The Query-Complexity of Preprocessing Attacks[⋆]

Ashrujit Ghoshal and Stefano Tessaro

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, Washington, USA
ashrujit@cs.washington.edu,tessaro@cs.washington.edu

Abstract. A large number of works prove lower bounds on space-time trade-offs in preprocessing attacks, i.e., trade-offs between the size of the advice and the time needed to break a scheme given such advice. We contend that the question of how much *time* is needed to produce this advice is equally important, and often highly non-trivial. However, this question has received significantly less attention. In this paper, we present lower bounds on the complexity of preprocessing attacks that depend on both offline and online time. As in the case of space-time trade-offs, we focus in particular on settings with ideal primitives, where both the offline and online time-complexities are approximated by the number of queries to the given primitive. We give generic results that highlight the benefits of salting to generically increase the offline costs of preprocessing attacks. The majority of our paper presents several results focusing on *salted* hash functions. In particular, we provide a fairly involved analysis of the pre-image- and collision-resistance security of the (two-block) Merkle-Damgård construction in our model.

1 Introduction

Preprocessing attacks leverage a suitably pre-computed *advice* string that only depends on some underlying primitive (e.g., a hash function, a block cipher, or an elliptic curve) to break a scheme using fewer resources than the best attack without such advice. For example, Hellman's [Hel80] seminal work shows that, for a given permutation $\pi : [N] \to [N]$, one can compute a suitable S-bit advice that allows inverting the permutation on any point in time $T \approx N/S$.

A number of works, starting from [Unr07,DTT10,DGK17,CDGS18,CDG18], prove lower bounds that establish inherent trade-offs between the *size* of the advice and the *online* time needed to break the scheme (often referred to as "space-time trade-offs"). A question that has received less attention, however, concerns the study of trade-offs between the *offline* time needed to compute the advice and the online time. To the best of our knowledge, the only such result, due to Corrigan-Gibbs and Kogan [CK18], focuses on the discrete logarithm problem in the generic-group model.

Initiating a more comprehensive and grounded study of such *offline-online time trade-offs* is the main goal of this paper. As in prior works on space-time trade-offs, we focus on proving lower bounds relying on ideal primitives, thus approximating time with query-complexity. In addition to providing a generic overview of how salting makes preprocessing attacks expensive, we revisit under a new lens the recent works [CDGS18,ACDW20,AGL22,GK22] on preprocessing attacks against *salted* hash functions and the Merkle-Damgård construction.

Why does time-complexity matter? One main reason to study preprocessing attacks is to model non-uniform security. In this case, it is indeed irrelevant *how long* it takes to compute a good advice string since its mere *existence* suffices for an attack, although such an attack may well never be explicitly found. This perspective has emerged from the debate around the use of non-uniformity in security [KM12,BL13], although often the issue can be bypassed entirely via cleverer uniform reductions, as in e.g. [GPR14]. The importance of non-uniform attacks in practice has also been a source of debate [Rog06].

Here, we are concerned with a more practical and less formal perspective where preprocessing is used in practical, explicit attacks for one of two reasons:

[⋆] A preliminary version of this paper appears in the proceedings of CRYPTO 2023. This is the full version.

1. An attack needs to run very quickly in the online stage (e.g., must succeed before a time-out occurs in an Internet protocol) but can afford to run much slower in an offline stage. For instance, Adrian et al. [ABD$^+$15] use offline computation to break 512-bit finite-field discrete logs in less than a minute of online time, hence compromising legacy versions of the Diffie-Hellman handshake.
2. The advice is computed once and for all and is re-used to attack multiple instances. This is the scenario of *Rainbow Tables* [Oec03], which leverage Hellman-type trade-offs to speed up attacks against unsalted password hashes.

In both cases, it is imperative that the offline time remains within a feasible range.

When is preprocessing worth it? Different preprocessing attacks exhibit very different offline-online time trade-offs. The main goal is to ensure that, thanks to preprocessing, the online complexity of an attack is better than the best preprocessing-free attack. For example, in Rainbow Tables, for a password dictionary of size N, the preprocessing takes time $T_1 \approx N$ to produce advice of size S, for which the online complexity is then $T_2 \approx N/S$. The online complexity bests the optimal online-only attacks, which is $\Omega(N)$; moreover, the preprocessing time is optimal since the *sum* of the offline and online time cannot beat the complexity of the best online only attack.

A more interesting example is finding collisions in the *salted* Merkle-Damgård (MD) construction, as studied in a line of recent works [ACDW20,AGL22,GK22]. Given a (random) compression function $h : [N] \times [M] \to [N]$, the offline phase of the optimal attack for two-block collisions finds S collisions of the form $h(a_i, m_i) = h(a_i, m'_i)$ for $m \neq m'$ and salts $a_1, \ldots, a_S$, for which offline time $T_1 \approx S \cdot \sqrt{N}$ is necessary. Then, the online phase, given the salt a, uses time $T_2 = N/S$ to find m such that $h(a, m) = a_i$ for some i, which yields a collision $m \| m_i$, $m \| m'_i$. This attack achieves the trade-off $T_1 \times T_2 = N^{3/2}$, and the online time beats the naïve Birthday attack whenever $T_1 = \Omega(N)$. It is not clear, however, whether the trade-off is optimal, and this is indeed one of the questions we are addressing below.

Our contributions. This paper initiates an in-depth investigation of the time-complexity of preprocessing attacks, and we focus primarily on salted constructions using hash functions, which is where the most interesting technical questions emerge. In particular:

– **Generic salting.** We propose a generic technique to analyze the common practice of salting to mitigate the effects of preprocessing attacks. We consider a model where every call to the underlying primitive is salted. Qualitatively, our result implies that in most settings, to beat the best online-only attack, one needs to invest an offline effort proportional to compromising the primitive on *every salt.*
– **Concrete bounds for random oracles.** Our generic technique can be combined with a recent work by Jaeger and Tessaro [JT20] to provide concrete quantitative upper bounds on the advantage of an offline-online adversary. These bounds are not always optimal, and we prove more refined bounds. We exemplify this situation by studying the pre-image-resistance and collision-resistance of a salted random oracle.
– **Merkle Damgård construction.** The technical bulk of this paper studies the salted Merkle-Damgård (MD) construction with a random ideal compression function. Here, salting achieves a more limited effect and still allows for non-trivial trade-offs between the offline and online complexity of an attack. We deliver quantitative upper bounds on the advantage of breaking pre-image-resistance and collision-resistance of the two-block salted MD for offline-online adversaries.

Salting defeats preprocessing. We start with a result that generically justifies the practice of salting cryptographic primitives to defeat preprocessing attacks. Such results were proved for space-time-complexity in [CDGS18], but we give an analogue result for offline-online query-complexity.

Concretely, we assume that we have a scheme Π^g that relies on a random function $g : [M] \to [N]$, and that the advantage in breaking Π^g, as a function of the number of queries to g, is a well understood quantity. (In particular, here we assume that the security depends only on the number of queries to g.) Now, we replace $g(\cdot)$ with a *salted* hash function $h(a, \cdot)$, where $h : [S] \times [M] \to [N]$. We aim to quantify security for an attacker $\mathcal{A}$ which, during an offline phase, is allowed to issue T_1 queries to $h(\cdot, \cdot)$. Then, after learning the random salt $a \leftarrow\$ [S]$, $\mathcal{A}$ attacks $\Pi^{h(a,\cdot)}$. In this online stage, $\mathcal{A}$ can issue T_2 queries to $h(\cdot, \cdot)$.

We show that if (roughly) T^* queries to g are needed to break Π^g in the *worst case* with very high probability, then for the attacker $\mathcal{A}$ to succeed with high probability as well, $T_1 \geqslant S \cdot T^*/2$ or $T_2 \geqslant T^*/2$ must hold. In other words, the *only* way to beat the best online-only attack is to invest an amount of preprocessing equivalent to that of breaking the scheme *for every choice of the salt*. At the core of this proof is a simple argument that shows how to build an adversary $\mathcal{B}$ against Π^g, achieving the same advantage as $\mathcal{A}$, with *expected* query-complexity $T_1/S + T_2$.

Quantitative bounds for salted random oracles. The above generic result holds for adversaries achieving high advantage. Overall, we would like to go one step further to obtain quantitative precise upper bounds on the advantage of $\mathcal{A}$ as a function of T_1 and T_2 and to characterize the whole advantage curve. As our first result, we combine the above reduction to an adversary with expected query-complexity with the work by Jaeger and Tessaro [JT20]. This allows us to show that any adversary attempting to break pre-image-resistance of a random oracle with a s-bit salt and n-bit outputs succeeds with probability at most

$$\frac{T_1}{S \cdot N} + \frac{T_2}{N} ,$$

where $N = 2^n$, $S = 2^s$ and, once again, T_1 and T_2 are the offline- and online-query-complexity. This bound ends up being nearly *exact* in that there are offline- and online-only attacks achieving each of the two terms.

Unfortunately, the same approach via [JT20] yields only suboptimal bounds for other properties, such as the collision resistance of a salted random oracle. Here, we give a direct proof that shows a bound of order

$$\frac{T_1}{S \cdot \sqrt{N}} + \frac{T_2^2}{N} .$$

This proof is of independent interest, and uses a compression argument to bound the number of salts for which a collision is found in the preprocessing stage. And indeed, the first term is matched by an attack that finds collisions for $\Omega(\lceil T_1/\sqrt{N}\rceil)$ salts, issuing $\sqrt{N}$ queries for each of these salts.

Trade-offs for Merkle-Damgård. Our first set of results aims to show that salting prevents offline-online trade-offs–the best attack is either fully offline or fully online. However, we show that this is not true if we cannot afford to salt *each* call to a primitive. We focus in particular on the salted Merkle-Damgård (MD) construction [Mer90,Dam90], which has also been central to a recent wave of works in the context of space-time trade-offs [ACDW20,AGL22,GK22]. Here, we are given a compression function $h : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ and a message M that consists of B blocks $M_1, \ldots, M_B \in \{0,1\}^\ell$. To hash M, one sets the initial value y_0 to equal the salt a and computes the final hash y_B by iterating

$$y_i \leftarrow h(y_{i-1}, M_i) .$$

Here, we focus on the case of messages of length at most two, which, as in the case of space-time trade-offs [ACDW20], already captures many of the challenges. (In fact, we believe that going beyond requires significantly new techniques than those we explore in this paper.) For *pre-image-resistance*, we prove a bound (which we show to be tight) of the form (when ignoring constant factors and lower-order terms)

$$\frac{T_2}{N} + \frac{T_1 T_2}{N^2} + \frac{T_1^2}{N^3} .$$

The most interesting term is the middle one: it is leading, e.g., for $T_1 = N^{6/5}$ and $T_2 = N^{4/5}$, and suggests an inherent trade-off between offline and online query-complexities. Indeed, this advantage is (roughly) matched by an actual attack that first evaluates $h(a_i, M)$ on N distinct M's for T_1/N different salts $a_1, \ldots, a_{T_1/N}$. Then, upon learning the value y to invert on, as well as the salt a, the online adversary spends its T_2 queries looking for $M \in \{0,1\}^n$ such that $h(a, M) = a_i$ for some $i \in [T_1/N]$, succeeding with probability $\Omega(T_1 T_2/N)$. Then, given it succeeds, the attacker knows N evaluations of $h(a_i, \cdot)$ and is thus likely able to find $M' \in \{0,1\}^n$ such that $h(a_i, M') = y$. Hence, (M, M') is a pre-image of y.

Collision-resistance of MD. Our most involved result is the analysis of the *collision-resistance* of the two-block MD construction, which in particular relies on a number of sophisticated compression arguments and results in a bound that we know to be only partially tight. Ignoring lower order terms and constant factors, we show a bound of order

$$\frac{T_2^2}{N} + \frac{T_1T_2}{N^{3/2}} + \frac{T_1}{N^{5/4}} + \frac{T_1^2}{N^{7/3}} \ .$$

Here, we show matching attacks for all terms except the last one. This (potential) lack of tightness of the last term is due to our combinatorial analysis of a special type of offline-only attack. Namely, an offline attacker could repeatedly attempt to find a special type of collision called a *diamond* for a (potential) salt a, namely, four (distinct) queries $h(a, x_1) = y_1$, $h(a, x_2) = y_2$, $h(y_1, x_1') = z_1$, $h(y_2, x_2') = z_2$ such that $z_1 = z_2$. If the attacker finds a diamond for k salts, then in the online phase it wins with probability k/N (with no further query). Therefore, this boils down to proving a tail inequality on the number of salts for which a diamond is found with T_1 queries. This is challenging since in the regime $T_1 \gg N$ the combinatorics of random functions are not very well understood. The challenge stems from the fact that the "outer" queries $h(y_1, x_1') = z_1$ and $h(y_2, x_2') = z_2$ in one diamond could, individually, be part of diamonds for different salts. Our proof uses compression arguments to provide a suitable tail bound, but we leave it as an open problem to improve our analysis (or show it is tight).

Combining space and time. In conclusion, we observe that our approach is entirely dual to that of space-time trade-offs. The latter completely ignores the issue of *time* to produce advice, whereas we completely ignore the issue of advice size. The obvious question is whether both can be combined, and we currently lack good techniques to combine space and query-complexity.

Relationship with Multi-Instance Security. We note that a remark in [CK18] observed that a lower bound against multiple-discrete-log algorithms also yields lower bounds on the preprocessing time for discrete-log algorithms with preprocessing (observation attributed to Dan Bernstein). We can extend this approach to relate the advantage of an offline-online adversary with the advantage of an adversary playing a multi-instance game. However, we find that this does not give tight bounds for the advantage of offline-online adversaries that succeed with small (sub-constant) probability. In Appendix C, we illustrate this via an example.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of all natural numbers and $\mathbb{N}_{>0} = \mathbb{N}\backslash\{0\}$. For $N \in \mathbb{N}_{>0}$, let $[N] = \{1, 2, \ldots, N\}$. For a set X, let $|X|$ be its size and X^+ denote one or more elements of X. For a set S and $r \in \mathbb{N}_{>0}$ such that $r \leqslant |\mathsf{S}|$, we denote using $\binom{\mathsf{S}}{r}$ the set of subsets of S with r elements. We denote $\mathsf{Fcs}(\mathsf{D}, \mathsf{R})$ the set of all functions mapping elements in D to the elements of R. Security notions are defined via games; for an example see Fig. 2. The probability that a game G outputs true is denoted using $\Pr[\mathsf{G}]$.

We let $x \leftarrow\!\!\$\ \mathcal{D}$ denote sampling x according to the distribution $\mathcal{D}$. If D is a set, we overload notation and let $x \leftarrow\!\!\$\ \mathsf{D}$ denote uniformly sampling from the elements of D. For a bit-string s we use $|s|$ to denote the number of bits in s. For a random variable X, we use $\mathsf{E}[X]$ to denote its expectation.

Merkle-Damgård. We recall the Merkle-Damgård hashing mechanism. For $n, \ell \in \mathbb{N}_{>0}$, let $h : \{0,1\}^n \times (\{0,1\}^\ell)^+ \to \{0,1\}^n$ be a compression function. We recursively define Merkle-Damgård (MD) hashing $\mathsf{MD}^h : \{0,1\}^n \times (\{0,1\}^\ell)^+ \to \{0,1\}^n$ as

$$\mathsf{MD}^h(a, M) = h(a, M)$$

for $a \in \{0,1\}^n, M \in \{0,1\}^\ell$ and

$$\mathsf{MD}^h(a, (M_1, M_2, \ldots, M_B)) = h(\mathsf{MD}^h(a, (M_1, M_2, \ldots, M_{B-1})), M_B)$$

for $a \in \{0,1\}^n$ and $M_1, \ldots, M_B \in \{0,1\}^\ell$. We refer to a as the *salt*.

The compression lemma. The compression lemma states 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., [DTT10]). *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*

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

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

Markov's inequality. We use Markov's inequality multiple times in this paper. We state it here for the sake of completeness.

Proposition 2. *Let X be a non-negative random variable and $a > 0$. Then*

$$\Pr[X \geqslant a] \leqslant \frac{\mathsf{E}[X]}{a}\,.$$

3 Offline-Online Trade-offs and the Role of Salting

We present some basic facts about offline-online trade-offs and discuss the role of salting. To do this, we define a notational framework that captures the generality of our statements.

3.1 A general framework for offline-online attacks

Games. We formalize security guarantees in cryptography using *games*, which we also use for security proofs. A game G describes an environment an adversary $\mathcal{A}$ can interact with, and the combination of G and $\mathcal{A}$ results in a random experiment $\mathsf{G}(\mathcal{A})$ (we refer to this as $\mathcal{A}$ "playing" the game G) which produces a Boolean output. We also denote this output as $\mathsf{G}(\mathcal{A})$.

Games with ideal primitives. We are interested in a special class of games that depend on an *ideal primitive*, such as a random oracle, random permutation, ideal cipher, etc. We model this via a distribution $\mathcal{I}$ on a set of functions. For example, a *random oracle* with (finite) domain D and range R would be modeled by the uniform distribution on $\mathsf{Fcs}(\mathsf{D},\mathsf{R})$.[1] Similarly, an ideal cipher with key space K and domain X can be modeled as a uniformly chosen function e from the set $\mathsf{Fcs}(\mathsf{K} \times \mathsf{X} \times \{-1, 1\}, \mathsf{X})$ such that $e(k, \cdot, 1)$ is a permutation on X for all $k \in \mathsf{K}$, and $e(k, \cdot, -1)$ is its inverse. We can also model a variant of the generic-group model (GGM) [Sho97] by looking at the uniform distribution of functions $f \in \mathsf{Fcs}(\mathbb{Z}_p \times \mathsf{X} \times \mathsf{X}, \mathsf{X} \times \mathsf{X})$, where $|\mathsf{X}| = p$, and

$$\pi(x, l_1, l_2) = (\phi(x), \phi(\phi^{-1}(l_1) + \phi^{-1}(l_2)))\,,$$

where $\phi : \mathbb{Z}_p \to \mathsf{X}$ is a bijective function.

Games with primitives. An *oracle* game G^π is one where both the adversary $\mathcal{A}$ and the game procedures are given access to an oracle π, from an understood set of possible functions π, which we refer to as *compatible* with the game G. We denote by $\mathsf{G}^\pi(\mathcal{A}^\pi)$ both the experiment where $\mathcal{A}$ plays the game, and is given access to the same π as the game as well as the random variable denoting the output. We say that an (oracle) game G is *compatible* with an ideal primitive $\mathcal{I}$, if the range of $\mathcal{I}$ is a subset of the compatible oracles for G. We write specifically

$$\mathsf{Adv}^{\mathsf{G}}_{\mathcal{I}}(\mathcal{A}) = \Pr[\mathsf{G}^\pi(\mathcal{A}^\pi)] \tag{1}$$

where $\pi \leftarrow\!\!\$\, \mathcal{I}$. One could define a more general notion that permits other advantage formats (e.g., to model distinguishing notions). This is straightforward, and outside the scope of this paper.

