}(y)$ over the randomness of $\mathcal{A}_1$ and $\mathcal{A}_2$.*
- *For any fixed $x \in \{0,1\}^a$ and $y = f(x)$, the attack succeeds with probability at least*

$$\epsilon \geq \left(\frac{1}{288 \cdot b}\right) \cdot \min\left(1, \frac{ST \cdot |f^{-1}(y)|}{2^a}, \left(\frac{S^2T|f^{-1}(y)|^2}{2^{2a}}\right)^{1/3}\right).$$

Due to a lack of space, we defer the proof of this lemma to the full version.

5 Impossibility Results

We next give impossibility results for attacks for 1-block and 2-block collisions for sponge hashing. This consists of upper bounding the best possible advantage of any (S, T) adversary.

5.1 Advantage Upper Bound for B=1

We prove an upper bound for the advantage of an adversary in finding a 1-block collision for the sponge construction. Formally, we prove the following theorem.

Theorem 4. *For all* $S, T, c, r \in \mathbb{N}$

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,1}(S,T) \leq \frac{2(S+c)T+1}{2^c} + \frac{T^2}{2^r} \ .$$

To prove this theorem, we use the result of [10] which relates the advantage upper bound of an adversary in the AI-RPM to that in BF-RPM (bit-fixing RPM). Due to lack of space we defer the preliminaries for BF-RPM, the proof of Theorem 4 and the argument why the bit-fixing technique cannot be used to prove an advantage upper bound for $B = 2$ better than $O(ST^2/2^c)$ to the full version.

5.2 Advantage Upper Bound for $B = 2$

In this section we prove an upper-bound the advantage of an adversary in finding a 2-block collision for the sponge construction in the AI-RPM, according to the game $\mathsf{G}^{\mathsf{ai\text{-}cr}}_{c,r,B}$ described in Fig. 4. First, without loss of generality, in what follows we assume that the adversary is deterministic. This is because we can transform any probabilistic attacker into a deterministic one by hard-wiring the best randomness (see Adleman [2]).

We reduce the task of bounding the advantage of an attacker in finding a 2-block collision in the sponge construction, to a "multi-instance" game where the adversary does not have a preprocessing phase but rather only has non-uniform auxiliary input, chosen *before* the random permutation Π. The latter game is easier to analyze. This is in line with the work of Akshima et al. [3].

We define the following "multi-instance" game $\mathsf{G}^{\mathsf{mi\text{-}cr}}_{c,r,B,u}(\sigma, \mathcal{A}_2)$, where the preprocessing part of the adversary $\mathcal{A}_1$ is degenerate and outputs the fixed string σ. More precisely, the game has the following steps:

1. $\Pi \leftarrow\!\!\$\ \mathsf{Perm}(\{0,1\}^{c+r})$
2. $U \leftarrow\!\!\$\ \binom{\{0,1\}^c}{u}$
3. Define $\mathcal{A}_1$ to be the algorithm that always outputs the string σ.
4. Return true if $\mathsf{AI\text{-}CR}_{\Pi,\mathsf{IV}}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)) = \mathsf{true}$ for every $\mathsf{IV} \in U$. Otherwise, return false.

For a string σ and an adversary $\mathcal{A}_2$, define

$$\mathsf{Adv}^{\mathsf{mi\text{-}cr}}_{\mathsf{Sp},c,r,B,u}(\sigma,\mathcal{A}_2) = \Pr\left[\mathsf{G}^{\mathsf{mi\text{-}cr}}_{c,r,B,u}(\sigma,\mathcal{A}_2)\right].$$

Lemma 3 (Reducing the problem to the multi-instance game). *Fix $c, r, B, S, T, u \in \mathbb{N}$. Then,*

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,B}(S,T) \leq 6 \cdot \left(\max_{\sigma,\mathcal{A}_2}\left\{\mathsf{Adv}^{\mathsf{mi\text{-}cr}}_{\mathsf{Sp},c,r,B,u}(\sigma,\mathcal{A}_2)\right\}\right)^{\frac{1}{u}} + 2^{S-u},$$

where the maximum is taken over all $\sigma \in \{0,1\}^S$ and T-query algorithms $\mathcal{A}_2$.

We refer the reader to [21] for a proof.

We next prove an upper bound on the advantage of any auxiliary-input adversary in finding a 2-block collision for the sponge construction. The main theorem is stated next.

Theorem 5. *For any $c, r, S, T \in \mathbb{N}$ and fixing $\hat{S} := S + c$, it holds that*

$$\mathsf{Adv}^{\mathsf{ai\text{-}cr}}_{\mathsf{Sp},c,r,2}(S,T) \leq \left(2^7 e \cdot \max\left\{\frac{T^2}{2^{c-1}}, \frac{T^2}{2^{r-1}}, \frac{\hat{S}T}{2^{c-3}}, \frac{\hat{S}^2T^4}{2^{2c-2}}\right\}\right) + \frac{1}{2^c}.$$

Theorem 5 follows as a direct corollary of Lemma 3 together with the following lemma, setting $u = S + c$ and observing that **(1)** the lemma holds trivially when $\frac{T^2}{2^{r-1}} > 1$ and **(2)** $\frac{uT^3}{2^{c+r-2}} \leq \frac{uT}{2^{c-1}}$ whenever $\frac{T^2}{2^{r-1}} \leq 1$.

Lemma 4 (Hardness of the multi-instance game). *Fix c, r, T, $u \in \mathbb{N}$ and $\sigma \in \{0,1\}^S$. Then, for any $\mathcal{A}_2$ that makes at most T queries to its oracle, it holds that*

$$\mathsf{Adv}^{\mathsf{mi\text{-}cr}}_{\mathsf{Sp},c,r,2,u}(\sigma,\mathcal{A}_2) \leq \left(2^7 \cdot e \cdot \max\left\{\frac{T^2}{2^{c-1}}, \frac{T^2}{2^{r-1}}, \frac{uT}{2^{c-3}}, \frac{uT^3}{2^{c+r-2}}, \frac{u^2T^4}{2^{2c-2}}\right\}\right)^u.$$

The rest of this section is devoted to the proof of Lemma 4.

We are interested in bounding the advantage of the best strategy, i.e., a pair $(\sigma,\mathcal{A}_2)$ where $\sigma \in \{0,1\}^S$ is a fixed string and $\mathcal{A}_2$ is a T-query algorithm, of finding collisions of length 2 in a sponge with respect to the game $\mathsf{G}^{\mathsf{mi\text{-}cr}}_{c,r,2,u}(\sigma,\mathcal{A}_2)$. Recall that in this game $\mathcal{A}_2$ needs to find proper collisions for u randomly chosen IVs, denoted U. The main idea in the proof is to use any such adversary $(\sigma,\mathcal{A}_2)$ in order to represent the permutation Π *as well as* the set of random IVs U with as few bits as possible. If the adversary is "too good to be true" we will get an impossible representation, contradicting Proposition 1.

Setup. Denote

$$\zeta^* := \log\left(\left(2^5 \cdot 4e \cdot \max\left\{\frac{T^2}{2^{c-1}}, \frac{T^2}{2^{r-1}}, \frac{uT}{2^{c-3}}, \frac{uT^3}{2^{c+r-2}}, \frac{u^2T^4}{2^{2c-2}}\right\}\right)^u \cdot \binom{2^c}{u} \cdot (2^{c+r})!\right).$$

Assume the existence of an adversary $\mathcal{A} = (\sigma, \mathcal{A}_2)$, where $\sigma \in \{0,1\}^S$ is a string and $\mathcal{A}_2$ is a T-query adversary, that contradict the inequality stated in the lemma. That is, there is $\zeta > \zeta^*$ such that

$$\mathsf{Adv}^{\mathsf{mi\text{-}cr}}_{\mathsf{Sp},c,r,2,u}(\mathcal{A}) := \zeta > \zeta^*. \tag{5}$$