[1] As usual, one must be more precise when formally defining a random oracle with $\mathsf{D} = \{0,1\}^*$, but we remain intentionally informal on this front; all of our examples can be assumed to work on a finite domain.

```
Game $\mathsf{pre\text{-}G}^{\pi}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$
$\mathsf{st} \leftarrow \mathcal{A}_1^{\pi}$
Return $\mathsf{G}^{\pi}(\mathcal{A}_2^{\pi}(\mathsf{st}))$
```

```
Game $s\text{-}\mathsf{pre\text{-}G}^{\pi}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$
$\mathsf{st} \leftarrow \mathcal{A}_1^{\pi}$
$a \leftarrow\!\!\$ \{0,1\}^s$
Return $\mathsf{G}^{\pi_a}(\mathcal{A}_2^{\pi}(\mathsf{st}, a))$
```

Fig. 1: Offline-Online security games for an original game G. Left: the unsalted case. Right: the salted case. Here, π is meant to be sampled from a salted ideal primitive $\mathcal{I}_s$, and $\pi_a(\cdot) = \pi(a, \cdot)$, i.e., the primitive with salt fixed to a.

Defining offline-online attacks. With the above formalism, given an oracle game G, we introduce a new oracle game $\mathsf{pre\text{-}G}$, which enhances the original game to model offline-online attacks. Both games preserve compatibility with any oracle. In particular, an adversary $\mathcal{A}$ is split into two parts, the offline adversary $\mathcal{A}_1$ and the online adversary $\mathcal{A}_2$. Initially, in the *offline stage*, $\mathcal{A}_1$ is given access *solely* to the game oracle π. At the end of this stage, $\mathcal{A}_1$ outputs a state st. Then, in the *online stage*, adversary $\mathcal{A}_2$ is initialized with state st and run in the game G. Both $\mathcal{A}_2$ and G are given access to the oracle π. Crucially, the game G might give $\mathcal{A}_2$ additional oracles plus additional initialization values, etc., which are not available in the offline stage. (This is formalized in Figure 1 on the left). We colloquially refer to $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as an offline-online adversary. Further, we say that $\mathcal{A}$ is a (T_1, T_2)-adversary if $\mathcal{A}_1$ makes at most T_1 queries and $\mathcal{A}_2$ makes at most T_2 queries. (Note that $\mathcal{A}_2$ could make additional game-dependent queries, which we would specify separately if necessary.) We overload notation and define advantage in terms of (T_1, T_2) as follows.

$$\mathsf{Adv}_{\mathcal{I}}^{\mathsf{G}}(T_1, T_2) = \max_{(T_1,T_2)\text{-adversaries } \mathcal{A}} \mathsf{Adv}_{\mathcal{I}}^{\mathsf{G}}(\mathcal{A}) \ . \tag{2}$$

Some basic facts. The following elementary fact, while straightforward, establishes some important baselines for when offline-online attacks are interesting. It relies on the basic observation that one can consider $\mathcal{A}_1$ and $\mathcal{A}_2$ to be a single online adversary.

Lemma 1. *Let G be a game compatible with the ideal primitive $\mathcal{I}$. For any (T_1, T_2)-adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{I}}^{\mathsf{pre\text{-}G}}(\mathcal{A}) = \mathsf{Adv}_{\mathcal{I}}^{\mathsf{G}}(\mathcal{B}) \ .$$

Here, $\mathcal{B}$ makes $T_1 + T_2$ queries to the primitive, and its time-complexity is the sum of the time-complexities of $\mathcal{A}_1$ and $\mathcal{A}_2$. Further, for any game-dependent type of query, $\mathcal{B}$ makes the same number of queries as $\mathcal{A}_2$.

For an ideal primitive $\mathcal{I}$, we let $Q(\mathcal{I})$ be an upper bound on the number of queries needed by an adversary, given oracle access to $\pi \leftarrow\!\!\$ \mathcal{I}$, to reconstruct π with probability 1. Then, the following also holds true and formalizes the fact that one can simulate an offline-online adversary by having the offline adversary first reconstruct π and then store it in st.

Lemma 2. *Let $\mathcal{I}$ be a primitive that is compatible with game G. For all offline-online adversaries $\mathcal{A}$, there exists a $(Q(\mathcal{I}), 0)$-adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{I}}^{\mathsf{pre\text{-}G}}(\mathcal{A}) = \mathsf{Adv}_{\mathcal{I}}^{\mathsf{pre\text{-}G}}(\mathcal{B})$$

Note that the adversary $\mathcal{B}$ could be much less efficient than $\mathcal{A}$; however, in many cases, we will study games that only target information-theoretic security, and time-complexity will not matter. It also naïvely follows that there always exists an *optimal* adversary that is a $(Q(\mathcal{I}), 0)$-adversary.

Which games are interesting? Some games are more interesting than others in the context of offline-online trade-offs, and the above two lemmas already provide some guidance.

Consider the problem of inverting a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$. It is well known that the best adversary takes time $\Omega(N)$ queries, where $N = 2^n$, to invert with constant probability. Then, Lemma 1 implies that any (T_1, T_2)-offline adversary needs $T_1 + T_2 = \Omega(N)$ to invert with constant probability. Thus, to get $T_2 = o(N)$ and beat the naïve inversion attack in the online phase, we need $T_1 = \Omega(N)$. Further, we already have an $(N, 0)$-adversary by Lemma 2, so we cannot really expect interesting trade-offs. The question becomes interesting only if we limit the state size between $\mathcal{A}_1$ and $\mathcal{A}_2$, which is exactly what is considered by prior works on space-time trade-offs.

This is in contrast to the setting of the discrete logarithm problem with preprocessing [CK18]. There, for a group of order p, by combining Lemma 1 with the well-known result by Shoup [Sho97], we get that $T_1 + T_2 = \Omega(\sqrt{p})$. Lemma 2 guarantees only a $(p, 0)$ attacker, so we can expect that $T_2 = o(\sqrt{p})$ while still having $T_1 = o(p)$. And indeed, one can achieve the trade-off $T_1 \cdot T_2 \geqslant p$, as indicated in [CK18].

3.2 The power of salting

A special case of interest is that of *salting*, where the cryptographic primitive $\mathcal{I}$ permits an additional input–called a *salt*–that is chosen in the online phase of an attack.

Generic salting. Let $\mathcal{I}$ be an ideal primitive, whose range is a subset of $\mathsf{Fcs}(\mathsf{D}, \mathsf{R})$. We define its s-bit *salted* version, denoted $\mathcal{I}_s$, as the ideal primitive with range $\mathsf{Fcs}(\{0,1\}^s \times \mathsf{D}, \mathsf{R})$; sampling a function $\pi : \{0,1\}^s \times \mathsf{D} \to \mathsf{R}$ occurs by first sampling 2^s independent copies $\pi_a \leftarrow\!\!\$\, \mathcal{I}$ for each $a \in \{0,1\}^s$ and then letting

$$\pi(a, x) = \pi_a(x) \ ,$$

for all $a \in \{0,1\}^s$ and $x \in \mathsf{D}$.

For any game G compatible with an ideal primitive $\mathcal{I}$, we can now define an s-bit salted version of the game, $s\text{-}\mathsf{pre}\text{-}\mathsf{G}$, that is compatible with $\mathcal{I}_s$. This is given on the right of Figure 1. Essentially, we now sample a primitive $\pi \leftarrow\!\!\$\, \mathcal{I}_s$ to which the adversary $\mathcal{A}$ is given access. However, in the online phase, the games themselves have access only to π_a for a randomly sampled salt a, which is revealed to only the adversary $\mathcal{A}_2$.

From salted to unsalted games. The following theorem relates the advantage of an offline-online adversary for the salted game with that of an adversary for the original game. We provide an interpretation below (and use this lemma quantitatively in Section 4).

Theorem 1. *Let G be a game compatible with an ideal primitive $\mathcal{I}$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a (T_1, T_2) offline-online adversary. Then, there exists an adversary $\mathcal{B}$ playing G such that*

$$\mathsf{Adv}^{s\text{-}\mathsf{pre}\text{-}\mathsf{G}}_{\mathcal{I}_s}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{G}}_{\mathcal{I}}(\mathcal{B}) \ .$$

The adversary $\mathcal{B}$ makes a number of queries, expressed as a random variable T with expectation $\mathsf{E}[T] \leqslant T_1/2^s + T_2$.

Proof. Given access to π in the range of $\mathcal{I}$, the adversary $\mathcal{B}$ samples a random salt a from $\{0,1\}^s$ and then samples $\pi_{a'} \leftarrow\!\!\$\, \mathcal{I}$ for $a' \neq a$. It then simulates an execution of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in $s\text{-}\mathsf{pre}\text{-}\mathsf{G}$ as follows: it answers the ideal-primitive query of $\mathcal{A}$ for salt a using oracle queries to π, and those for salt $a' \neq a$ using the local evaluation of $\pi_{a'}$. It is immediate that $\mathcal{B}$ perfectly simulates the execution of $s\text{-}\mathsf{pre}\text{-}\mathsf{G}$ to $\mathcal{A}$. Therefore, $\mathcal{A}$ wins if and only if $\mathcal{B}$ does and the claim about advantages follows.

Observe that $\mathcal{B}$ queries its ideal primitive, sampled from $\mathcal{I}$ only when it receives an ideal object query from $\mathcal{A}$ that is prefixed by the actual salt a. Since it is not given access to a when simulating $\mathcal{A}_1$, its queries are independent of a, and each of them is indeed on salt a with probability $1/2^s$. In contrast, $\mathcal{A}_2$ makes queries after learning a, and therefore, those queries are on salt a with probability upper bounded by one. By linearity of expectation, the total number of queries T to $\pi = \pi_a$ made by $\mathcal{B}$ satisfies $\mathsf{E}[T] \leqslant T_1/|\mathsf{S}| + T_2$, as we intended to show. $\square$

Salting generically defeats preprocessing attacks. We illustrate one first main application of Theorem 1, i.e., the fact that salting generically defeats preprocessing in a *qualitative* sense. Here, "qualitative" means that we only look at the power of attacks that achieve large advantage. Subsequent sections (Sections 4 and 5) take a more quantitative angle on this, studying the whole advantage curve.

We now say that a game G compatible with $\mathcal{I}$ is (T^*, ϵ)-hard if $\mathsf{Adv}_{\mathcal{I}}^{\mathsf{G}}(\mathcal{A}) \leqslant \epsilon$ for all T^*-query $\mathcal{A}^*$. We say that game G is (T^*, ϵ)-expected-hard if the same holds for all adversaries running in *expected* time at most T^*. The following fact is helpful.

Lemma 3. *If* G *is* $(T^*, 0.4)$*-hard, then it is* $(T^*/2, 0.9)$*-expected-hard.*

Proof. By contradiction, let $\mathcal{A}$ run in expected time at most $T^*/2$, and $\mathsf{Adv}_{\mathcal{I}}^{\mathsf{G}}(\mathcal{A}) > 0.9$. Then, build $\mathcal{B}$ that runs $\mathcal{A}$ for T^* queries and then aborts with some default answer if $\mathcal{A}$ did not finish running. Let T be the running time of $\mathcal{A}$. Then, for $\pi \leftarrow\$\ \mathcal{I}$,

$$\begin{aligned}\Pr\left[\mathsf{G}^{\pi}(\mathcal{B}^{\pi})\right] &= \Pr\left[\mathsf{G}^{\pi}(\mathcal{A}^{\pi}) \ \wedge\ T \leqslant T^*\right] \\ &\geqslant \Pr\left[\mathsf{G}^{\pi}(\mathcal{A}^{\pi})\right] - \Pr\left[T > T^*\right] > 0.9 - 0.5 = 0.4 ,\end{aligned}$$

where we used Markov's inequality and the fact that $\mathsf{E}\left[T\right] \leqslant T^*/2$. □

Now, say that s-pre-G is (T_1, T_2, ϵ)-hard if for all (T_1, T_2)-adversaries $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{I}}^{s\text{-pre-}\mathsf{G}}(\mathcal{A}) \leqslant \epsilon$. Then, Lemma 3 and Theorem 1 yield the following corollary.

Corollary 1. *If* G *is* $(T^*, 0.4)$*-hard, then* s-pre-G *is* $(T_1, T_2, 0.9)$*-hard for any* T_1, T_2 *such that* $T_1/2^s + T_2 \leqslant T^*/2$.

This means that if a (T_1, T_2)-adversary is to achieve advantage larger than 0.9 in s-pre-G, then $T_1 \geqslant 2^s T^*/4$ or $T_2 \geqslant T^*/4$. In other words, in order to win s-pre-G with an advantage larger than 0.9, an attacker needs to either use online time which is (almost) as large as that of the best online attack achieving advantage 0.4 or run 2^s times that amount of time in the offline stage.

Moving on. We can easily revisit the remainder of this paper using what we saw in this section. First of all, our conclusion about salting applies only to large advantage adversaries since otherwise we cannot prove an analogue of Lemma 3. Section 4 examines *tight* exact bounds on the advantage of (T_1, T_2)-adversaries that hold for each choice of T_1 and T_2. Second, this conclusion applies only to the case where G salts *every call* to the primitive. In Section 5, we characterize the pre-image- and the collision-resistance of the salted MD construction against offline-online attacks and show that salting, while still useful, has a more limited effect.

4 Offline-Online Security of Salted Random oracles

In this section, we study the security of salted monolithic random oracles against offline-online adversaries. Specifically, we consider the security properties of pre-image-resistance and collision-resistance. Our analysis begins by applying Theorem 1 in conjunction with [JT20, Theorem 1] to derive advantage upper bounds for offline-online adversaries against these properties. This approach already yields a tight bound for pre-image-resistance, but not for collision-resistance. We then use a non-generic technique to prove a tight bound for the latter.

4.1 Pre-image-resistance of a salted random oracle

Oracle game PR^h in Fig. 2 formalizes the preimage-resistance of oracle h, which has co-domain $\{0,1\}^n$. In the game the adversary is given as input y, which is randomly sampled from $\{0,1\}^n$. It has oracle access to h and wins if it manages to output x such that $h(x) = y$.

We aim to upper bound the advantage of offline-online adversaries $\mathcal{A}$ against pre-image-resistance of salted random oracles. Let $H_{s,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^s \times \{0,1\}^\ell, \{0,1\}^n)$. The quantity of interest is $\mathsf{Adv}_{H_{s,\ell,n}}^{s\text{-pre-PR}}(T_1, T_2)$. Using Theorem 1 and [JT20, Theorem 1], we get the following corollary.

```
Game $\mathsf{PR}^h(\mathcal{B})$
$x \leftarrow_\$ \{0,1\}^*$
$y \leftarrow h(x)$
$x' \leftarrow \mathcal{B}^h(y)$
If $h(x') = y$:
    Return true
Return false
```

```
Game $\mathsf{CR}^h(\mathcal{A})$
$(M, M') \leftarrow \mathcal{A}^h$
If $M \neq M'$ and $h(M) = h(M')$
    Return true
Return false
```

Fig. 2: Left: Oracle Game PR^h for preimage-resistance of oracle h. Right: Oracle Game CR^h for collision-resistance of oracle h.

Corollary 2. *Let $T_1, T_2, n, s, \ell \in \mathbb{N}_{>0}$. Then*

$$\mathsf{Adv}^{s\text{-pre-PR}}_{H_{s,\ell,n}}(T_1, T_2) \leqslant 5\left(\frac{T_1}{2^{s+n}} + \frac{T_2}{2^n}\right) .$$

Proof. We fix the adversary (T_1, T_2)-adversary $\mathcal{A}$ that maximizes $\mathsf{Adv}^{s\text{-pre-PR}}_{H_{s,\ell,n}}(T_1, T_2)$. From Theorem 1 we have that there exists an adversary $\mathcal{B}$ that makes at most T queries to its h oracle, where $\mathsf{E}[T] \leqslant T_1/2^s + T_2$ and

$$\mathsf{Adv}^{s\text{-pre-PR}}_{H_{s,\ell,n}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{PR}}_{H_{s,\ell,n}}(\mathcal{B}) .$$

Using Theorem 1 in [JT20], we can show that $\mathsf{Adv}^{\mathsf{PR}}_{H_{s,\ell,n}}(\mathcal{B}) \leqslant \frac{5\mathsf{E}[T]}{2^n}$, which completes the proof.

Tightness. We remark that this bound is tight up to constant factors. To see the tightness of the term $T_2/2^n$, consider the online-only adversary that simply makes k distinct queries with the salt a. It fails only if all the queries have answer different from y, which happens with probability $(1 - 1/2^n)^{T_2} \leqslant e^{-T_2/2^n}$. Since $e^{-x} \leqslant 1 - x/2$ for $x \leqslant 1.5$, for $T_2 \leqslant 2^n$, $e^{-T_2/2^n} \leqslant 1 - T_2/2^{n+1}$. This means the adversary succeeds with probability at least $T_2/2^{n+1}$, meaning the second term in the bound is tight up to constant factors.

To see why the first term is tight, consider the adversary $\mathcal{A}_1$ which makes 2^n queries on different inputs for $k = T_1/2^n$ different salts (where T_1 is a multiple of 2^n). In the online phase, it simply checks whether it had made a query with salt a, that had output y; if so it returns the query input.

Let the set of k salts $\mathcal{A}_1$ had made queries on be S. For each salt in S, the probability that $\mathcal{A}_1$ had not made query with that salt that had answer y is at most $(1 - 1/2^n)^{2^n} \leqslant 1/e$. So, for each salt in S with probability at least $(1 - 1/e)$, $\mathcal{A}_1$ had made a query that had answer y. Now $\mathcal{A}$ wins if the a sampled is in S, and $\mathcal{A}_1$ has made a query with salt a that had answer y; this probability is at least $(1 - 1/e)k/2^s$ since a is sampled at random. Since $k = T_1/2^n$, it follows that the second term in the bound is tight as well.

4.2 Collision-resistance of a salted random oracle

Oracle game CR^h in Fig. 2 formalizes the collision-resistance of oracle h. In the game the adversary has oracle access to h and wins if it manages to output M, M' such that $M \neq M'$ and $h(M) = h(M')$.

We aim to upper bound the advantage of offline-online adversaries $\mathcal{A}$ against collision-resistance of salted random oracles. Let $H_{s,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^s \times \{0,1\}^\ell, \{0,1\}^n)$. We seek to tightly upper bound $\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2)$. Using Theorem 1 and [JT20, Theorem 1] we get the following corollary.

Corollary 3. *Let $T_1, T_2, n, s, \ell \in \mathbb{N}_{>0}$. Then*

$$\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2) \leqslant 5\sqrt{2}\left(\frac{T_1}{2^{s+n/2}} + \frac{T_2}{2^{n/2}}\right) .$$

Proof. We fix the adversary (T_1, T_2)-adversary $\mathcal{A}$ that maximizes $\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2)$. From Theorem 1 we have that there exists an adversary $\mathcal{B}$ such that it makes at most T queries to its h oracle, where $\mathsf{E}[T] \leqslant T_1/2^s + T_2$, and

$$\mathsf{Adv}^{s\text{-pre-PR}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{PR}}_{H_{s,\ell,n}}(\mathcal{B}) \ .$$

Using [JT20, Theorem 1], we can show that $\mathsf{Adv}^{\mathsf{CR}}_{H_{s,\ell,n}}(\mathcal{B}) \leqslant 5\sqrt{\frac{2\mathsf{E}[T]^2}{2^n}}$, which concludes the proof.

Tight bound. The bound in corollary 3 is suboptimal. In Theorem 2, we obtain a better bound for $\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2)$.

Theorem 2. *Let $n, s, \ell, T_1, T_2 \in \mathbb{N}_{>0}$. Let $H_{s,\ell,n}$ be the uniform distribution on $\mathsf{Fcs}(\{0,1\}^s \times \{0,1\}^\ell, \{0,1\}^n)$. Then, we have that*

$$\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2) \leqslant \frac{\binom{T_2}{2}}{2^n} + \frac{T_2 T_1}{2^{s+n}} + \frac{eT_1}{2^{s+n/2}} + \frac{n}{2^{s+1}} + \frac{1}{2^n} \ .$$

Tightness. We argue that this bound is tight up to constant factors. Initially, observe that for $T_2 \geqslant 2^{n/2}$, the right side becomes greater than one, and the bound always holds. For $T_2 \leqslant 2^{n/2}$, we have that $\frac{T_1 T_2}{2^{n+s}} \leqslant \frac{eT_1}{2^{s+n/2}}$. Therefore, the term $\frac{T_1 T_2}{2^{n+s}}$ is never the dominant term in the bound, and it suffices to show attacks that achieve advantage of the order $\frac{\binom{T_2}{2}}{2^n}$ and $\frac{eT_1}{2^{s+n/2}}$ to show that this bound is tight. A birthday style attack with T_2 queries achieves advantage of the order $\frac{\binom{T_2}{2}}{2^n}$. Finally, we prove the following theorem to show that term $\frac{eT_1}{2^{s+n/2}}$ is tight up to constant factors.

Theorem 3. *Let $T_1, s, n, \ell \in \mathbb{N}_{>0}$ such that n is a multiple of 2 and T_1 is a multiple of $2^{n/2+1}$. Let $H_{s,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^s \times \{0,1\}^\ell, \{0,1\}^n)$. Then there exists a $(T_1, 0)$-adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(\mathcal{A}) \geqslant \frac{(1-1/e)T_1}{2^{s+n/2+1}} \ .$$

We defer the formal proof of this theorem to Appendix A.3.We next prove Theorem 2.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be the (T_1, T_2)-offline-online adversary that maximizes $\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(T_1, T_2)$. We can treat $\mathcal{A}$ as deterministic by fixing its randomness that maximizes its advantage.

We seek to upper bound the probability that the adversary $\mathcal{A}$ finds a one-block collision for the randomly chosen salt a that it gets as input in its online phase. We can assume without loss of generality that if $\mathcal{A}$ outputs a collision, it must have made the relevant queries either in the offline or the online phase. This is without loss of generality because if $\mathcal{A}$ does not make one of these queries, we can construct a $(T_1, T_2 + 2)$-offline-online adversary $\mathcal{A}'$ that does whatever $\mathcal{A}$ does, and at the end of its online phase, makes the two relevant queries if not made earlier after $\mathcal{A}$ outputs M, M'. The term T_2 would then be replaced by $T_2 + 2$ in our bounds; for ease of readability, we omit this.

Also, without loss of generality we can assume that no query across the offline and online phases is repeated because the adversary can simply remember the query answer since we do not restrict its memory or the amount of advice it can pass on from offline to the online phase.

We define the following three events.

1. $\mathsf{onecoll_{on}}$: $\mathcal{A}_2$ makes two queries $h(a, M) = z$, $h(a, M') = z$ for some $M \neq M'$
2. $\mathsf{onecoll_{offon}}$: $\mathcal{A}_1$ makes a query $h(a, M) = z$, and $\mathcal{A}_2$ makes a query whose answer is z
3. $\mathsf{onecoll_{off}}$: $\mathcal{A}_1$ makes two queries $h(a, M) = z$, $h(a, M') = z$ for some $M \neq M'$

Observe that if none of $\mathsf{onecoll_{on}}, \mathsf{onecoll_{offon}}, \mathsf{onecoll_{off}}$ happen, $\mathcal{A}$ cannot find a collision. We have that

$$\Pr\left[s\text{-pre-}\mathsf{CR}^h_{H_{s,\ell,n}}(\mathcal{A})\right] \leqslant \Pr\left[\mathsf{onecoll_{on}}\right] + \Pr\left[\mathsf{onecoll_{offon}}\right] + \Pr\left[\mathsf{onecoll_{on}}\right] \ . \quad (3)$$

We upper bound the probability of these three events one by one.

UPPER BOUNDING $\Pr[\mathsf{onecoll_{on}}]$. This event happens only if $\mathcal{A}_2$ makes two queries that collide. The probability of any two queries of $\mathcal{A}_2$ colliding is $1/2^n$. Using a union bound over all pairs of queries of $\mathcal{A}_2$, we have

$$\Pr[\mathsf{onecoll_{on}}] \leqslant \frac{\binom{T_2}{2}}{2^n} . \tag{4}$$

UPPER BOUNDING $\Pr[\mathsf{onecoll_{offon}}]$. Observe that $\mathsf{onecoll_{offon}}$ happens only if there is an online query that has the same answer as one of the offline queries that had input salt a. There are a total of T_1 offline queries, and a is random. Therefore, the expected number of offline queries with salt a is $T_1/2^s$. We have that

$$\begin{aligned}\Pr[\mathsf{onecoll_{offon}}] &\leqslant \sum_{k=0}^{T_1} \Pr[\text{There are } k \text{ offline queries with salt } a] \frac{kT_2}{2^n} \\ &= \mathsf{E}[\text{Number of offline queries with salt } a] \frac{T_2}{2^n} \\ &= \frac{T_1 T_2}{2^{s+n}} . \end{aligned} \tag{5}$$

UPPER BOUNDING $\Pr[\mathsf{onecoll_{off}}]$. The main challenge of this proof is proving an upper bound on $\Pr[\mathsf{onecoll_{off}}]$. We do it as follows: we define an event $\mathsf{off\text{-}oneblk\text{-}}k$ since there are k different salts for which a one-block collision has been found in the offline phase. We have that for any k,

$$\Pr[\mathsf{onecoll_{off}}] \leqslant \Pr\left[\mathsf{onecoll_{off}} \,\middle|\, \neg\mathsf{off\text{-}oneblk\text{-}}k\right] + \Pr[\mathsf{off\text{-}oneblk\text{-}}k] . \tag{6}$$

Since $\mathsf{onecoll_{off}}$ happens if for the salt a that is chosen uniformly at random from $\{0,1\}^s$, $\mathcal{A}_1$ had queried $h(a, M), h(a, M')$ that have the same answer, we have that $\Pr\left[\mathsf{onecoll_{off}} \,\middle|\, \neg\mathsf{off\text{-}oneblk\text{-}}k\right] \leqslant k/2^s$. We upper bound $\Pr[\mathsf{off\text{-}oneblk\text{-}}k]$ using a compression argument.

The encoding procedure encodes the random oracle h as follows.

1. It runs $\mathcal{A}_1^h$ and initializes a list L to the empty list and a set S to the empty set.
2. For every query $h(a, M)$ made by $\mathcal{A}_1$, it does the following:
   (a) Let $z = h(a, M)$. If there was *exactly one* earlier query by $\mathcal{A}_1$ of the form (a, M') for some $M' \neq M$ that had answer z, and $|\mathsf{S}| < 2k$, it adds the index of the query $h(a, M')$ and the current query to S.
   (b) Otherwise, it adds $h(a, M)$ to L.
3. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L in the lexicographical order of the inputs.
4. If $|\mathsf{S}| < 2k$ it outputs $\varnothing$; otherwise, it outputs L, S.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$, it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$, it does the following:
   (a) If the index of the query is in S, and there is an earlier query $h(a, M')$ for some M' by $\mathcal{A}_1$ such that its index is in S, answer this query with $h(a, M')$.
   (b) Otherwise, it removes the element in front of L and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L

Correctness of decoding: For adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}oneblk\text{-}}k$ to happen, the encoding algorithm will never return $\varnothing$ because by the definition of $\mathsf{off\text{-}oneblk\text{-}}k$ there will be at least k different salts a_i for which $\mathcal{A}_1$ queries $h(a_i, M_i) = z_i$, $h(a_i, M'_i) = z_i$; meaning the size of S will be $2k$. For such an adversary $\mathcal{A}$ it is easy to verify the decoding algorithm will always produce the correct output because for the answers of h that were not added to L, the decoding algorithm recovers them using the set of indices S. Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}oneblk\text{-}}k] .$$

The size of the output space of the encoding algorithm is upper bounded by $\binom{T_1}{2k} \cdot (2^n)^{2^{s+\ell}-k}$. The size of the input space is $(2^n)^{2^{s+\ell}}$. From the compression lemma (Proposition 1), we have that

$$\Pr\left[\mathsf{off\text{-}oneblk}\text{-}k\right] \leqslant \Pr\left[\text{ Decoding is correct }\right] \leqslant \frac{\binom{T_1}{2k}}{2^{kn}} \,.$$

We $k = \max\left(\frac{eT_1}{2^{n/2}}, n/2\right)$. If $\max\left(\frac{eT_1}{2^{n/2}}, n/2\right) = \frac{eT_1}{2^{n/2}}$, then $k \geqslant n/2$. Therefore,

$$\frac{\binom{T_1}{2k}}{2^{nk}} \leqslant \left(\frac{eT_1}{2^{n/2}(2k)}\right)^{2k} \leqslant \left(\frac{1}{2}\right)^n \,.$$

If $\max\left(\frac{eT_1}{2^{n/2}}, n/2\right) = n/2$, then $k \geqslant \frac{eT_1}{2^{n/2}}$. Therefore,

$$\frac{\binom{T_1}{2k}}{2^{nk}} \leqslant \left(\frac{eT_1}{2^{n/2}(2k)}\right)^{2k} \leqslant \left(\frac{1}{2}\right)^n \,.$$

Therefore, for $k = \max\left(\frac{eT_1}{2^{n/2}}, n/2\right)$, $\Pr\left[\mathsf{off\text{-}oneblk}\text{-}k\right] \leqslant 1/2^n$. Therefore, from (6) we have

$$\Pr\left[\mathsf{onecoll}_{\mathsf{off}}\right] \leqslant \frac{eT_1}{2^{s+n/2}} + \frac{n}{2^{s+1}} + \frac{1}{2^n} \,. \tag{7}$$

Plugging (4), (5) and (7) into (3) gives us that

$$\mathsf{Adv}^{s\text{-}\mathsf{pre\text{-}CR}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant \frac{\binom{T_2}{2}}{2^n} + \frac{T_1T_2}{2^{n+s}} + \frac{eT_1}{2^{s+n/2}} + \frac{n}{2^{s+1}} + \frac{1}{2^n} \,.$$

□

5 Offline-Online Security of Two-Block Merkle-Damgård

In previous sections, we focused on proving security guarantees against offline-online attacks on constructions where every query to the ideal primitive is salted. Here, we will see an example of a construction (Merkle Damgård) where only the first query to the underlying primitive (random oracle) is salted. Specifically, we will study the pre-image-resistance and collision-resistance for two-block Merkle-Damgård.

The main takeaway from this section is that for primitives that are not salted for every call, we can have parameter regimes where a term of the form T_1T_2 dominates the bound, meaning that in these regimes there are trade-offs between the number of offline and online queries for the advantage to be close to one. This contrasts with what we saw earlier in Section 3.2. There we demonstrated that for constructions that salt every query to the ideal primitive, an adversary must have either online time close to the online time of the best online-only attack that attains an advantage close to one, or offline time close to the best offline-only attack that attains an advantage close to one to achieve an advantage close to one.

5.1 Pre-image-resistance of two-block Merkle-Damgård

In this section we study the offline-online attacks against the pre-image-resistance of the two-block Merkle-Damgaard (MD) construction. Pre-image-resistance of two-block Merkle-Damgård is formalized in the game $\mathsf{2\text{-}PR\text{-}MD}^h_n$ in Figure 3. A salt a and a value y are sampled uniformly at random from $\{0,1\}^n$ and given as input to $\mathcal{A}$. $\mathcal{A}$ can make queries to h and wins if it outputs a message M that is one or two blocks long whose MD evaluation with a is y.

Let $H_{n,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. We are interested in proving an upper bound on $\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(T_1, T_2)$. We prove the following theorem.

Theorem 4. *Let $T_1, T_2, n, \ell \in \mathbb{N}_{>0}$. Let $H_{n,\ell,n}$ be the uniform distribution over* $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. *Then*

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(T_1, T_2) \leqslant \frac{2T_2 + 1}{2^n} + \frac{T_1T_2 + nT_1 + T_1}{2^{2n}} + \frac{4eT_1^2}{2^{3n}} \,.$$

```
Game 2-PR-MD$^h_n(\mathcal{A})$
$a \leftarrow\!\!\$ \{0,1\}^n$
$y \leftarrow\!\!\$ \{0,1\}^n$
$M \leftarrow \mathcal{A}^h(a, y)$
If $|M| \in \{\ell, 2\ell\}$ and $\mathsf{MD}^h(a, M) = y$
    Return true
Return false
```

Fig. 3: Oracle game $\mathsf{2\text{-}PR\text{-}MD}^h_n$ formalizing the pre-image-resistance of the two-block MD construction.

Offline-online trade-offs. Note that in the regime $T_1 = 2^{n(1+\epsilon)}, T_2 = 2^{n(1-\epsilon)}$ for $0 < \epsilon < 1/2$, the term $T_1T_2/2^{2n}$ dominates the bound, meaning there is an offline-online query trade-off in that regime.

Tightness. We show that this bound is tight up to factors of n. Observe that the dominant terms are of the order $T_2/2^n$, $T_1T_2/2^{2n}$, and $T_1^2/2^{3n}$. We briefly describe how we show this.

The tightness of $T_2/2^n$ follows easily – the online only attack simply makes T_2 queries with the given salt. The advantage of this attack is at least $(1 - 1/e)T_2/2^n$ for $T_2 \leqslant 2^n$, as argued in the tightness discussion for pre-image-resistance of the salted random oracle.

The following theorem proves that the term $T_1T_2/2^{2n}$ is tight.

Theorem 5. *Let $T_1, T_2, n, \ell \in \mathbb{N}_{>0}$ such that T_1 is a multiple of 2^n, $T_1T_2 \leqslant 2^{2n}$. Let $H_{n,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. Then there exists a (T_1, T_2)-adversary such that*

$$\mathsf{Adv}^{\mathsf{2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(\mathcal{A}) \geqslant \frac{T_1T_2(1 - 2/e)}{2^{2n+1}} .$$

We defer the proof of this theorem to Appendix A.2.

The following theorem proves that the term $T_1^2/2^{3n}$ is tight.

Theorem 6. *Let $T_1, n, \ell \in \mathbb{N}_{>0}$ such that T_1 is a multiple of 2^{n+1}, $T_1 \leqslant 2^{3n/2}$. Let $H_{n,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. Then there exists a $(T_1, 0)$-adversary such that*

$$\mathsf{Adv}^{\mathsf{2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(\mathcal{A}) \geqslant \frac{T_1^2(1 - 2/e)}{2^{3n+4}} .$$

We defer the proof of this theorem to Appendix A.2.

Proof (Theorem 4). Let $\mathcal{A}$ be the (T_1, T_2)-adversary that maximizes $\mathsf{Adv}^{\mathsf{2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(T_1, T_2)$. Without loss of generality $\mathcal{A}$ is deterministic and does not repeat any queries. We also assume without loss of generality that $\mathcal{A}$ makes all the queries needed to compute the MD evaluation of the messages it outputs.

We can formulate an alternate version of the game for pre-image-resistance of MD in game H in Fig. 4. Note that whenever $\mathcal{A}$ wins $\mathsf{2\text{-}PR\text{-}MD}^h_n$, it has to win H because from our assumption that $\mathcal{A}$ makes all the queries needed to compute the MD evaluation of the messages it outputs, it follows that at least one of the following happens.

– $\mathcal{A}$ makes a query $h(a, M') = y$ – meaning it has found a one-block message M' whose MD evaluation with salt a is y.
– $\mathcal{A}$ queries $h(a, M') = z$ and $h(z, M'') = y$ – meaning it has found a two-block message (M', M'') whose MD evaluation with salt a is y.

In either of these cases the flag win is set in H, meaning $\mathcal{A}$ wins the game. Therefore,

$$\mathsf{Adv}^{\mathsf{2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}}(\mathcal{A}) \leqslant \Pr[\mathsf{H}(\mathcal{A})] .$$

```
Game $\mathsf{H}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$
$h \leftarrow_\$ \mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathsf{oneblkinv}, \mathsf{twoblkinv} \leftarrow \mathsf{false}$
$\tau \leftarrow [], a, y \leftarrow \perp$
$\mathsf{st} \leftarrow \mathcal{A}_1^{\mathrm{H}}$
$a \leftarrow_\$ \{0,1\}^n$
$y \leftarrow_\$ \{0,1\}^n$
If $\exists M : ((a, M), y) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$
    $\mathsf{oneblkinv} \leftarrow \mathsf{true}$
If $\exists M, M', z$ :
    $((a, M), z), ((z, M'), y) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$
    $\mathsf{twoblkinv} \leftarrow \mathsf{true}$
$\mathcal{A}_2^{\mathrm{H}}(\mathsf{st}, a, y)$
Return win
```

```
Oracle $\mathrm{H}(a', M')$
$\tau \leftarrow \tau \cup \{((a', M'), h(a', M'))\}$
If $\exists M' : ((a, M'), y) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$
    $\mathsf{oneblkinv} \leftarrow \mathsf{true}$
If $\exists M', M'', z$ :
    $((a, M'), z), ((z, M''), y) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$
    $\mathsf{twoblkinv} \leftarrow \mathsf{true}$
Return $h(a', M')$
```

Fig. 4: H using in the analysis of pre-image-resistance of two-block MD. The events introduced in this game are marked in red.

Note that win is set to true H only if one of $\mathsf{oneblkinv}, \mathsf{twoblkinv}$ is set to true. It follows that

$$\Pr[\mathsf{H}(\mathcal{A})] = \Pr[\mathcal{A} \text{ sets } \mathsf{win}] \leqslant \Pr[\mathsf{oneblkinv}] + \Pr[\mathsf{twoblkinv}] \ .$$

We show that

$$\Pr[\mathsf{oneblkinv}] \leqslant \frac{T_2}{2^n} + \frac{T_1}{2^{2n}} \ ,$$

and

$$\Pr[\mathsf{twoblkinv}] \leqslant \frac{T_2 + 1}{2^n} + \frac{T_1 T_2 + n T_1}{2^{2n}} + \frac{4eT_1^2}{2^{3n}} \ .$$

Putting it all together would give us the theorem.