Define $\mathcal{G}$ to be the set of permutations-sets of IV pairs for which the attacker succeeds in winning the game for every IV in the set relative to the permutation, That is,

$$\mathcal{G} = \left\{ (U, \Pi) \;\middle|\; \begin{array}{c} U \in \binom{\{0,1\}^c}{u}, \\ \Pi \in \mathsf{Perm}(\{0,1\}^{c+r}), \end{array} \forall \mathsf{IV} \in U : \mathsf{AI\text{-}CR}_{\Pi,\mathsf{IV}}(\mathcal{A}) = \mathsf{true} \right\}.$$

Recall that ζ is defined to be the advantage of $\mathcal{A}$ in the game $\mathsf{G}^{\mathsf{mi\text{-}cr}}_{c,r,2,u}(\mathcal{A})$ in which Π and U are chosen uniformly, and then $\mathcal{A}$ needs to find a collision with respect to every one of the u IVs in U. Therefore,

$$|\mathcal{G}| = \zeta \cdot \binom{2^c}{u} \cdot (2^{c+r})!.$$

In what follows we define an encoding and a decoding procedure such that the encoding procedure gets as input U, Π such that $U \in \binom{\{0,1\}^c}{u}$ and $\Pi \in \mathsf{Perm}(\{0,1\}^{c+r})$, and it outputs an L bit string, where $L = \log\left(\zeta^* \cdot \binom{2^c}{u} \cdot (2^{c+r})!\right)$. The decoding procedure takes as input the string L and outputs U^*, Π^*. It will hold that $U^* = U$ and $\Pi^* = \Pi$ with probability ζ.[11] Using Proposition 1, this would give us that

$$\log \zeta \leq L - \log\left(\binom{2^c}{u} \cdot (2^{c+r})!\right) \Longrightarrow \zeta \leq \zeta^*$$

which is a contradiction to the assumption (see (5)).

Using $\mathcal{A}$, we shall define procedures $\mathsf{Encode}, \mathsf{Decode}$ such that for every $(U, \Pi) \in \mathcal{G}$, $\mathsf{Decode}(\mathsf{Encode}(U, \Pi)) = (U, \Pi)$ and the size of the output of $\mathsf{Encode}(U, \Pi)$ is at most L bits where

$$L = \log\left(\left(2^5 \cdot 4e \cdot \max\left\{\frac{T^2}{2^{c-1}}, \frac{T^2}{2^{r-1}}, \frac{uT}{2^{c-3}}, \frac{uT^3}{2^{c+r-2}}, \frac{u^2T^4}{2^{2c-2}}\right\}\right)^u \cdot \binom{2^c}{u} \cdot (2^{c+r})!\right).$$

Using Proposition 1, this would give us that

$$\epsilon \leq \left(2^7 \cdot e \cdot \max\left\{\frac{T^2}{2^{c-1}}, \frac{T^2}{2^{r-1}}, \frac{uT}{2^{c-3}}, \frac{uT^3}{2^{c+r-2}}, \frac{u^2T^4}{2^{2c-2}}\right\}\right)^u.$$

This immediately gives the bound claimed in the statement of the lemma. The rest of the proof of the lemma would define $\mathsf{Encode}, \mathsf{Decode}$, show an upper bound

[11] Essentially, we will show that for all $(U, \Pi) \in |\mathcal{G}|$, if the encoding procedure produces output L, then the decoding procedure on input L outputs U^*, Π^* such that $U^* = U$ and $\Pi^* = \Pi$.

on the size of the output of Encode and that $\mathsf{Decode}(\mathsf{Encode}(U, \Pi)) = (U, \Pi)$ for all $(U, \Pi) \in \mathcal{G}$.

Notation and Definitions. Fix $(U, \Pi) \in \mathcal{G}$. Let $U = \{\mathsf{IV}_1, \ldots, \mathsf{IV}_u\}$ where the IV_i's are ordered lexicographically. Let $\mathsf{Qrs}(\mathsf{IV}) \in (\{0,1\}^{r+c})^T$ be the list of queries that $\mathcal{A}_2$ makes to Π or Π^{-1} when executed with input (σ, IV). Namely, for $\mathsf{IV} \in \{0,1\}^c$,

$$\mathsf{Qrs}(\mathsf{IV}) = \left\{ s \in \{0,1\}^{c+r} \mid \mathcal{A}_2(\sigma, \mathsf{IV}) \text{ queries } \Pi \text{ or } \Pi^{-1} \text{ on } s \right\}$$

Note that $\mathsf{Qrs}(\mathsf{IV})$ is indeed a set as we can assume (without loss of generality) that $\mathcal{A}_2$ never repeats queries in a single execution (since $\mathcal{A}_2$ can just remember all of its past queries).

Let $\mathsf{Ans}(\mathsf{IV}) \in (\{0,1\}^{r+c})^T$ be the list of answers to the queries of that $\mathcal{A}_2$ to Π or Π^{-1} when executed with input (σ, IV). Namely, for $\mathsf{IV} \in \{0,1\}^c$,

$$\mathsf{Ans}(\mathsf{IV}) = \left\{ s \in \{0,1\}^{c+r} \mid \mathcal{A}_2(\sigma, \mathsf{IV}) \text{ queries } \Pi \text{ or } \Pi^{-1} \text{ on } s \right\}$$

We say that $\mathsf{IV}' \in \mathsf{SIVs}(\mathsf{IV})$ if there is some $s[2] \in \{0,1\}^r$ such that $s[2] \,\|\, \mathsf{IV}'$ is an entry in $\mathsf{Qrs}(\mathsf{IV})$ or $\mathsf{Ans}(\mathsf{IV})$. Namely, for $\mathsf{IV}, \mathsf{IV}' \in \{0,1\}^c$,

$$\mathsf{IV}' \in \mathsf{SIVs}(\mathsf{IV}) \iff \exists s[2] \in \{0,1\}^r \text{ s.t. } s[2] \,\|\, \mathsf{IV}' \in \mathsf{Qrs}(\mathsf{IV}) \cup \mathsf{Ans}(\mathsf{IV}).$$

We define the set of *fresh* IVs in U. An IV IV_i for $i \in [u]$ is called fresh if it was never an IV in either input or output of any query performed by $\mathcal{A}_2$ while being executed on IV_j for $j \leq i-1$ which are fresh. The first IV IV_1 is always fresh. An IV IV_i for $i \geq 2$ is fresh if for any fresh IV_j for $j \leq i-1$, $\mathsf{IV}_i \notin \mathsf{SIVs}(\mathsf{IV}_j)$. Namely, denoting the set of fresh IVs by U_{fresh}, we have the following inductive (on $i \in [u]$) definition:

$$\mathsf{IV}_i \in U_{\mathsf{fresh}} \iff \forall j \leq i-1, \mathsf{IV}_j \in U_{\mathsf{fresh}} \colon \mathsf{IV}_i \notin \mathsf{SIVs}(\mathsf{IV}_j).$$

Looking ahead, we define U_{fresh} like this because we run $\mathcal{A}_2$ on the IVs in U_{fresh} in lexicographical order, and this definition ensures that each IV that $\mathcal{A}_2$ is executed on was not queried by it previously. Denote

$$F := |U_{\mathsf{fresh}}| \quad \text{and} \quad U_{\mathsf{fresh}} = \{\mathsf{IV}'_1, \ldots, \mathsf{IV}'_F\} \text{ (ordered lexicographically).}$$