We next upper bound $\Pr[\mathsf{oneblkinv}], \Pr[\mathsf{twoblkinv}]$.

Towards upper bounding $\Pr[\mathsf{oneblkinv}]$, we define the two following events:

1. $\mathsf{oneblkinv}_{\mathsf{off}}$: $\mathcal{A}_1$ makes a query with input salt a and output y
2. $\mathsf{oneblkinv}_{\mathsf{on}}$: $\mathcal{A}_2$ makes a query which has output y

It is easy to see that if $\mathsf{oneblkinv}$ happens, then at least one of $\mathsf{oneblkinv}_{\mathsf{off}}$, $\mathsf{oneblkinv}_{\mathsf{on}}$ has to happen. Therefore

$$\Pr[\mathsf{oneblkinv}] \leqslant \Pr[\mathsf{oneblkinv}_{\mathsf{off}}] + \Pr[\mathsf{oneblkinv}_{\mathsf{on}}] \ .$$

We first upper bound $\Pr[\mathsf{oneblkinv}_{\mathsf{on}}]$. Each query by $\mathcal{A}_2$ has answer y with probability $1/2^n$. Using a union bound over all queries of $\mathcal{A}_2$, we have that

$$\Pr[\mathsf{oneblkinv}_{\mathsf{on}}] \leqslant T_2/2^n \ .$$

We next upper bound $\Pr[\mathsf{oneblkinv}_{\mathsf{off}}]$. Consider the set of (s, y) pairs such that there is a query by $\mathcal{A}_1$ with input salt s and answer y. There are at most T_1 such pairs. Note that, $\mathsf{oneblkinv}_{\mathsf{off}}$ happens only if (a, y) which is sampled uniformly at random, is among those at most T_1 pairs. Hence,

$$\Pr[\mathsf{oneblkinv}_{\mathsf{off}}] \leqslant T_1/2^{2n} \ .$$

Putting this together, we have the required bound on $\Pr[\mathsf{oneblkinv}]$.

We next upper bound $\Pr[\mathsf{twoblkinv}]$. We define the three following events.

1. $\mathsf{twoblkinv}_{\mathsf{off}}$: $\mathcal{A}_1$ makes queries $h(a, M) = z$ and $h(z, M') = y$ for some M, M', z
2. $\mathsf{twoblkinv}_{\mathsf{offon}}$: $\mathcal{A}_1$ makes a query $h(z, M) = y$ and $\mathcal{A}_2$ makes a query $h(a, M') = z$ for some M, M', z
3. $\mathsf{twoblkinv}_{\mathsf{on}}$: $\mathcal{A}_2$ makes a query with answer y

It is easy to see that if $\mathsf{twoblkinv}$ happens then at least one of $\mathsf{twoblkinv}_{\mathsf{off}}$, $\mathsf{twoblkinv}_{\mathsf{offon}}$, $\mathsf{twoblkinv}_{\mathsf{on}}$ has to happen. Therefore,

$$\Pr[\mathsf{twoblkinv}] \leqslant \Pr[\mathsf{twoblkinv}_{\mathsf{off}}] + \Pr[\mathsf{twoblkinv}_{\mathsf{offon}}] + \Pr[\mathsf{twoblkinv}_{\mathsf{on}}] \ . \tag{8}$$

We upper bound these probabilities one by one starting with $\Pr[\mathsf{twoblkinv}_{\mathsf{on}}]$. Observe that every query by $\mathcal{A}_2$ has probability $1/2^n$ of having answer y. Using a union bound over all queries of $\mathcal{A}_2$, it follows that

$$\Pr[\mathsf{twoblkinv}_{\mathsf{on}}] \leqslant \frac{T_2}{2^n} \ . \tag{9}$$

We next upper bound $\Pr[\mathsf{twoblkinv}_{\mathsf{offon}}]$. Observe that this event happens only if $\mathcal{A}_2$ makes a query that has answer z such that $\mathcal{A}_1$ made a query with salt z that had answer y. Therefore, using total probability

$$\begin{aligned}\Pr[\mathsf{twoblkinv}_{\mathsf{offon}}] &\leqslant \sum_{k=1}^{T_1} \Pr[\mathcal{A}_1 \text{ made } k \text{ queries with answer } y] \frac{kT_2}{2^n} \\ &= \frac{T_2}{2^n} \sum_{k=1}^{T_1} \mathsf{E}[\text{Number of queries of } \mathcal{A}_1 \text{ with answer } y] \\ &= \frac{T_2 T_1}{2^{2n}} \ . \end{aligned} \tag{10}$$

The last equality follows since each query of $\mathcal{A}_1$ has answer y with probability $1/2^n$.

Finally, we upper bound $\Pr[\mathsf{twoblkinv}_{\mathsf{off}}]$. For this we initially take a short detour and define the event $(m+1)$-col as the event that $\mathcal{A}_1$ has made $m+1$ distinct random oracle queries, all of which have the same answer. We claim that

$$\Pr\left[\mathsf{twoblkinv}_{\mathsf{off}} \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] \leqslant \frac{mT_1}{2^{2n}} \ .$$

This is because if $(m+1)$-col does not happen, there can be at most mT_1 pairs of queries such that the answer of the one query of the pair is the input of the other query (as otherwise there would be a $m+1$-multi-collision since there are T_1 queries made by $\mathcal{A}_1$). Now, $\mathsf{twoblkinv}_{\mathsf{off}}$ happens only if (a, y) that is sampled uniformly at random is such that one of these at most mT_1 pairs have a as the salt for one query and y as the answer of the other query. This happens with probability at most $\frac{mT_1}{2^{2n}}$.

Finally, we upper bound $\Pr[(m+1)\text{-}\mathsf{col}]$. For any subset of $m+1$ queries made by $\mathcal{A}_1$, the probability that they have the same answer is $1/2^{nm}$. Using a union bound over all possible $m+1$ sized subsets of the queries of $\mathcal{A}_1$, we have that

$$\Pr[(m+1)\text{-}\mathsf{col}] \leqslant \frac{\binom{T_1}{m+1}}{2^{nm}} \ .$$

We let $m = \max\left(n, \frac{4eT_1}{2^n}\right)$. If $n \leqslant \frac{4eT_1}{2^n}$, we have that $m = \frac{4eT_1}{2^n} \geqslant n$. Therefore,

$$\Pr[(m+1)\text{-}\mathsf{col}] \leqslant \frac{\binom{T_1}{m+1}}{2^{mn}} \leqslant \left(\frac{eT_1}{(m+1)2^n}\right)^{m+1} 2^n \leqslant \left(\frac{1}{4}\right)^m 2^n \leqslant \left(\frac{1}{2}\right)^n \ .$$

Otherwise, if $n > \frac{4eT_1}{2^n}$, we have that $m = n > \frac{4eT_1}{2^n}$. Therefore,

$$\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant \frac{\binom{T_1}{m+1}}{2^{nm}} \leqslant \left(\frac{eT_1}{(m+1)2^n}\right)^{m+1} 2^n \leqslant \left(\frac{1}{4}\right)^{m+1} 2^n \leqslant \left(\frac{1}{2}\right)^n .$$

Therefore, we have that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$, $\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n$. Hence

$$\Pr\left[\mathsf{twoblkinv}_{\mathsf{off}}\right] \leqslant \frac{nT_1}{2^{2n}} + \frac{4eT_1^2}{2^{3n}} + \frac{1}{2^n} . \tag{11}$$

Plugging (9) to (11) into (8) gives us the required bound for $\Pr\left[\mathsf{twoblkinv}\right]$ and concludes the proof. □

5.2 Collision-resistance of two-block Merkle-Damgård

In this section, we study the collision-resistance of two-block Merkle-Damgård (MD) against offline-online adversaries. Collision-resistance of two-block MD is formalized by the oracle game $\mathsf{2\text{-}CR\text{-}MD}_n^h$ in Fig. 5. In this game a salt a is picked at random from $\{0,1\}^n$ that is given to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ has oracle access to h, and wins if it can output two messages M, M' that are distinct; both at most 2 blocks long, and satisfy $\mathsf{MD}^h(a, M) = \mathsf{MD}^h(a, M')$.

The game $\mathsf{pre\text{-}2\text{-}CR\text{-}MD}_n^h$ captures the collision-resistance of 2-block MD against offline-online adversaries. We prove the following upper bound on $\mathsf{Adv}_{H_{n,\ell,n}}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD},n}$.

Theorem 7. *Let $T_1, T_2, s, \ell, n \in \mathbb{N}_{>0}$. Let $H_{n,\ell,n}$ be the uniform distribution on $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$.*

$$\begin{aligned}\mathsf{Adv}_{H_{s,\ell,n}}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}(\mathcal{A}) \leqslant{}& \frac{2T_2^2 + nT_2/2 + 3n^2/2 + 99n/2 + 33}{2^n} \\ &+ \left(\frac{T_1T_2}{2^{3n/2}}\right)(n^2 + 5n + 83) \\ &+ \left(\frac{T_1}{2^{5n/4}}\right)(53n + 14n^{1/2} + 56n^{1/3} + 342) + 468\left(\frac{T_1^2}{2^{7n/3}}\right) .\end{aligned}$$

Offline-online trade-offs. Note that, in the regime of parameters $T_1 = 2^{n(1+\epsilon)}, T_2 = 2^{n(1/2-\epsilon)}$ for $0 < \epsilon < 1/6$, the term $T_1T_2/2^{3n/2}$ dominates the bound, i.e., there is a trade-off between the number of offline and online queries in that regime.

Tightness of the bound. We show that the first three terms in the above bound are tight by giving matching attacks. We could not find an attack matching the last term in the bound and leave improving it or showing it tight to be future research.

We briefly describe how we show the other terms to be tight. The first term is dominated by $T_2^2/2^n$ – we can show that this is tight up to constant factors using the birthday attack, which achieves advantage of the order $T_2^2/2^n$.

In the second term, ignoring constants and powers of n, the dominant factor is $\frac{T_1T_2}{2^{3n/2}}$. We prove this theorem to show that it is tight.

Theorem 8. *Let $T_1, T_2, n, \ell \in \mathbb{N}_{>0}$ such that n is a multiple of 2, T_1 is a multiple of $2^{n/2+1}$, and $T_1T_2 \leqslant 2^{3n/2}$. Let $H_{n,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. There exists a (T_1, T_2) adversary $\mathcal{A}$ such that*

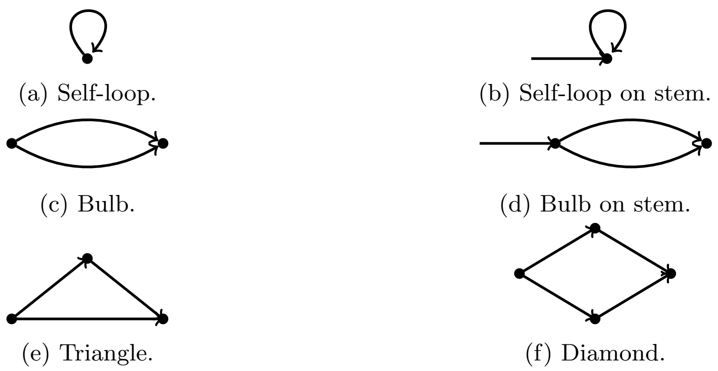
$$\mathsf{Adv}_{H_{n,\ell,n}}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}(\mathcal{A}) \geqslant \frac{(1-2/e)T_1T_2}{2^{3n/2+3}} .$$

The proof of this theorem is in Appendix A.1.

In the third term, ignoring constants and powers of n, the dominant factor is $\frac{T_1}{2^{5n/4}}$. We give an attack that achieves advantage of the order $\frac{T_1^2}{2^{5n/2}}$. While $\frac{T_1^2}{2^{5n/2}} \leqslant \frac{T_1}{2^{5n/4}}$ for $T_1 \leqslant 2^{5n/4}$, observe that both of them become one at $T_1 = 2^{5n/4}$. Formally, we prove the following theorem.

```
Game $\mathsf{2\text{-}CR\text{-}MD}_n^h(\mathcal{A})$
$a \leftarrow\!\!\$\ \{0,1\}^n$
$(M, M') \leftarrow \mathcal{A}^h(a)$
If $|M|, |M'| \in \{\ell, 2\ell\}$ and $M \neq M'$ and $\mathsf{MD}^h(a, M) = \mathsf{MD}^h(a, M')$
    Return $\mathsf{true}$
Return $\mathsf{false}$
```

Fig. 5: Oracle game $\mathsf{2\text{-}CR\text{-}MD}_n^h$ formalizing collision-resistance of two-block MD.



(a) Self-loop.

(b) Self-loop on stem.

(c) Bulb.

(d) Bulb on stem.

(e) Triangle.

(f) Diamond.

Fig. 6: The structure of the six different types of two-block MD collisions in the query graph. The nodes in the query graph are labelled with values in $\{0,1\}^n$, and there is an edge (a, a') labelled with M if the adversary made a query $h(a, M) = a'$. We omit the node and edge labels for simplicity.

Theorem 9. *Let $T_1, T_2, n, \ell \in \mathbb{N}_{>0}$ such that n is a multiple of 2, and T_1 is a multiple of $2^{n/2+1}$. Let $H_{n,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^\ell, \{0,1\}^n)$. There exists a (T_1, T_2) adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}_{H_{n,\ell,n}}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}(\mathcal{A}) \geqslant \frac{(1-2/e)T_1^2}{2^{5n/2+6}} .$$

The proof of this theorem is in Appendix A.1.

We now proceed to prove Theorem 7.

Proof. The proof of this theorem fixes the (T_1, T_2)-offline-online adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that maximizes $\mathsf{Adv}_{H_{n,\ell,n}}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}(T_1, T_2)$. We can treat $\mathcal{A}$ as deterministic by fixing its randomness that maximizes its advantage. Without loss of generality we can assume that $\mathcal{A}$ does not repeat any query across the offline and online phases because we have no restrictions on the memory of the adversary.

We rewrite the collision-resistance game for two-block MD in game H in Figure 7. Note that whenever $\mathcal{A}$ wins $\mathsf{G}_{n,\ell}^{\mathsf{2\text{-}PR\text{-}MD}}$, from our assumption that $\mathcal{A}$ makes all the queries needed to compute the MD evaluation of the messages it outputs, at least one of the following happens.

- $\mathcal{A}$ makes a query $h(a, M') = a$, meaning it has found a one-block message M' whose MD evaluation with salt a is a. This is sufficient for a two-block collision because for any $M'' \in \{0,1\}^\ell$, (M', M'') and M'' have the same MD evaluation with salt a.
- $\mathcal{A}$ makes queries $h(a, M') = z$ and $h(z, M'') = z$; this is a two-block collision because (M', M'') and M' have the same MD evaluation with salt a.
- $\mathcal{A}$ makes queries $h(a, M') = z$ and $h(a, M'') = z$ for $M' \neq M''$; this is a two-block collision because M', and M'' have the same MD evaluation with salt a.
- $\mathcal{A}$ makes queries $h(a, M') = z$, $h(y, M'') = z$ and $h(y, M''') = z$ for $M'' \neq M'''$; this is a two-block collision because (M', M'') and (M', M'') have the same MD evaluation with salt a.

```
Game $\mathsf{H}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$
$h \leftarrow\!\!\$\ \mathsf{Fcs}(\{0,1\}^n \times \{0,1\}^m, \{0,1\}^n)$
$\mathsf{win} \leftarrow \mathsf{false}$, $\tau \leftarrow []$, $a \leftarrow \bot$
$\mathsf{sl}, \mathsf{sos}, \mathsf{bulb}, \mathsf{bos}, \mathsf{tri}, \mathsf{dia} \leftarrow \mathsf{false}$
$\mathsf{st} \leftarrow \mathcal{A}_1^{\mathrm{H}}$
$a \leftarrow\!\!\$\ \{0,1\}^n$
If $\exists M' : ((a, M'), a) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{sl} \leftarrow \mathsf{true}$
If $\exists M', M'', z$ :
    $((a, M'), z), ((z, M''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{sos} \leftarrow \mathsf{true}$
If $\exists M' \neq M'', z$ :
    $((a, M'), z), ((a, M''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{bulb} \leftarrow \mathsf{true}$
If $\exists M', M'' \neq M''', y, z$ :
    $((a, M'), y), ((y, M''), z) \in \tau$,
    $((y, M'''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{bos} \leftarrow \mathsf{true}$
If $\exists M', M'', M''', y, z$ :
    $((a, M'), y), ((y, M''), z) \in \tau$,
    $((a, M'''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{tri} \leftarrow \mathsf{true}$
If $\exists M' \neq M''', M'' \neq M'''', x, y, z$ :
    $((a, M'), x), ((a, M''), y) \in \tau$,
    $((x, M'''), z), ((y, M''''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{dia} \leftarrow \mathsf{true}$
$\mathcal{A}_2^{\mathrm{H}}(\mathsf{st}, a)$
Return $\mathsf{win}$

Oracle $\mathrm{H}(a, M)$
$\tau \leftarrow \tau \cup \{((a, M), h(a, M))\}$
If $\exists M' : ((a, M'), a) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{sl} \leftarrow \mathsf{true}$
If $\exists M', M'', z$ :
    $((a, M'), z), ((z, M''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{sos} \leftarrow \mathsf{true}$
If $\exists M' \neq M'', z$ :
    $((a, M'), z), ((a, M''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{bulb} \leftarrow \mathsf{true}$
If $\exists M', M'' \neq M''', y, z$ :
    $((a, M'), y), ((y, M''), z) \in \tau$,
    $((y, M'''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{bos} \leftarrow \mathsf{true}$
If $\exists M', M'', M''', y, z$ :
    $((a, M'), y), ((y, M''), z) \in \tau$,
    $((a, M'''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{tri} \leftarrow \mathsf{true}$
If $\exists M' \neq M''', M'' \neq M'''', x, y, z$ :
    $((a, M'), x), ((a, M''), y) \in \tau$,
    $((x, M'''), z), ((y, M''''), z) \in \tau$:
    $\mathsf{win} \leftarrow \mathsf{true}$, $\mathsf{dia} \leftarrow \mathsf{true}$
$\mathcal{A}_2^{\mathrm{H}}(\mathsf{st}, a)$
Return $h(a, M)$
```

Fig. 7: H using in the analysis of collision-resistance of two-block MD against offline-online adversaries. The events introduced in this game are marked in red.

- $\mathcal{A}$ makes queries $h(a, M') = y$, $h(y, M'') = z$ and $h(a, M''') = z$; this is a two-block collision because (M', M'') and M''' have the same MD evaluation with salt a.
- $\mathcal{A}$ makes queries $h(a, M') = y$, $h(y, M'') = z$, $h(a, M''') = y'$, and $h(y', M''') = z$ for $M' \neq M'''$ and $M'' \neq M''''$; this is a two-block collision because (M', M'') and (M''', M'''') have the same MD evaluation with salt a.

If any of these occur, win is set in H, meaning $\mathcal{A}$ wins the game. Therefore,

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{n,\ell,n}}(\mathcal{A}) \leqslant \Pr[\mathsf{H}(\mathcal{A})] \ .$$

The game H defines events $\mathsf{sl}, \mathsf{sos}, \mathsf{bulb}, \mathsf{bos}, \mathsf{tri}, \mathsf{dia}$. We name the events this way because of the following alternative view of MD collisions via the query graph of $\mathcal{A}$: the nodes of the query graph are labelled with strings from $\{0,1\}^n$, and whenever $\mathcal{A}$ makes a query $h(a, M) = a'$, an edge (a, a') labelled M is added to the graph. Finding a two block collision can be viewed as finding one of the following structures in the query graph: self-loop, self-loop on stem, bulb, bulb-on-stem, triangle, and diamond; Fig. 6 shows these structures.

It follows from inspection that in game H, win is set only if one of these event among $\{\mathsf{sl}, \mathsf{sos}, \mathsf{bulb}, \mathsf{bos}, \mathsf{tri}, \mathsf{dia}\}$ happen. Therefore, using the union bound we have that

$$\begin{aligned} \Pr[\mathsf{H}(\mathcal{A})] &= \Pr[\mathcal{A} \text{ sets } \mathsf{win}] \\ &\leqslant \sum_{\mathsf{event} \in \{\mathsf{sl}, \mathsf{sos}, \mathsf{bulb}, \mathsf{bos}, \mathsf{tri}, \mathsf{dia}\}} \Pr[\mathsf{event}] \end{aligned} \qquad (12)$$

Our proof is divided into these following lemmas each of which upper bound the probability of these events.

Lemma 4.

$$\Pr[\mathsf{sl}] \leqslant \frac{1}{2^n} + \frac{n}{2^n} + \frac{2eT_1}{2^{2n}} + \frac{T_2}{2^n} .$$

Lemma 5.

$$\Pr[\mathsf{sos}] \leqslant \frac{T_2 + 3 + nT_2 + n^2}{2^n} + \frac{2eT_1T_2 + 6enT_1}{2^{2n}} + \frac{8e^2T_1^2}{2^{3n}} .$$

Lemma 6.

$$\Pr[\mathsf{bulb}] \leqslant \frac{\binom{T_2}{2}}{2^n} + \frac{T_2T_1}{2^{2n}} + \frac{eT_1}{2^{3n/2}} + \frac{n}{2^{n+1}} + \frac{1}{2^n} .$$

Lemma 7.

$$\begin{aligned}\Pr[\mathsf{bos}] \leqslant & \frac{\binom{T_2}{2} + nT_2/2 + n^2/2 + 4}{2^n} + \frac{eT_1T_2 + enT_1}{2^{3n/2}} + \frac{nT_1T_2 + T_1T_2^2 + 2enT_1}{2^{2n}} \\ & + \frac{4e^2T_1^2}{2^{5n/2}} + \frac{4eT_1^2T_2}{2^{3n}} .\end{aligned}$$

Lemma 8.

$$\begin{aligned}\Pr[\mathsf{tri}] \leqslant & \frac{\binom{T_2}{2} + 18n + 8}{2^n} + \frac{3(24e)^{1/2}T_1 + 3(8en)^{1/2}T_1}{2^{3n/2}} + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} \\ & + \frac{2nT_1T_2 + T_1T_2 + T_2^2T_1 + 12e(2)^{1/2}T_1^{3/2}}{2^{2n}} + \frac{8eT_1^2T_2 + 2T_1T_2}{2^{3n}} .\end{aligned}$$

Lemma 9.

$$\begin{aligned}\Pr[\mathsf{dia}] \leqslant & \frac{\binom{T_2}{2} + 30n + 16}{2^n} + \frac{4eT_1^2T_2 + 2T_1T_2 + 4eT_1^2T_2^2}{2^{3n}} + \\ & \frac{nT_1T_2 + T_2^2T_1 + n^2T_1T_2 + nT_1T_2^2}{2^{2n}} + \frac{4eT_1^3T_2}{2^{4n}} \\ & + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{(8e)^{1/2}nT_1 + 10(2e)^{1/2}nT_1}{2^{3n/2}} \\ & + \frac{240e^{2/3}T_1^2}{2^{7n/3}} + \frac{8(2)^{1/2}e^{3/2}T_1^2 + 40(2e)^{1/2}T_1^2}{2^{5n/2}} .\end{aligned}$$

Plugging in Lemmas 4 to 9 into (12) and grouping the terms, we get that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant & \frac{33 + 99n/2 + 2T_2 + 3nT_2/2 + 3n^2/2 + 4\binom{T_2}{2}}{2^n} \\ & + \frac{(2e + 8ne)T_1 + (3+n)T_1T_2^2 + 12e(2)^{1/2}T_1^{3/2}}{2^{2n}} \\ & + \frac{(2 + 2e + 4n + n^2)T_1T_2}{2^{2n}} + \frac{eT_1T_2}{2^{3n/2}} \\ & + \frac{(e + en + 3(24e)^{1/2} + 3(8en)^{1/2} + (8e)^{1/2}n + 10(2e)^{1/2}n)T_1}{2^{3n/2}} \\ & + \frac{8e^2T_1^2 + 16eT_1^2T_2 + 4T_1T_2 + 4eT_1^2T_2^2}{2^{3n}} \\ & + \frac{(4e^2 + 8(2)^{1/2}e^{3/2} + 40(2e)^{1/2})T_1^2}{2^{5n/2}} \\ & + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} + \frac{4eT_1^3T_2}{2^{4n}} + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{240e^{2/3}T_1^2}{2^{7n/3}} .\end{aligned} \tag{13}$$

Note that this bound trivially holds when $T_2 > 2^{n/2}$ because in that case $T_2^2/2^n > 1$ and the left hand side is a probability which is at most 1. Similarly, it trivially holds when $T_2T_1 > 2^{3n/2}$ or $T_1 > 2^{5n/4}$. Therefore, we can assume $T_2 \leqslant 2^{n/2}$, $T_1 \leqslant 2^{5n/4}$, and $T_1T_2 \leqslant 2^{3n/2}$. For $T_1 \leqslant 2^{5n/4}$, we have the following inequalities.

$$\frac{T_1^{3/2}}{2^{2n}}, \frac{T_1^2}{2^{3n}}, \frac{T_1^2}{2^{5n/2}}, \frac{T_1^{4/3}}{2^{5n/3}}, \frac{T_1^2T_2}{2^{3n}}, \frac{T_1^3T_2}{2^{4n}} \leqslant \frac{T_1}{2^{5n/4}}$$

For $T_2 \leqslant 2^{n/2}$, we have the following inequalities.

$$\frac{T_1T_2^2}{2^{2n}}, \frac{T_1T_2^2}{2^{2n}} \leqslant \frac{T_1T_2}{2^{3n/2}}$$

For $T_1T_2 \leqslant 2^{3n/2}$ we have that $T_1^2T_2^2/2^{3n} \leqslant T_1T_2/2^{3n/2}$. Further for any $T_1, T_2 \geqslant 0$, we have the following inequalities.

$$\begin{aligned}
&\frac{T_1}{2^{2n}}, \frac{T_1}{2^{3n/2}}, \frac{T_1}{2^{4n/3}} \leqslant \frac{T_1}{2^{5n/4}} \\
&\frac{T_1T_2}{2^{2n}}, \frac{T_1T_2}{2^{3n}} \leqslant \frac{T_1T_2}{2^{3n/2}}
\end{aligned}$$

Using all of these inequalities in (13), we have that

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant\ & \frac{33 + 99n/2 + 2T_2 + 3nT_2/2 + 3n^2/2 + 4\binom{T_2}{2}}{2^n} \\
&+ \frac{((2e + 8ne) + 12e(2)^{1/2})T_1}{2^{5n/4}} + \frac{(3+n)T_1T_2}{2^{3n/2}} \\
&+ \frac{(2 + 2e + 4n + n^2)T_1T_2}{2^{3n/2}} + \frac{eT_1T_2}{2^{3n/2}} \\
&+ \frac{(e + en + 3(24e)^{1/2} + 3(8en)^{1/2} + (8e)^{1/2}n + 10(2e)^{1/2}n)T_1}{2^{5n/4}} \\
&+ \frac{8e^2T_1}{2^{5n/4}} + \frac{(16e + 4 + 4e)T_1T_2}{2^{3n/2}} \\
&+ \frac{(4e^2 + 8(2)^{1/2}e^{3/2} + 40(2e)^{1/2})T_1}{2^{5n/4}} \\
&+ \frac{9(2)^{1/3}eT_1}{2^{5n/4}} + \frac{4eT_1T_2}{2^{3n/2}} + \frac{40(en)^{1/3}T_1}{2^{5n/4}} + \frac{240e^{2/3}T_1^2}{2^{7n/3}} \ .
\end{aligned}$$

Consolidating terms we get that

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant\ & \frac{2T_2^2 + nT_2/2 + 3n^2/2 + 99n/2 + 33}{2^n} \\
&+ \left(\frac{T_1T_2}{2^{3n/2}}\right)(n^2 + 5n + 27e + 9) \\
&+ \left(\frac{T_1}{2^{5n/4}}\right)(n(9e + (8e)^{1/2} + 10(2e)^{1/2}) + n^{1/2}(3(8e)^{1/2}) \\
&+ n^{1/3}(40e^{1/3}) + (3e + 12e(2)^{1/2} + 12e^2 + 8(2)^{1/2}e^{3/2} + 40(2e)^{1/2} \\
&+ 9(2)^{1/3}e + 3(24e)^{1/2}) + \frac{240e^{2/3}T_1^2}{2^{7n/3}} \ .
\end{aligned}$$

We observe that the following inequalities hold

$$\begin{aligned}
&27e + 9 \leqslant 83\\
&9e + (8e)^{1/2} + 10(2e)^{1/2} \leqslant 53\\
&3(8e)^{1/2} \leqslant 14\\
&40e^{1/3} \leqslant 56\\
&3e + 12e(2)^{1/2} + 12e^2 + 8(2)^{1/2}e^{3/2} + 40(2e)^{1/2} + 9(2)^{1/3}e + 3(24e)^{1/2} \leqslant 342\\
&240e^{2/3} \leqslant 468
\end{aligned}$$

Using this we have that

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant\ & \frac{2T_2^2 + nT_2/2 + 3n^2/2 + 99n/2 + 33}{2^n}\\
&+ \left(\frac{T_1T_2}{2^{3n/2}}\right)(n^2 + 5n + 83)\\
&+ \left(\frac{T_1}{2^{5n/4}}\right)(53n + 14n^{1/2} + 56n^{1/3} + 342) + 468\left(\frac{T_1^2}{2^{7n/3}}\right) .
\end{aligned}$$

We prove Lemmas 4 to 9 in Sections 5.3 to 5.8, respectively. □

5.3 Proof of Lemma 4

Proof (Lemma 4). We define the two following events.

1. $\mathsf{sl}_{\mathsf{off}}$: $\mathcal{A}_1$ made a query $h(a, M) = a$
2. $\mathsf{sl}_{\mathsf{on}}$: $\mathcal{A}_2$ made a query $h(a, M) = a$

Notice that sl happens only if at least one of $\mathsf{sl}_{\mathsf{off}}$ or $\mathsf{sl}_{\mathsf{on}}$ happens. Therefore, we have that

$$\Pr[\mathsf{sl}] \leqslant \Pr[\mathsf{sl}_{\mathsf{off}}] + \Pr[\mathsf{sl}_{\mathsf{on}}] \ . \tag{14}$$

We first prove an upper bound on $\Pr[\mathsf{sl}_{\mathsf{on}}]$. Note that, for every query $h(a, M)$ made by $\mathcal{A}_2$, the probability that its answer is a is $1/2^n$. Therefore, using a union bound over all the queries of $\mathcal{A}_2$, we have that

$$\Pr[\mathsf{sl}_{\mathsf{on}}] \leqslant T_2/2^n \ .$$

To prove an upper bound on $\Pr[\mathsf{sl}_{\mathsf{off}}]$, we define the following event $\mathsf{off\text{-}sl\text{-}}k$: $\mathcal{A}_1$ makes at least k different queries such that the input salt of the query is the answer of the query. Using total probability, we have that for any k

$$\Pr[\mathsf{sl}_{\mathsf{off}}] \leqslant \Pr\left[\mathsf{sl}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k\right] + \Pr[\mathsf{off\text{-}sl\text{-}}k] + \frac{k}{2^n} \ . \tag{15}$$

Note that, if the adversary $\mathcal{A}_1$ makes at most k different queries, such that the input salt of the query is the answer, $\mathsf{sl}_{\mathsf{off}}$ happens only if the salt a that is sampled uniformly at random is same as the salt for one of those at most k queries. Therefore, $\Pr\left[\mathsf{sl}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k\right] \leqslant k/2^n$.

We upper bound $\Pr[\mathsf{off\text{-}sl\text{-}}k]$ as follows. Let B_j be the indicator random variable that indicates whether the j-th query of $\mathcal{A}_1$ is such that its answer is same as its input salt. Since $\mathcal{A}_1$ does not repeat queries, all the B_j's are independent, and $\Pr[B_j] = 1/2^n$. From the definition of B_j's, it follows that

$$\Pr[\mathsf{off\text{-}sl\text{-}}k] = \Pr\left[\sum_{j=1}^{T_1} B_j \geqslant k\right] .$$

We rewrite the term on the right as

$$\Pr\left[\sum_{j=1}^{T_1} B_j \geqslant k\right] = \Pr\left[\exists S \subseteq [T_1], |S| = k : \forall j \in S, B_j = 1\right] .$$

Using a union bound over all subsets of T_1 of size k, we have

$$\Pr\left[\exists S \subseteq [T_1], |S| = k : \forall j \in S, B_j = 1\right] \leqslant \sum_{S \subseteq [T_1], |S| = k} \Pr\left[\forall j \in S, B_j = 1\right] .$$

Since there are $\binom{T_1}{k}$ subsets of $[T_1]$ of size k, and all the B_j's are independent and $\Pr[B_j = 1] = 1/2^n$, we have that

$$\Pr\left[\exists S \subseteq [T_1], |S| = k : \forall j \in S, B_j = 1\right] \leqslant \frac{\binom{T_1}{k}}{2^{nk}} .$$

Therefore

$$\Pr\left[\textsf{off-sl-}k\right] \leqslant \frac{\binom{T_1}{k}}{2^{nk}} \leqslant \left(\frac{eT_1}{k2^n}\right)^k .$$

Plugging this in (15),we have that for any k,

$$\Pr\left[\textsf{sl}_{\textsf{off}}\right] \leqslant \left(\frac{eT_1}{k2^n}\right)^k + \frac{k}{2^n} .$$

We let $k = \max\left(n, \frac{2eT_1}{2^n}\right)$. If $n \leqslant \frac{2eT_1}{2^n}$, we have that $k = \frac{2eT_1}{2^n} \geqslant n$. Therefore,

$$\frac{\binom{T_1}{k}}{2^{nk}} \leqslant \left(\frac{eT_1}{k2^n}\right)^k \leqslant \left(\frac{1}{2}\right)^k \leqslant \frac{1}{2^n} .$$

Otherwise if $n > \frac{2eT_1}{2^n}$, we have that $k = n > \frac{2eT_1}{2^n}$. Therefore,

$$\frac{\binom{T_1}{k}}{2^{nk}} \leqslant \left(\frac{eT_1}{k2^n}\right)^k \leqslant \left(\frac{1}{2}\right)^k = \frac{1}{2^n} .$$

Hence,

$$\Pr\left[\textsf{sl}_{\textsf{off}}\right] \leqslant \frac{1}{2^n} + \frac{n}{2^n} + \frac{2eT_1}{2^{2n}} .$$

Plugging this back into (14) gives us

$$\Pr\left[\textsf{sl}\right] \leqslant \frac{1}{2^n} + \frac{n}{2^n} + \frac{2eT_1}{2^{2n}} + \frac{T_2}{2^n} .$$

□

5.4 Proof of Lemma 5

Proof (Lemma 5). We first define the three following events.

1. $\textsf{sos}_{\textsf{off}}$: $\mathcal{A}_1$ made queries $h(a, M') = z$ and $h(z, M'') = z$
2. $\textsf{sos}_{\textsf{offon}}$: $\mathcal{A}_1$ made a query $h(y, M'') = y$ and $\mathcal{A}_2$ made a query $h(a, M') = y$
3. $\textsf{sos}_{\textsf{on}}$: $\mathcal{A}_2$ made a query $h(z, M') = z$, i.e., a query whose answer is the same as its input salt

From inspection one can verify that sos happens only if at least one of $\mathsf{sos}_{\mathsf{off}}$, $\mathsf{sos}_{\mathsf{on}}$, $\mathsf{sos}_{\mathsf{offon}}$ happen. It follows that

$$\Pr\left[\mathsf{sos}\right] \leqslant \Pr\left[\mathsf{sos}_{\mathsf{off}}\right] + \Pr\left[\mathsf{sos}_{\mathsf{on}}\right] + \Pr\left[\mathsf{sos}_{\mathsf{offon}}\right] \ . \tag{16}$$

We first upper bound $\Pr\left[\mathsf{sos}_{\mathsf{on}}\right]$. Note that for every query $h(a, M)$ made by $\mathcal{A}_2$, the probability that its answer is a is $1/2^n$. Therefore, using a union bound over all the queries of $\mathcal{A}_2$, we have that

$$\Pr\left[\mathsf{sos}_{\mathsf{on}}\right] \leqslant T_2/2^n \ . \tag{17}$$

We next upper bound $\Pr\left[\mathsf{sos}_{\mathsf{offon}}\right]$. Recall that the event $\mathsf{off\text{-}sl\text{-}}k$ defined in the proof of Lemma 4: $\mathcal{A}_1$ makes at least k different queries such that the input salt of the query is the answer. We have that

$$\Pr\left[\mathsf{sos}_{\mathsf{offon}}\right] \leqslant \Pr\left[\mathsf{sos}_{\mathsf{offon}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k\right] + \Pr\left[\mathsf{off\text{-}sl\text{-}}k\right] \ .$$

In this case, $\mathsf{sos}_{\mathsf{offon}}$ happens only if $\mathcal{A}_2$ makes a query whose answer is the input salt of one of at most k such queries. Therefore, we have that for any k,

$$\Pr\left[\mathsf{sos}_{\mathsf{offon}}\right] \leqslant \Pr\left[\mathsf{off\text{-}sl\text{-}}k\right] + \frac{kT_2}{2^n} \ .$$

As seen in the proof of Lemma 4, setting $k = \max\left(n, \frac{2eT_1}{2^n}\right)$ makes $\Pr\left[\mathsf{off\text{-}sl\text{-}}k\right] \leqslant 1/2^n$. Therefore, by setting this value of k, we have that

$$\Pr\left[\mathsf{sos}_{\mathsf{offon}}\right] \leqslant \frac{nT_2}{2^n} + \frac{2eT_1T_2}{2^{2n}} + \frac{1}{2^n} \ . \tag{18}$$

We finally upper bound $\Pr\left[\mathsf{sos}_{\mathsf{off}}\right]$. For this we recall $(m+1)\text{-}\mathsf{col}$ as the event that we defined in the proof of Theorem 4. We say that $(m+1)\text{-}\mathsf{col}$ happens if the $\mathcal{A}_1$ has made $m+1$ distinct random oracle queries that all have the same answer. Using total probability, we have that for any k, m

$$\begin{aligned}\Pr\left[\mathsf{sos}_{\mathsf{off}}\right] &\leqslant \Pr\left[\mathsf{sos}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}\right] + \Pr\left[\mathsf{off\text{-}sl\text{-}}k \vee (m+1)\text{-}\mathsf{col}\right] \\ &\leqslant \Pr\left[\mathsf{sos}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}\right] + \Pr\left[\mathsf{off\text{-}sl\text{-}}k\right] + \Pr\left[(m+1)\text{-}\mathsf{col}\right] \ . \end{aligned} \tag{19}$$

We claim that

$$\Pr\left[\mathsf{sos}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}sl\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}\right] \leqslant \frac{mk}{2^n} \ .$$

This is because if $\mathsf{off\text{-}sl\text{-}}k$ and $(m+1)\text{-}\mathsf{col}$ do not happen, there can be at most $k \cdot m$ salts a that satisfy that $\mathcal{A}_1$ makes a query $h(a, M') = z$ and $h(z, M'') = z$. The probability that the salt a that is sampled uniformly at random is among one of those at most $k \cdot m$ salts is at most $\frac{mk}{2^n}$.