Denote

$$\forall i \in [F] \colon \mathsf{Q}_i := \mathsf{Qrs}(\mathsf{IV}'_i) \quad \text{and} \quad \mathsf{Q}_{\mathsf{fresh}} := \mathsf{Q}_1 \,\|\, \mathsf{Q}_2 \,\|\, \ldots \,\|\, \mathsf{Q}_F,$$

where $\|$ is the concatenation operator. Let $\mathsf{Q}_{\mathsf{fresh}}[r]$ be the rth query in the list $\mathsf{Q}_{\mathsf{fresh}}$. Note that $r \in [F \cdot T]$. For every $\mathsf{IV} \in U \setminus U_{\mathsf{fresh}}$, let t_{IV} be the minimum value such that $\mathsf{Q}_{\mathsf{fresh}}[t_{\mathsf{IV}}]$ is a query either with input or output of the form $(*, \mathsf{IV})$. Let $b_{\mathsf{IV}} = 0$ if input of $\mathsf{Q}_{\mathsf{fresh}}[t_{\mathsf{IV}}]$ was of the form $(*, \mathsf{IV})$ and 1 otherwise. Define the set of *prediction* queries as

$$\mathsf{P} := \{2t_{\mathsf{IV}} - b_{\mathsf{IV}} \mid \mathsf{IV} \in U \setminus U_{\mathsf{fresh}}\}.$$

The encoding algorithm will output $U_{\mathsf{fresh}}, \mathsf{P}$, which suffices to recover the set U by running $\mathcal{A}_2$.

Structure of collisions. Since adversary $\mathcal{A}_2$ succeeds on all of the IVs in U, it holds that for every $j \in [F]$, the output of the adversary is (α_j, α'_j) such that $\alpha_j \neq \alpha'_j$, $\mathsf{Sp}_{\Pi}(\mathsf{IV}'_j, \alpha_j) = \mathsf{Sp}_{\Pi}(\mathsf{IV}'_j, \alpha'_j)$ and both $\alpha_j \neq \alpha'_j$. We can assume without loss of generality that the last blocks of α_j and α'_j are distinct (because otherwise we can trim α_j, α'_j to obtain a shorter collision).

Definition 2 (Crucial queries). *The queries to Π, Π^{-1} in Q_j include a subset of queries that we call the* crucial queries. *The subset consists of earliest appearing queries in Q_j that are required to compute $\mathsf{Sp}_{\Pi}(\mathsf{IV}'_j, \alpha_j)$ and $\mathsf{Sp}_{\Pi}(\mathsf{IV}'_j, \alpha'_j)$. It follows that for 2-block collisions, this subset consists of at most four queries.*

We say that a query made by running while running on (σ, IV'_j) is NEW if either of the following hold.

- the query is $\Pi(m, \mathsf{IV})$ with answer (m', IV') and neither $\Pi(m, \mathsf{IV})$ or $\Pi^{-1}(m', \mathsf{IV}')$ had been queried by $\mathcal{A}_2$ while running on $\mathsf{IV}'_1, \ldots, \mathsf{IV}'_{j-1}$.
- the query is $\Pi^{-1}(m, \mathsf{IV})$ with answer (m', IV') and neither $\Pi^{-1}(m, \mathsf{IV})$ or $\Pi(m', \mathsf{IV}')$ had been queried by $\mathcal{A}_2$ while running on $\mathsf{IV}'_1, \ldots, \mathsf{IV}'_{j-1}$.

If a query is not NEW we classify it into one of 2 types: REPEATEDUSED, and REPEATEDUNUSED. A REPEATEDUSED query is one such that it was a crucial query for IV'_i where $i < j$. A REPEATEDUNUSED query is one such that it is not a NEW or a REPEATEDUSED query.

Our goal is to compress (U, Π) and we are going to achieve this by using our collision finding adversary $\mathcal{A}_2$. The encoding procedure shall output the set U_{fresh}, the set P, the list $\tilde{\Pi}$ with some entries removed and some additional lists and sets. We will be describing the details of these lists and sets below and which entries we remove from $\tilde{\Pi}$. Our main goal is to show that when we remove entries of $\tilde{\Pi}$ and instead using additional lists and set, we are actually compressing. Our ways to compress will depend on the crucial queries in each Q_j for $j \in [F]$.