From our calculations in the proof of Theorem 4, we have that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$, $\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n$. We also know that for $k = \max\left(n, \frac{2eT_1}{2^n}\right)$, $\Pr\left[\mathsf{off\text{-}sl\text{-}}k\right] \leqslant 1/2^n$. We set m, k to these values and obtain from (19) that

$$\Pr\left[\mathsf{sos}_{\mathsf{off}}\right] \leqslant \frac{2}{2^n} + \frac{n^2}{2^n} + \frac{6enT_1}{2^{2n}} + \frac{8e^2T_1^2}{2^{3n}} \ . \tag{20}$$

This is because for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$, $k = \max\left(n, \frac{2eT_1}{2^n}\right)$,

$$k \cdot m \leqslant n^2 + 6enT_1/2^n + 8e^2T_1^2/2^{2n} \ .$$

Plugging (17), (18) and (20) into (16), we get that

$$\Pr\left[\mathsf{sos}\right] \leqslant \frac{T_2 + 3 + nT_2 + n^2}{2^n} + \frac{2eT_1T_2 + 6enT_1}{2^{2n}} + \frac{8e^2T_1^2}{2^{3n}} \ .$$

□

5.5 Proof of Lemma 6

Proof. We define the three following events.

1. $\mathsf{bulb}_{\mathsf{off}}$: $\mathcal{A}_1$ made queries $h(a, M) = y$, $h(a, M') = y$ for some $M \neq M'$ and y
2. $\mathsf{bulb}_{\mathsf{offon}}$: $\mathcal{A}_1$ made queries $h(a, M) = y$, and $\mathcal{A}_2$ made a query with answer y for some M, y
3. $\mathsf{bulb}_{\mathsf{on}}$: $\mathcal{A}_2$ made queries $h(a, M) = y$, $h(a, M') = y$ for some $M \neq M'$ and y

Observe that bulb happens only if at least one of $\mathsf{bulb}_{\mathsf{off}}$, $\mathsf{bulb}_{\mathsf{offon}}$, $\mathsf{bulb}_{\mathsf{on}}$ happen. Therefore

$$\Pr[\mathsf{bulb}] \leqslant \Pr[\mathsf{bulb}_{\mathsf{off}}] + \Pr[\mathsf{bulb}_{\mathsf{offon}}] + \Pr[\mathsf{bulb}_{\mathsf{on}}] \ . \tag{21}$$

The rest of the proof consists of upper bounding these probabilities one by one. We begin with $\Pr[\mathsf{bulb}_{\mathsf{on}}]$. Observe that $\mathsf{bulb}_{\mathsf{on}}$ happens only if $\mathcal{A}_2$ makes two queries that have the same answer. The probability of any two queries of $\mathcal{A}$ having the same answer is $1/2^n$. Using a union bound over all pairs of queries of $\mathcal{A}_2$, we have that

$$\Pr[\mathsf{bulb}_{\mathsf{on}}] \leqslant \frac{\binom{T_2}{2}}{2^n} \ . \tag{22}$$

We next upper bound $\Pr[\mathsf{bulb}_{\mathsf{offon}}]$. Let Q_a be the random variable denoting the number of queries $\mathcal{A}_1$ makes with salt a. Using total probability

$$\begin{aligned}\Pr[\mathsf{bulb}_{\mathsf{offon}}] &= \sum_{i=1}^{T_1} \Pr[Q_a = k] \Pr\left[\mathsf{bulb}_{\mathsf{offon}} \,\middle|\, Q_a = k\right] \\ &= \sum_{i=1}^{T_1} \Pr[Q_a = k] \cdot \frac{kT_2}{2^n} \\ &= \frac{T_2}{2^n} \cdot \mathsf{E}[Q_a] = \frac{T_1 T_2}{2^{2n}} \ . \end{aligned} \tag{23}$$

The second equality above follows because if $\mathcal{A}_1$ makes k queries with salt a, the probability that $\mathsf{bulb}_{\mathsf{offon}}$ happens is at most $kT_2/2^n$ using a union bound over all queries of $\mathcal{A}_2$. The final equality uses the fact that $\mathsf{E}[Q_a] = T_1/2^n$, because $\mathcal{A}_1$ makes T_1 queries and a is sampled uniformly at random.

Finally, we upper bound $\Pr[\mathsf{bulb}_{\mathsf{off}}]$. We define an event $\mathsf{off\text{-}bulbs\text{-}}k$ as follows: there is a set of at least k distinct salts $a_1, \ldots, a_k$ such that for each a_i, $\mathcal{A}_1$ has made a pair of queries $h(a_i, M_i) = z$ and $h(a_i, M_i') = z$ for $M_i \neq M_i'$.

We have that for any k,

$$\Pr[\mathsf{bulb}_{\mathsf{off}}] \leqslant \Pr\left[\mathsf{bulb}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}bulbs\text{-}}k\right] + \Pr[\mathsf{off\text{-}bulbs\text{-}}k] \ . \tag{24}$$

Since $\mathsf{bulb}_{\mathsf{off}}$ happens if for the salt a that is chosen uniformly at random from $\{0,1\}^n$, $\mathcal{A}_1$ had queried $h(a, M), h(a, M')$ that have the same answer, we have that $\Pr\left[\mathsf{bulb}_{\mathsf{off}} \,\middle|\, \neg\mathsf{off\text{-}bulbs\text{-}}k\right] \leqslant k/2^n$. We upper bound $\Pr[\mathsf{off\text{-}bulbs\text{-}}k]$ using a compression argument.

Note that the event $\mathsf{off\text{-}bulbs\text{-}}k$ is similar to the event $\mathsf{off\text{-}oneblk\text{-}}k$ we defined in the proof of Theorem 4, the only difference being the salt length was s there and is n here. However, $\Pr[\mathsf{off\text{-}oneblk\text{-}}k]$ did not depend on s, hence we can prove the same bound for $\Pr[\mathsf{off\text{-}bulbs\text{-}}k]$. We showed in that proof that for $k = \max\left(\frac{eT_1}{2^{n/2}}, n/2\right)$, $\Pr[\mathsf{off\text{-}oneblk\text{-}}k] \leqslant 1/2^n$.

Hence from (24) we have

$$\Pr[\mathsf{bulb}_{\mathsf{off}}] \leqslant \frac{eT_1}{2^{3n/2}} + \frac{n}{2^{n+1}} + \frac{1}{2^n} \ . \tag{25}$$

Plugging (22), (23) and (25) into (21) gives us that

$$\Pr[\mathsf{bulb}] \leqslant \frac{\binom{T_2}{2}}{2^n} + \frac{T_2 T_1}{2^{2n}} + \frac{eT_1}{2^{3n/2}} + \frac{n}{2^{n+1}} + \frac{1}{2^n} \ .$$

□

5.6 Proof of Lemma 7

Proof. We define the five following events.

1. $\mathsf{bos}_{\mathsf{off}}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$, $h(y, M''') = z$ for some $M', M'' \neq M'''$ and y
2. $\mathsf{bos}_{\mathsf{offon},1}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$ and $\mathcal{A}_2$ makes a query with answer z for some z.
3. $\mathsf{bos}_{\mathsf{offon},2}$: $\mathcal{A}_1$ makes a query $h(y, M') = z$, and $\mathcal{A}_2$ makes a query with answer z and another with answer y, for some y, z, M'.
4. $\mathsf{bos}_{\mathsf{offon},3}$: $\mathcal{A}_1$ makes queries $h(y, M') = z$, and $h(y, M'') = z$ and $\mathcal{A}_2$ makes query with answer z for some $M' \neq M'', y, z$.
5. $\mathsf{bos}_{\mathsf{on}}$: $\mathcal{A}_2$ makes queries two queries that have the same answer.

Observe that bulb happens only if at least one of $\mathsf{bos}_{\mathsf{off}}$, $\mathsf{bos}_{\mathsf{offon},1}$, $\mathsf{bos}_{\mathsf{offon},2}$, $\mathsf{bos}_{\mathsf{offon},3}$, $\mathsf{bos}_{\mathsf{on}}$ happens. To see why this is true, observe that for bos to happen $\mathcal{A}$ has to make queries $q_1 = h(a, M') = y$, $q_2 = h(y, M'') = z$, $q_3 = h(y, M''') = z$ for $M'' \neq M'''$, and some y, z. We show that all the possibilities are covered.

- If q_2, q_3 are both online, then $\mathsf{bos}_{\mathsf{on}}$ happens.
- If one of q_2, q_3 is online, the other offline and
  - if q_1 is offline then $\mathsf{bos}_{\mathsf{offon},1}$ happens
  - if q_1 is online, then $\mathsf{bos}_{\mathsf{offon},1}$ happens
- If both q_2, q_3 are offline and
  - q_1 is online, then $\mathsf{bos}_{\mathsf{offon},3}$ happens
  - q_1 is offline, then $\mathsf{bos}_{\mathsf{off}}$ happens

Therefore,

$$\Pr[\mathsf{bos}] \leqslant \Pr[\mathsf{bos}_{\mathsf{off}}] + \Pr[\mathsf{bos}_{\mathsf{offon},1}] + \Pr[\mathsf{bos}_{\mathsf{offon},2}] + \Pr[\mathsf{bos}_{\mathsf{offon},3}] + \Pr[\mathsf{bos}_{\mathsf{on}}] \ . \tag{26}$$

We upper bound these probabilities one-by-one. First off, we upper bound $\Pr[\mathsf{bos}_{\mathsf{on}}]$. For every pair of queries made by $\mathcal{A}_2$, the probability that they have the same answer is $1/2^n$. From a union bound over all pairs of queries by $\mathcal{A}_1$ it follows that

$$\Pr[\mathsf{bos}_{\mathsf{on}}] \leqslant \frac{\binom{T_2}{2}}{2^n} \ . \tag{27}$$

Next, we upper bound $\Pr[\mathsf{bos}_{\mathsf{offon},2}]$. Fix a query $q = h(y, M) = z$ made by $\mathcal{A}_1$ and a pair of queries q', q'' by $\mathcal{A}_2$. The probability that the answer of q' is y and that of q'' is z, is $1/2^{2n}$. Taking a union bound over all possible q, q', q'' we have that

$$\Pr[\mathsf{bos}_{\mathsf{offon},2}] \leqslant \frac{T_2^2 T_1}{2^{2n}} \ . \tag{28}$$

Next, we upper bound $\Pr[\mathsf{bos}_{\mathsf{offon},3}]$. Recall the event $\mathsf{off\text{-}bulbs\text{-}}k$ defined in the proof of Lemma 6: there is a set of at least k distinct salts $a_1, \dots, a_k$ such that for each a_i, $\mathcal{A}_1$ has made a pair of queries $h(a_i, M_i) = z$ and $h(a_i, M_i') = z$. For a salt a_i, we refer to the adversary querying $h(a_i, M_i) = z$ and $h(a_i, M_i') = z$ for some $M_i \neq M_i'$ as a bulb query for the salt. In other words $\mathsf{off\text{-}bulbs\text{-}}k$ is the event that the adversary makes bulb queries for at least k salts.

We have that

$$\Pr[\mathsf{bos}_{\mathsf{offon},3}] \leqslant \Pr\left[\mathsf{bos}_{\mathsf{offon},3} \,\middle|\, \neg\mathsf{off\text{-}bulbs\text{-}}k\right] + \Pr[\mathsf{off\text{-}bulbs\text{-}}k] \ .$$

If $\mathcal{A}_1$ makes bulb queries for at most k salts, $\mathsf{bos}_{\mathsf{offon},3}$ happens only if $\mathcal{A}_1$ makes a query whose answer is among these at most k salts. Hence, $\Pr\left[\mathsf{bos}_{\mathsf{offon},3} \,\middle|\, \neg\mathsf{off\text{-}bulbs\text{-}}k\right] \leqslant kT_1/2^n$. We know from the analysis in Lemma 6, that for $k = \max\left(eT_1/2^{n/2}, n/2\right)$, $\Pr[\mathsf{off\text{-}bulbs\text{-}}k] \leqslant 1/2^n$. Therefore, it follows that

$$\begin{aligned}\Pr[\mathsf{bos}_{\mathsf{offon},3}] &\leqslant \frac{1}{2^n} + \frac{T_2}{2^n} \cdot \left(\frac{n}{2} + \frac{eT_1}{2^{n/2}}\right) \\ &= \frac{nT_2/2 + 1}{2^n} + \frac{eT_1T_2}{2^{3n/2}} \ . \end{aligned} \tag{29}$$

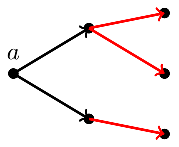


Fig. 8: Illustrating successor queries of a salt in the query graph. The queries marked in red to a salt a are successor queries for a.

Next, we upper bound $\Pr[\mathsf{bos}_{\mathsf{offon},1}]$. We define the notion of a query being $\mathcal{A}_1$ being a *successor* query for a particular salt a. We say that a query $q = h(y, M) = z$ made by $\mathcal{A}_1$ is a successor query for a salt a if

- $y \neq a, z \neq a, y \neq z$
- there is a query $q' = h(a, M) = y$ made by $\mathcal{A}_1$ for some M

In Fig. 8, we illustrate how successor queries to a salt look like in the query graph of $\mathcal{A}$.

Let Q_a denote the number of successor queries for a salt a. Note that $\mathsf{bos}_{\mathsf{offon},1}$ happens only if $\mathcal{A}_2$ makes a query whose answer is same as the answer of one of the successor queries made of a. We have that

$$\begin{aligned}\Pr[\mathsf{bos}_{\mathsf{offon},1}] &= \sum_{k=1}^{T_1} \Pr[Q_a = k] \cdot \frac{kT_1}{2^n} \\ &= \mathsf{E}[Q_a] \frac{T_1}{2^n} .\end{aligned}$$

We prove an upper bound on Q_a. Recall the event $m + 1$-col: $(m + 1)$-col happens if the $\mathcal{A}_1$ has made $m + 1$ distinct h all of which have the same answer. We have using total expectation

$$\mathsf{E}[Q_a] \leqslant \mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] + \mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \Pr[(m+1)\text{-}\mathsf{col}] . \tag{30}$$

We claim that $\mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] \leqslant mT_1/2^n$. This is because we have that $\mathsf{E}\left[\sum_{a\in\{0,1\}^n} Q_a \,\middle|\, \neg m+1\text{-}\mathsf{col}\right] \leqslant mT_1$- since if there is no $(m+1)$-multicollision, a query can be a successor query to at most m different salts. Since a is sampled uniformly at random from $\{0,1\}^n$, the claim follows.

We have from earlier analysis in Theorem 4 that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$

$$\Pr[(m+1)\text{-}\mathsf{col}] \leqslant 1/2^n .$$

Moreover $\mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \leqslant T_1$ since a salt cannot have more than T_1 successor queries. Therefore, plugging this into (30)

$$\mathsf{E}[Q_a] \leqslant \frac{4eT_1^2}{2^{2n}} + \frac{nT_1}{2^n} + \frac{T_1}{2^{2n}} .$$

So, we have

$$\Pr[\mathsf{bos}_{\mathsf{offon},1}] = \mathsf{E}[Q_a]\frac{T_2}{2^n} \leqslant \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} . \tag{31}$$

Finally, we upper bound $\Pr[\mathsf{bos}_{\mathsf{off}}]$. We have that for any k, m

$$\Pr[\mathsf{bos}_{\mathsf{off}}] \leqslant \Pr\left[\mathsf{bos}_{\mathsf{off}} \,\middle|\, \neg(m+1)\text{-}\mathsf{col} \wedge \neq \mathsf{off\text{-}bulbs}\text{-}k\right] + \Pr[(m+1)\text{-}\mathsf{col}] + \Pr[\mathsf{off\text{-}bulbs}\text{-}k] .$$

We claim that $\Pr\left[\mathsf{bos}_{\mathsf{off}} \,\middle|\, \neg(m+1)\text{-}\mathsf{col} \wedge \neq \mathsf{off\text{-}bulbs}\text{-}k\right] \leqslant k \cdot m/2^n$, because if there are no $m + 1$-multi-collisions and the adversary finds bulbs for at most k salts, there are at most $k \cdot m$ salts such that these when sampled cause $\mathsf{bos}_{\mathsf{off}}$. Since a is sampled uniformly at random, it follows that this probability is at most $k \cdot m/2^n$.

We let $k = \max\left(\frac{eT_1}{2^{n/2}}, n/2\right)$, $m = \max\left(n, \frac{4eT_1}{2^n}\right)$, we have that

$$k \cdot m \leqslant n^2/2 + \frac{eT_1 n}{2^{n/2}} + \frac{2eT_1 n}{2^n} + \frac{4e^2T_1^2}{2^{3n/2}};.$$

This implies

$$\begin{aligned}\Pr[\mathsf{bos}_{\mathsf{off}}] \leqslant& \frac{1}{2^n} + \frac{1}{2^n} + \frac{1}{2^n} \cdot \left(n^2/2 + \frac{eT_1 n}{2^{n/2}} + \frac{2eT_1 n}{2^n} + \frac{4e^2T_1^2}{2^{3n/2}}\right) \\ =& \frac{n^2/2+2}{2^n} + \frac{enT_1}{2^{3n/2}} + \frac{2enT_1}{2^{2n}} + \frac{4e^2T_1^2}{2^{5n/2}}\ . \end{aligned} \tag{32}$$

Plugging (27) to (29), (31) and (32) into (26) we get

$$\begin{aligned}\Pr[\mathsf{bos}] \leqslant& \frac{\binom{T_2}{2} + nT_2/2 + n^2/2 + 4}{2^n} + \frac{eT_1T_2 + enT_1}{2^{3n/2}} + \frac{nT_1T_2 + T_1T_2^2 + 2enT_1}{2^{2n}} \\ &+ \frac{4e^2T_1^2}{2^{5n/2}} + \frac{4eT_1^2T_2}{2^{3n}}\ .\end{aligned}$$

□

5.7 Proof of Lemma 8

Proof (Lemma 8). We define the six following events.

1. $\mathsf{tri}_{\mathsf{off}}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$, $h(a, M''') = z$ for some M', M'', M''' and $y \neq a$
2. $\mathsf{tri}_{\mathsf{offon},1}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$ and $\mathcal{A}_2$ makes a query with answer z for some z.
3. $\mathsf{tri}_{\mathsf{offon},2}$: $\mathcal{A}_1$ makes a query $h(y, M') = z$, and $\mathcal{A}_2$ makes a query with answer z and another with answer y, for some y, z, M'.
4. $\mathsf{tri}_{\mathsf{offon},3}$: $\mathcal{A}_1$ makes queries $h(a, M') = z$, and $\mathcal{A}_2$ makes query with answer z for some M', z.
5. $\mathsf{tri}_{\mathsf{offon},4}$: $\mathcal{A}_1$ makes queries $h(a, M') = z$, $h(y, M'') = z$ and $\mathcal{A}_2$ makes a query with answer y for some y, z, M', M''
6. $\mathsf{tri}_{\mathsf{on}}$: $\mathcal{A}_2$ makes queries two queries that have the same answer.

Observe that tri happens only if at least one of $\mathsf{tri}_{\mathsf{off}}$, $\mathsf{tri}_{\mathsf{offon},1}$, $\mathsf{tri}_{\mathsf{offon},2}$, $\mathsf{tri}_{\mathsf{offon},3}$, $\mathsf{tri}_{\mathsf{offon},4}$, $\mathsf{tri}_{\mathsf{on}}$ happens. To see why this is true, observe that for tri to happen $\mathcal{A}$ has to make queries $q_1 = h(a, M') = y$, $q_2 = h(y, M'') = z$, $q_3 = h(a, M''') = z$ for M', M'', M''', y, z. We show that all the possibilities are covered.

– If q_2, q_3 are both online, then $\mathsf{tri}_{\mathsf{on}}$ happens.
– If q_2 is online, q_3 is offline then $\mathsf{tri}_{\mathsf{offon},3}$ happens
– If q_3 is online, q_2 is offline and
  • q_1 is online, then $\mathsf{tri}_{\mathsf{offon},2}$ happens
  • q_1 is offline, then $\mathsf{tri}_{\mathsf{offon},1}$ happens
– If both q_2, q_3 are offline and
  • q_1 is online, then $\mathsf{tri}_{\mathsf{offon},4}$ happens
  • q_1 is offline, then $\mathsf{tri}_{\mathsf{off}}$ happens

Therefore

$$\begin{aligned}\Pr[\mathsf{tri}] \leqslant& \Pr[\mathsf{tri}_{\mathsf{off}}] + \Pr[\mathsf{tri}_{\mathsf{offon},1}] + \Pr[\mathsf{tri}_{\mathsf{offon},2}] + \Pr[\mathsf{tri}_{\mathsf{offon},3}] \\ &+ \Pr[\mathsf{tri}_{\mathsf{offon},4}] + \Pr[\mathsf{tri}_{\mathsf{on}}]\ .\end{aligned} \tag{33}$$

We upper bound these probabilities one-by-one.

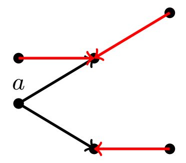


Fig. 9: Illustrating meeting queries of a salt in the query graph. The queries marked in red to a salt a are meeting queries for a.

We start with upper bounding $\Pr[\mathsf{tri}_{\mathsf{on}}]$. Observe that $\mathsf{tri}_{\mathsf{on}}$ is the same event as $\mathsf{bos}_o n$ in the proof of Lemma 7. Therefore, from (27), it follows that

$$\Pr[\mathsf{tri}_{\mathsf{on}}] \leqslant \frac{\binom{T_2}{2}}{2^n} . \tag{34}$$

Next we upper bound $\Pr[\mathsf{tri}_{\mathsf{offon},1}]$. Observe that $\mathsf{tri}_{\mathsf{offon},1}$ is the same event as $\mathsf{bos}_{\mathsf{offon},1}$ in the proof of Lemma 7. Therefore, from (31), it follows that

$$\Pr[\mathsf{tri}_{\mathsf{offon},1}] \leqslant \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} . \tag{35}$$

Next we upper bound $\Pr[\mathsf{tri}_{\mathsf{offon},2}]$. Observe that $\mathsf{tri}_{\mathsf{offon},2}$ is the same event as $\mathsf{bos}_{\mathsf{offon},2}$ in the proof of Lemma 7. Therefore, from (28), it follows that

$$\Pr[\mathsf{tri}_{\mathsf{offon},2}] \leqslant \frac{T_2^2T_1}{2^{2n}} . \tag{36}$$

Next we upper bound $\Pr[\mathsf{tri}_{\mathsf{offon},3}]$. Observe that $\mathsf{tri}_{\mathsf{offon},3}$ is the same event as $\mathsf{bulb}_{\mathsf{offon}}$ in the proof of Lemma 6. Therefore, from (23), it follows that

$$\Pr[\mathsf{tri}_{\mathsf{offon},3}] \leqslant \frac{T_1T_2}{2^{2n}} . \tag{37}$$

Next we upper bound $\Pr[\mathsf{tri}_{\mathsf{offon},4}]$. We define the notion of a query being $\mathcal{A}_1$ being a *meeting* query for a particular salt a. We say that a query $q = h(y, M) = z$ $(y \neq a)$ made by $\mathcal{A}_1$ is a meeting query for a salt a if:

- $y \neq a, y \neq z, a \neq z$
- there is a query $q' = h(a, M) = z$ made by $\mathcal{A}_1$

In Fig. 9, we illustrate how meeting queries to a salt look like in the query graph of $\mathcal{A}$.

Let Q_a denote the number of meeting queries for a salt a. Note that $\mathsf{tri}_{\mathsf{offon},4}$ happens only if $\mathcal{A}_2$ makes a query whose answer is same as the answer of one of the meeting queries made of a. We have that

$$\begin{aligned}\Pr[\mathsf{tri}_{\mathsf{offon},4}] &= \sum_{k=1}^{T_1} \Pr[Q_a = k] \cdot \frac{kT_1}{2^n} \\ &= \mathsf{E}[Q_a]\frac{T_1}{2^n} .\end{aligned}$$

We prove an upper bound on $\mathsf{E}[Q_a]$. Recall the event $m+1\text{-}\mathsf{col}$: $(m+1)\text{-}\mathsf{col}$ happens if the $\mathcal{A}_1$ has made $m+1$ distinct h all of which have the same answer. We have using total expectation

$$\mathsf{E}[Q_a] \leqslant \mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] + \mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \Pr[(m+1)\text{-}\mathsf{col}] . \tag{38}$$

We claim that $\mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] \leqslant mT_1/2^n$. This is because we have that $\mathsf{E}\left[\sum_{a\in\{0,1\}^n} Q_a \,\middle|\, \neg m+1\text{-}\mathsf{col}\right] \leqslant mT_1$ – since if there is no $(m+1)$-multicollision, a query can be a meeting query to at most m different salts. Since a is sampled uniformly at random from $\{0,1\}^n$, the claim follows.

We have from earlier analysis in Theorem 4 that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$

$$\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n \ .$$

Moreover $\mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \leqslant T_1$ since a salt cannot have more than T_1 meeting queries. Therefore, plugging this into (38)

$$\mathsf{E}\left[Q_a\right] \leqslant \frac{4eT_1^2}{2^{2n}} + \frac{nT_1}{2^n} + \frac{T_1}{2^{2n}} \ .$$

So, we have

$$\Pr\left[\mathsf{tri}_{\mathsf{offon},4}\right] = \mathsf{E}\left[Q_a\right]\frac{T_2}{2^n} \leqslant \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} \ . \tag{39}$$

We finally upper bound $\Pr\left[\mathsf{tri}_{\mathsf{off}}\right]$ in Lemma 10.

Lemma 10.

$$\begin{aligned}\Pr\left[\mathsf{tri}_{\mathsf{off}}\right] \leqslant & \frac{12e(2)^{1/2}T_1^{3/2}}{2^{2n}} + \frac{3(8en)^{1/2}T_1}{2^{3n/2}} + \frac{3(24e)^{1/2}T_1}{2^{3n/2}} + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} \\ & + \frac{18n}{2^n} + \frac{8}{2^n} \ .\end{aligned} \tag{40}$$

Plugging (34) to (37), (39) and (40) in (33) we have that

$$\begin{aligned}\Pr\left[\mathsf{tri}\right] \leqslant & \frac{\binom{T_2}{2}}{2^n} + \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} + \frac{T_2^2T_1}{2^{2n}} + \frac{T_1T_2}{2^{2n}} + \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} \\ & + \frac{T_1T_2}{2^{3n}} + \frac{12e(2)^{1/2}T_1^{3/2}}{2^{2n}} + \frac{3(8en)^{1/2}T_1}{2^{3n/2}} + \frac{3(24e)^{1/2}T_1}{2^{3n/2}} + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} \\ & + \frac{18n}{2^n} + \frac{8}{2^n} \\ \leqslant & \frac{\binom{T_2}{2} + 18n + 8}{2^n} + \frac{3(24e)^{1/2}T_1 + 3(8en)^{1/2}T_1}{2^{3n/2}} + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} \\ & + \frac{2nT_1T_2 + T_1T_2 + T_2^2T_1 + 12e(2)^{1/2}T_1^{3/2}}{2^{2n}} + \frac{8eT_1^2T_2 + 2T_1T_2}{2^{3n}}\end{aligned}$$

□

We defer the proof of Lemma 10 to Section 5.9.

5.8 Proof of Lemma 9

Proof (Lemma 9). We define the six following events.

1. $\mathsf{dia}_{\mathsf{off}}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$, $h(a, M''') = y'$, $h(a, M'''') = z$ for some $M' \neq M''', M'' \neq M''''$ and $y \neq y'$
2. $\mathsf{dia}_{\mathsf{offon},1}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$ and $\mathcal{A}_2$ makes a query with answer z for some z.
3. $\mathsf{dia}_{\mathsf{offon},2}$: $\mathcal{A}_1$ makes a query $h(y, M') = z$, and $\mathcal{A}_2$ makes a query with answer z and another with answer y, for some y, z, M'.
4. $\mathsf{dia}_{\mathsf{offon},3}$: $\mathcal{A}_1$ makes queries $h(a, M') = y$, $h(y, M'') = z$, $h(y', M''') = z$, $\mathcal{A}_2$ makes a query with answer y' for some $y \neq y'$, M', M'', M'''
5. $\mathsf{dia}_{\mathsf{offon},4}$: $\mathcal{A}_1$ makes queries $h(y, M') = z$, $h(y', M'') = z$, $\mathcal{A}_2$ makes queries with answer y', y for some $y \neq y'$, M', M''
6. $\mathsf{dia}_{\mathsf{on}}$: $\mathcal{A}_2$ makes queries two queries that have the same answer.

Observe that dia happens only if at least one of $\mathsf{dia}_{\mathsf{off}}$, $\mathsf{dia}_{\mathsf{offon},1}$, $\mathsf{dia}_{\mathsf{offon},2}$, $\mathsf{dia}_{\mathsf{offon},3}$, $\mathsf{dia}_{\mathsf{offon},4}$, $\mathsf{dia}_{\mathsf{on}}$ happens. To see why this is true, observe that for dia to happen $\mathcal{A}$ has to make queries $q_1 = h(a, M') = y$, $q_2 = h(y, M'') = z$, $q_3 = h(a, M''') = y'$, $q_4 = h(y', M'''') = z$ for $M', M'', M''', M'''', y, y', z$ such that $y \neq y'$, $M' \neq M'''$, $M'' \neq M''''$. We show that all the possibilities are covered.

- If q_2, q_4 are both online, then $\mathsf{dia}_{\mathsf{on}}$ happens.
- If q_2 is online, q_4 is offline and
  - q_3 is online, then $\mathsf{dia}_{\mathsf{offon},2}$ happens
  - q_3 is offline, then $\mathsf{dia}_{\mathsf{offon},1}$ happens
- If q_2 is offline, q_4 is online and
  - q_1 is online, then $\mathsf{dia}_{\mathsf{offon},2}$ happens
  - q_1 is offline, then $\mathsf{dia}_{\mathsf{offon},1}$ happens
- If q_2, q_4 are both offline
  - q_1, q_3 are both online, then $\mathsf{tri}_{\mathsf{offon},4}$ happens
  - one of q_1, q_3 is offline, the other offline, then $\mathsf{tri}_{\mathsf{offon},3}$ happens
  - q_1, q_3 are both offline, then $\mathsf{tri}_{\mathsf{off}}$ happens

Therefore,

$$\begin{aligned}\Pr[\mathsf{dia}] \leqslant \Pr[\mathsf{dia}_{\mathsf{off}}] + \Pr[\mathsf{dia}_{\mathsf{offon},1}] + \Pr[\mathsf{dia}_{\mathsf{offon},2}] + \Pr[\mathsf{dia}_{\mathsf{offon},3}] \\ + \Pr[\mathsf{dia}_{\mathsf{offon},4}] + \Pr[\mathsf{dia}_{\mathsf{on}}] \ .\end{aligned} \tag{41}$$

We upper bound these probabilities one-by-one.

We start with upper bounding $\Pr[\mathsf{dia}_{\mathsf{on}}]$. Observe that $\mathsf{dia}_{\mathsf{on}}$ is the same event as $\mathsf{bos}_o n$ in the proof of Lemma 7. Therefore, from (27), it follows that

$$\Pr[\mathsf{dia}_{\mathsf{on}}] \leqslant \frac{\binom{T_2}{2}}{2^n} \ . \tag{42}$$

Next we upper bound $\Pr[\mathsf{dia}_{\mathsf{offon},1}]$. Observe that $\mathsf{dia}_{\mathsf{offon},1}$ is the same event as $\mathsf{bos}_{\mathsf{offon},1}$ in the proof of Lemma 7. Therefore, from (31), it follows that

$$\Pr[\mathsf{dia}_{\mathsf{offon},1}] \leqslant \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} \ . \tag{43}$$

Next we upper bound $\Pr[\mathsf{dia}_{\mathsf{offon},2}]$. Observe that $\mathsf{dia}_{\mathsf{offon},2}$ is the same event as $\mathsf{bos}_{\mathsf{offon},2}$ in the proof of Lemma 7. Therefore, from (28), it follows that

$$\Pr[\mathsf{dia}_{\mathsf{offon},2}] \leqslant \frac{T_2^2T_1}{2^{2n}} \ . \tag{44}$$

Next we upper bound $\Pr[\mathsf{dia}_{\mathsf{offon},3}]$. We first define the notion of a *successor-hitting* query for a salt. We say that a query $q = h(y, M) = z$ made by $\mathcal{A}_1$ is a successor-hitting query for a salt a if

- $z \neq a, y \neq a, y \neq z$
- there are queries $q' = h(a, M') = b$, $q'' = h(b, M'') = z$ with $b \neq a, z \neq b$ made by $\mathcal{A}_1$ for some M, M'

Informally, a query q is a successor-hitting query to salt a if there query $q' = h(y', M') = z$ such that q' is a successor query to a and $q' \neq q$. In Fig. 10, we illustrate how successor-hitting queries to a salt look like in the query graph of $\mathcal{A}$.

Let Q_a denote the number of successor-hitting queries for a salt a. Note that $\mathsf{dia}_{\mathsf{offon},3}$ happens only if $\mathcal{A}_2$ makes a query whose answer is same as the answer of one of the successor-hitting queries made of a. We have that

$$\begin{aligned}\Pr[\mathsf{dia}_{\mathsf{offon},3}] &= \sum_{k=1}^{T_1} \Pr[Q_a = k] \cdot \frac{kT_1}{2^n} \\ &= \mathsf{E}[Q_a] \frac{T_1}{2^n} \ .\end{aligned}$$

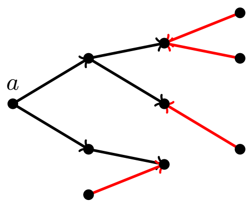


Fig. 10: Illustrating successor-hitting queries of a salt in the query graph. The queries marked in red to a salt a are successor-hitting queries for a.

We prove an upper bound on $\mathsf{E}[Q_a]$. Recall the event $m+1$-col: $(m+1)$-col happens if the $\mathcal{A}_1$ has made $m+1$ distinct h all of which have the same answer. We have using total expectation

$$\mathsf{E}[Q_a] \leqslant \mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] + \mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \Pr\left[(m+1)\text{-}\mathsf{col}\right] . \tag{45}$$

We claim that $\mathsf{E}\left[Q_a \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] \leqslant m^2 T_1/2^n$. This is because we have that $\mathsf{E}\left[\sum_{a\in\{0,1\}^n} Q_a \,\middle|\, \neg m+1\text{-}\mathsf{col}\right] \leqslant m^2 T_1$ – since if there is no $(m+1)$-multicollision, a query can be a sucessor-hitting query to at most m^2 different salts. Since a is sampled uniformly at random from $\{0,1\}^n$, the claim follows.

We have from earlier analysis in Theorem 4 that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$

$$\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n .$$

Moreover $\mathsf{E}\left[Q_a \,\middle|\, (m+1)\text{-}\mathsf{col}\right] \leqslant T_1$ since a salt cannot have more than T_1 successor-hitting queries. Therefore, plugging this into (45)

$$\mathsf{E}[Q_a] \leqslant \frac{4eT_1^3}{2^{3n}} + \frac{n^2 T_1}{2^n} + \frac{T_1}{2^{2n}} .$$

So, we have

$$\Pr\left[\mathsf{dia}_{\mathsf{offon},3}\right] = \mathsf{E}[Q_a]\frac{T_2}{2^n} \leqslant \frac{4eT_1^3 T_2}{2^{4n}} + \frac{n^2 T_1 T_2}{2^{2n}} + \frac{T_1 T_2}{2^{3n}} . \tag{46}$$

Next, we upper bound $\Pr\left[\mathsf{dia}_{\mathsf{offon},4}\right]$. We have that

$$\Pr\left[\mathsf{dia}_{\mathsf{offon},4}\right] \leqslant \Pr\left[\mathsf{dia}_{\mathsf{offon},4} \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] + \Pr\left[(m+1)\text{-}\mathsf{col}\right] .$$

We claim that $\Pr\left[\mathsf{dia}_{\mathsf{offon},4} \,\middle|\, \neg(m+1)\text{-}\mathsf{col}\right] \leqslant mT_1T_2^2/2^{2n}$. This is because, given there are no $(m+1)$-multicollision there can be at most mT_1 pairs of queries that collide. For $\mathsf{dia}_{\mathsf{offon},4}$ to happen, $\mathcal{A}_2$ needs to make two queries such that the answer of the queries are the input salts of a pair of colliding offline queries. Using a union bound over all colliding offline queries and pairs of queries by $\mathcal{A}_2$, the claim follows. Moreover, we have from earlier analysis in Theorem 4 that for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$

$$\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n .$$

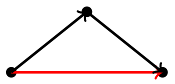
Therefore, we have that

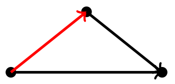
$$\Pr\left[\mathsf{dia}_{\mathsf{offon},4}\right] \leqslant \frac{4eT_1^2 T_2^2}{2^{3n}} + \frac{nT_1T_2^2}{2^{2n}} + \frac{1}{2^n} . \tag{47}$$

We finally upper bound $\Pr\left[\mathsf{dia}_{\mathsf{off}}\right]$ in Lemma 11.