We classify the IV'_jth for each $j \in [F]$ into the first of the following cases it satisfies, e.g., if the crucial queries for IV'_j satisfies both cases 1 and 2, we categorize it into 1.

1. One of the crucial queries for IV'_j is a query such that the last c bits of the answer is IV'_j
2. All of the crucial queries to are NEW.
3. At least one of the crucial queries is REPEATEDUSED.
4. There is exactly one REPEATEDUNUSED crucial query.
5. There are exactly two REPEATEDUNUSED crucial queries.

Claim 6 *We claim that each IV'_j will be categorized into one of the above cases.*

Proof. To begin with, observe that given how we define *fresh* IV'_j, there will have to be a NEW crucial query such that either it is a query to Π with input of the form $(*, \mathsf{IV}'_j)$ or it is a query to Π^{-1} with answer of the form $(*, \mathsf{IV}'_j)$, because we have assumed without loss of generality that $\mathcal{A}_2$ makes all the queries while running on IV'_j required to find the collision. If there is a NEW crucial query to Π^{-1} with answer of the form $(*, \mathsf{IV}'_j)$ then case 1 is satisfied. Also observe that there are at most two crucial queries that are not NEW since we are looking at 2-block collisions because any query whose input or output is of the form $(*, \mathsf{IV}'_j)$ is NEW for all IV'_j as they are fresh. So it follows if IV'_j is not categorized into 1, it will either have all NEW crucial queries (case 2) or have at least one REPEATEDUSED query (case 3) or have one (case 4) or two REPEATEDUNUSED (case 5) queries. This proves the claim. ■

Due to a lack of space, we defer all the details of how we handle each of the cases, and achieve the required amount of compression to the full version.

Acknowledgements

Ilan Komargodski is supported in part by an Alon Young Faculty Fellowship, by a JPM Faculty Research Award, by a grant from the Israel Science Foundation (ISF Grant No. 1774/20), and by a grant from the US-Israel Binational Science Foundation and the US National Science Foundation (BSF-NSF Grant No. 2020643). Part of Ashrujit Ghoshal's and Cody Freitag's work was done during an internship at NTT Research. Cody Freitag is also supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–2139899 and DARPA Award HR00110C0086. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Defense Advanced Research Projects Agency (DARPA).

References

1. Abusalah, H., Alwen, J., Cohen, B., Khilko, D., Pietrzak, K., Reyzin, L.: Beyond Hellman's time-memory trade-offs with applications to proofs of space. In: Advances in Cryptology - ASIACRYPT. pp. 357–379 (2017) 9
2. Adleman, L.: Two theorems on random polynomial time. In: 19th Annual Symposium on Foundations of Computer Science (sfcs 1978). pp. 75–83 (1978) 23
3. Akshima, Cash, D., Drucker, A., Wee, H.: Time-space tradeoffs and short collisions in Merkle-Damgård hash functions. In: Advances in Cryptology - CRYPTO. pp. 157–186 (2020) 5, 7, 8, 10, 15, 16, 19, 23
4. Akshima, Guo, S., Liu, Q.: Time-space lower bounds for finding collisions in Merkle-Damgård hash functions. To appear in CRYPTO (2022) 5
5. Barkan, E., Biham, E., Shamir, A.: Rigorous bounds on cryptanalytic time/memory tradeoffs. In: Advances in Cryptology - CRYPTO. pp. 1–21 (2006) 9

6. Bernstein, D.J., Lange, T.: Non-uniform cracks in the concrete: The power of free precomputation. In: Advances in Cryptology - ASIACRYPT. pp. 321–340 (2013) 10
7. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indifferentiability of the sponge construction. In: Advances in Cryptology - EUROCRYPT. pp. 181–197 (2008) 2, 3
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: ECRYPT hash workshop. vol. 2007. Citeseer (2007) 2, 3
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the security of the keyed sponge construction. In: Symmetric Key Encryption Workshop. vol. 2011 (2011) 2
10. Coretti, S., Dodis, Y., Guo, S.: Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In: Advances in Cryptology - CRYPTO. pp. 693–721 (2018) 4, 5, 6, 7, 8, 9, 10, 14, 17, 23
11. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.P.: Random oracles and non-uniformity. In: Advances in Cryptology - EUROCRYPT. pp. 227–258 (2018) 5, 8, 9, 19
12. Corrigan-Gibbs, H., Kogan, D.: The discrete-logarithm problem with preprocessing. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 415–447 (2018) 10
13. Corrigan-Gibbs, H., Kogan, D.: The function-inversion problem: Barriers and opportunities. In: Theory of Cryptography - TCC. pp. 393–421 (2019) 9
14. Damgård, I.: Collision free hash functions and public key signature schemes. In: Advances in Cryptology - EUROCRYPT. pp. 203–216 (1987) 2
15. De, A., Trevisan, L., Tulsiani, M.: Time space tradeoffs for attacks against one-way functions and PRGs. In: Advances in Cryptology - CRYPTO. pp. 157–186 (2010) 9, 18
16. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In: Advances in Cryptology - EUROCRYPT. pp. 473–495 (2017) 5, 14
17. Fiat, A., Naor, M.: Rigorous time/space trade-offs for inverting functions. SIAM J. Comput. **29**(3), 790–803 (1999) 4, 5, 7, 9, 12, 13, 22
18. Fouque, P., Joux, A., Mavromati, C.: Multi-user collisions: Applications to discrete logarithm, even-mansour and PRINCE. In: Advances in Cryptology - ASIACRYPT. pp. 420–438 (2014) 9
19. Gazi, P., Tessaro, S.: Provably robust sponge-based PRNGs and KDFs. In: Advances in Cryptology - EUROCRYPT. pp. 87–116 (2016) 2
20. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: 41st Annual Symposium on Foundations of Computer Science, FOCS. pp. 305–313. IEEE Computer Society (2000) 8, 9, 14
21. Ghoshal, A., Komargodski, I.: On time-space tradeoffs for bounded-length collisions in Merkle-Damgård hashing (2022) 5, 24
22. Golovnev, A., Guo, S., Horel, T., Park, S., Vaikuntanathan, V.: Data structures meet cryptography: 3SUM with preprocessing. In: 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC. pp. 294–307 (2020) 9
23. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theory **26**(4), 401–406 (1980) 4, 5, 7, 9, 12, 19, 22
24. Merkle, R.C.: Secrecy, Authentication and Public Key Systems. Ph.D. thesis, UMI Research Press, Ann Arbor, Michigan (1982) 2
25. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Advances in Cryptology - CRYPTO. pp. 369–378 (1987) 2

26. Merkle, R.C.: A certified digital signature. In: Advances in Cryptology - CRYPTO. pp. 218–238 (1989) 2
27. Mihalcik, J.: An analysis of algorithms for solving discrete logarithms in fixed groups. Tech. rep., Naval Postgraduate School Monterey CA (2010) 10
28. Morin, P., Mulzer, W., Reddad, T.: Encoding arguments. ACM Comput. Surv. **50**(3), 46:1–46:36 (2017) 8
29. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Advances in Cryptology - CRYPTO. pp. 617–630 (2003) 4, 5
30. Sr., R.H.M., Thompson, K.: Password security - A case history. Commun. ACM **22**(11), 594–597 (1979) 10
31. Unruh, D.: Random oracles and auxiliary input. In: Advances in Cryptology - CRYPTO. pp. 205–223 (2007) 4, 5, 8, 14, 17
32. Wee, H.: On obfuscating point functions. In: 37th Annual ACM Symposium on Theory of Computing, STOC. pp. 523–532 (2005) 9, 14
33. Yao, A.C.: Coherent functions and program checkers (extended abstract). In: STOC. pp. 84–94 (1990) 4, 5, 9