Lemma 11.

$$\begin{aligned}\Pr\left[\mathsf{dia}_{\mathsf{off}}\right] \leqslant & \frac{30n+15}{2^n} + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{(8e)^{1/2}nT_1 + 10(2e)^{1/2}nT_1}{2^{3n/2}} \\ & + \frac{240e^{2/3}T_1^2}{2^{7n/3}} + \frac{8(2)^{1/2}e^{3/2}T_1^2 + 40(2e)^{1/2}T_1^2}{2^{5n/2}} .\end{aligned} \tag{48}$$


(a) Category 1 Triangle.


(b) Category 2 Triangle.

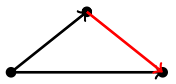

(c) Category 3 Triangle.

Fig. 11: The different categories of triangles. The edge colored red is the last of the three queries made.

Plugging (42) to (44) and (46) to (48) in (41) we have that

$$\begin{aligned}
\Pr[\mathsf{dia}] \leqslant & \frac{\binom{T_2}{2}}{2^n} + \frac{4eT_1^2T_2}{2^{3n}} + \frac{nT_1T_2}{2^{2n}} + \frac{T_1T_2}{2^{3n}} + \frac{T_2^2T_1}{2^{2n}} + \frac{4eT_1^3T_2}{2^{4n}} + \frac{n^2T_1T_2}{2^{2n}} \\
& + \frac{T_1T_2}{2^{3n}} + \frac{4eT_1^2T_2^2}{2^{3n}} + \frac{nT_1T_2^2}{2^{2n}} + \frac{1}{2^n} + \frac{30n+15}{2^n} \\
& + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{(8e)^{1/2}nT_1 + 10(2e)^{1/2}nT_1}{2^{3n/2}} \\
& + \frac{240e^{2/3}T_1^2}{2^{7n/3}} + \frac{8(2)^{1/2}e^{3/2}T_1^2 + 40(2e)^{1/2}T_1^2}{2^{5n/2}} \\
= & \frac{\binom{T_2}{2} + 30n + 16}{2^n} + \frac{4eT_1^2T_2 + 2T_1T_2 + 4eT_1^2T_2^2}{2^{3n}} \\
& + \frac{nT_1T_2 + T_2^2T_1 + n^2T_1T_2 + nT_1T_2^2}{2^{2n}} + \frac{4eT_1^3T_2}{2^{4n}} \\
& + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{(8e)^{1/2}nT_1 + 10(2e)^{1/2}nT_1}{2^{3n/2}} \\
& + \frac{240e^{2/3}T_1^2}{2^{7n/3}} + \frac{8(2)^{1/2}e^{3/2}T_1^2 + 40(2e)^{1/2}T_1^2}{2^{5n/2}} .
\end{aligned}$$

We defer the proof of Lemma 11 to Section 5.10.

5.9 Proof of Lemma 10

Proof (Lemma 10). We say that $\mathcal{A}_1$ has found a triangle for a salt a if it has made queries $h(a, M) = y$, $h(y, M') = z$, $h(a, M'') = z$ for some M, M', M'', y, z. We define the event $\mathsf{off\text{-}tri\text{-}}k$ as $\mathcal{A}_1$ finding triangles for at least k salts. For any k,

$$\Pr[\mathsf{tri_{off}}] \leqslant \Pr[\mathsf{tri_{off}} \mid \neg\mathsf{off\text{-}tri\text{-}}k] + \Pr[\mathsf{off\text{-}tri\text{-}}k] . \tag{49}$$

We claim that $\Pr[\mathsf{tri_{off}} \mid \neg\mathsf{off\text{-}tri\text{-}}k] \leqslant k/2^n$ – this is because if $\mathcal{A}_1$ finds triangles for at most k different salts, $\mathsf{tri_{off}}$ happens with probability at most $k/2^n$.

Towards upper bounding $\mathsf{off\text{-}tri\text{-}}k$, we define a categorization of the triangles. For a triangle on salt a, let the three queries forming the triangle be $q_1 = h(a, M) = y$, $q_2 = h(y, M') = z$, $q_3 = h(a, M'') = z$.

- We say that a triangle is of category 1, if the query q_3 was the last among the three
- We say that a triangle is of category 2, if the query q_1 was the last among the three
- We say that a triangle is of category 3, if the query q_2 was the last among the three

We illustrate the categories of triangles in Fig. 11.

We define the event $\mathsf{off\text{-}1\text{-}tri\text{-}}k_1$ as $\mathcal{A}_1$ find triangles of category 1 for k_1 different salts. We define the event $\mathsf{off\text{-}2\text{-}tri\text{-}}k_2$ as $\mathcal{A}_1$ find triangles of category 2 for k_2 different salts. We define the event $\mathsf{off\text{-}3\text{-}tri\text{-}}k_3$ as $\mathcal{A}_1$ find triangles of category 3 for k_3 different salts. We would have that for any k_1, k_2, k_3,

$$\Pr[\mathsf{off\text{-}tri\text{-}}3\max(k_1,k_2,k_3)] \leqslant \Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1] + \Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2] + \Pr[\mathsf{off\text{-}3\text{-}tri\text{-}}k_3] . \tag{50}$$

We prove the three following claims.

Claim. For any $k_1 \in \mathbb{N}_{>0}$, such that k_1 is a multiple of 2

$$\Pr\left[\textsf{off-1-tri-}k_1\right] \leqslant \left(\frac{16e^2T_1^3}{k_1^2 2^{2n}}\right)^{k_1/2} + \left(\frac{4enT_1^2}{k_1^2 2^n}\right)^{k_1/2} + \frac{1}{2^n} .$$

Claim. For any $k_2 \in \mathbb{N}_{>0}$, such that k_2 is a multiple of 2

$$\Pr\left[\textsf{off-2-tri-}k_2\right] \leqslant \left(\frac{16e^2T_1^3}{k_2^2 2^{2n}}\right)^{k_2/2} + \left(\frac{4enT_1^2}{k_2^2 2^n}\right)^{k_2/2} + \frac{1}{2^n} .$$

Claim. For any $k_3 \in \mathbb{N}_{>0}$, such that k_3 is a multiple of 6

$$\Pr\left[\textsf{off-3-tri-}k_3\right] \leqslant \left(\frac{12eT_1^2}{k_3^2 2^n}\right)^{k_3/6} + \left(\frac{27e^3T_1^4}{k_3^3 2^{2n}}\right)^{k_3/3} .$$

We let $k_1 = \max\left(\left(\frac{32e^2T_1^3}{2^{2n}}\right)^{1/2}, \left(\frac{8enT_1^2}{2^n}\right)^{1/2}, 2n\right)$. We then have that

$$\Pr\left[\textsf{off-1-tri-}k_1\right] \leqslant \frac{3}{2^n} .$$

Similarly, we let $k_2 = \max\left(\left(\frac{32e^2T_1^3}{2^{2n}}\right)^{1/2}, \left(\frac{8enT_1^2}{2^n}\right)^{1/2}, 2n\right)$, and have that

$$\Pr\left[\textsf{off-2-tri-}k_2\right] \leqslant \frac{3}{2^n} .$$

We let $k_3 = \max\left(\left(\frac{24eT_1^2}{2^n}\right)^{1/2}, \left(\frac{54e^3T_1^4}{2^{2n}}\right)^{1/3}, 6n\right)$, and have that

$$\Pr\left[\textsf{off-3-tri-}k_3\right] \leqslant \frac{2}{2^n} .$$

We have that

$$\begin{aligned}\max(k_1,k_2,k_3) &\leqslant \left(\frac{32e^2T_1^3}{2^{2n}}\right)^{1/2} + \left(\frac{8enT_1^2}{2^n}\right)^{1/2} + \left(\frac{24eT_1^2}{2^n}\right)^{1/2}, \left(\frac{54e^3T_1^4}{2^{2n}}\right)^{1/3} + 6n \\ &= \frac{4e(2)^{1/2}T_1^{3/2}}{2^n} + \frac{(8en)^{1/2}T_1}{2^{n/2}} + \frac{(24e)^{1/2}T_1}{2^{n/2}} + \frac{3(2)^{1/3}eT_1^{4/3}}{2^{2n/3}} + 6n .\end{aligned}$$

Let $k = 3 \cdot \left(\frac{4e(2)^{1/2}T_1^{3/2}}{2^n} + \frac{(8en)^{1/2}T_1}{2^{n/2}} + \frac{(24e)^{1/2}T_1}{2^{n/2}} + \frac{3(2)^{1/3}eT_1^{4/3}}{2^{2n/3}} + 6n\right)$. Using (50) we have that

$$\Pr\left[\textsf{off-tri-}k\right] \leqslant \frac{8}{2^n} . \tag{51}$$

Using the fact that we showed earlier that $\Pr\left[\textsf{tri} \,\middle|\, \neg\textsf{off-tri-}k\right] \leqslant k/2^n$, setting k as above, using (49) we have that

$$\Pr\left[\textsf{tri}_{\textsf{off}}\right] \leqslant \frac{12e(2)^{1/2}T_1^{3/2}}{2^{2n}} + \frac{3(8en)^{1/2}T_1}{2^{3n/2}} + \frac{3(24e)^{1/2}T_1}{2^{3n/2}} + \frac{9(2)^{1/3}eT_1^{4/3}}{2^{5n/3}} + \frac{18n}{2^n} + \frac{8}{2^n} .$$

We now prove these claims one by one.

Upper bounding $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1]$. First off, we upper bound $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1]$. We have that for any m,

$$\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1] \leqslant \Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] + \Pr[m+1\text{-}\mathsf{col}] \ . \tag{52}$$

We upper bound $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}]$ using a compression argument. Before giving the compression argument, we recall the notion of successor queries we introduced in the proof of Lemma 7: we say that a query $q = h(y, M)$ $(y \neq a)$ made by $\mathcal{A}_1$ is a successor query for a salt a if there is a query $q' = h(a, M) = y$ made by $\mathcal{A}_1$.

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $m+1$-col happens or $\mathsf{off\text{-}1\text{-}tri\text{-}}k_1$ does not happen, it outputs $\varnothing$
3. It finds a set of salts S of size $k_1/2$ such that for each $a \in S$
   - $\mathcal{A}_1$ found a category 1 triangle for a
   - The salt a has no more than $2mT_1/k_1$ successor queries
4. If no such S is found, return $\varnothing$
5. It marks the q_2, q_3 queries corresponding to the triangle for each salt in S
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_3 query for a salt $a \in S$, then it adds the index of this query in T, and it adds the lexicographical index of q_2 among all *successor* queries for the salt a in the list L_2
   (b) Otherwise it adds $h(a, M)$ to L_1.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_1 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T, it removes the element i on the front of list L_2, and locates the query q which has lexicographic order i among all the successor queries of the salt a. It answers with the answer of q
   (b) Otherwise it removes the element in front of L_1 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_1

Correctness of decoding: First off, we argue that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Note that the encoding algorithm reaches this line only if the event $\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}$ was caused by running $\mathcal{A}_1$, which means there is a set of k_1 salts S' such that $\mathcal{A}_1$ found category one triangles for them. Now, since $\mathcal{A}_1$ makes at most T_1 queries, and $\neq (m+1)$-col happens, there are at most mT_1 salt-successor query pairs. Therefore, for at least $k_1/2$ salts (note that from our assumption k_1 is a multiple of 2, so $k_1/2$ is an integer) in S' such that it has at most $2T_1m/k_1$ successor queries (because otherwise there would be more than mT_1 salt, successor query pairs). Therefore, such a set $S \subseteq S'$ of size $k_1/2$ always exists.

If the encoding algorithm does not return $\varnothing$, it is easy to see the decoding algorithm decodes correctly, because the query answers that the encoding algorithm does not add in the list L_1 can be recovered by the decoding algorithm correctly using S, L_2. Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \ .$$

Observe that the encoding algorithm removes $k_1/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were q_3 queries for different salts. It instead adds an unordered set T of $k_1/2$ values in $[T_1]$, and an ordered list of $k_1/2$ elements where each value is at most $2mT_1/k_1$. Using the compression lemma, we have that

$$\Pr[\text{ Decoding is correct }] \leqslant \frac{(2mT_1/k_1)^{k_1/2}\binom{T_1}{k_1/2}}{2^{nk_1/2}} \leqslant \left(\frac{4emT_1^2}{k_1^2 2^n}\right)^{k_1/2} .$$

Therefore,

$$\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{4emT_1^2}{k_1^2 2^n}\right)^{k_1/2} .$$

We know from previous analyses that $\Pr[(m+1)\text{-}\mathsf{col}] \leqslant 1/2^n$ for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$. Plugging this value of m we have

$$\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{16e^2T_1^3}{k_1^2 2^{2n}}\right)^{k_1/2} + \left(\frac{4enT_1^2}{k_1^2 2^n}\right)^{k_1/2} .$$

Plugging this into (52), we get that

$$\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1] \leqslant \left(\frac{16e^2T_1^3}{k_1^2 2^{2n}}\right)^{k_1/2} + \left(\frac{4enT_1^2}{k_1^2 2^n}\right)^{k_1/2} + \frac{1}{2^n} . \tag{53}$$

Upper bounding $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_2]$. We next upper bound $\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2]$. We have that for any m,

$$\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2] \leqslant \Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}] + \Pr[m+1\text{-}\mathsf{col}] . \tag{54}$$

We upper bound $\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}]$ using a compression argument. Before giving the compression argument, we recall the notion of meeting queries we introduced in the proof earlier: we say that a query $q = h(y, M) = z$ $(y \neq a)$ made by $\mathcal{A}_1$ is a meeting query for a salt a if:

- $y \neq a, y \neq z, a \neq z$
- there is a query $q' = h(a, M) = z$ made by $\mathcal{A}_1$

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $m+1$-col happens or $\mathsf{off\text{-}2\text{-}tri\text{-}}k_2$ does not happen, it outputs $\varnothing$
3. It finds a set of salts S of size $k_2/2$ such that for each $a \in S$
   - $\mathcal{A}_1$ found a category 2 triangle for a
   - The salt a has no more than $2mT_1/k_2$ meeting queries
4. If no such S is found, return $\varnothing$
5. It marks the q_1, q_2 queries corresponding to the triangle for each salt in S
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_1 query for a salt $a \in S$, then it adds the index of this query in T, and adds the lexicographical index of q_2 among all *successor* queries for the salt a in the list L_2
   (b) Otherwise it adds $h(a, M)$ to L_1.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_1 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T, it removes the element i on the front of list L_2, and locates the query q which has lexicographic order i among all the successor queries of the salt a. It answers with the answer of q
   (b) Otherwise it removes the element in front of L_1 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_1

Correctness of decoding: First off, we argue that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Note that the encoding algorithm reaches this line only if the event $\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}$ was caused by running $\mathcal{A}_1$, which means there is a set of k_2 salts S' such that $\mathcal{A}_1$ found category two triangles for them. Now, since $\mathcal{A}_1$ makes at most T_1 queries, and $\neg(m+1)\text{-}\mathsf{col}$ happens, there are at most mT_1 salt-meeting query pairs. Because otherwise there exists a query which is a meeting query to at least $m+1$ different salts- which means $(m+1)\text{-}\mathsf{col}$ has happened. Therefore, for at least $k_2/2$ salts in S' such that it has at most $2T_1m/k_2$ meeting queries (because otherwise there would be more than mT_1 salt, meeting query pairs). Therefore, such a set $S \subseteq S'$ of size $k_2/2$ always exists.

If the encoding algorithm does not return $\varnothing$, it is easy to see the decoding algorithm decodes correctly, because the query answers that the encoding algorithm does not add in the list L_1 can be recovered by the decoding algorithm correctly using S, L_2. Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}] \ .$$

Observe that the encoding algorithm removes $k_2/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were q_1 queries for different salts. It instead adds an unordered set T of $k_2/2$ values in $[T_1]$, and an ordered list of $k_2/2$ elements where each value is at most $2mT_1/k_2$. Using the compression lemma, we have that

$$\Pr[\text{ Decoding is correct }] \leqslant \frac{(2mT_1/k_2)^{k_2/2}\binom{T_1}{k_2/2}}{2^{nk_2/2}} \leqslant \left(\frac{4emT_1^2}{k_2^2 2^n}\right)^{k_2/2} \ .$$

Therefore,

$$\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{4emT_1^2}{k_2^2 2^n}\right)^{k_2/2} \ .$$

We know from previous analyses that $\Pr[(m+1)\text{-}\mathsf{col}] \leqslant 1/2^n$ for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$. Plugging this value of m we have

$$\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{16e^2T_1^3}{k_2^2 2^{2n}}\right)^{k_2/2} + \left(\frac{4enT_1^2}{k_2^2 2^n}\right)^{k_2/2} \ .$$

Plugging this into (54), we get that

$$\Pr[\mathsf{off\text{-}2\text{-}tri\text{-}}k_2] \leqslant \left(\frac{16e^2T_1^3}{k_2^2 2^{2n}}\right)^{k_2/2} + \left(\frac{4enT_1^2}{k_2^2 2^n}\right)^{k_2/2} + \frac{1}{2^n} \ . \tag{55}$$

Upper bounding $\Pr[\mathsf{off\text{-}3\text{-}tri\text{-}}k_3]$. Finally, we upper bound $\Pr[\mathsf{off\text{-}3\text{-}tri\text{-}}k_3]$. One could hope for an analysis similar to what we did for $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_1]$ and $\Pr[\mathsf{off\text{-}1\text{-}tri\text{-}}k_2]$– with the difference that in the compression argument, the encoding algorithm omits the answer of the q_2 queries and instead stores information like the

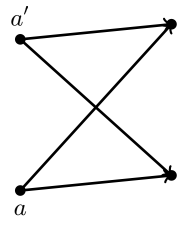


Fig. 12: The zigzag structure for a salt pair (a, a') in the query graph.

set S, L_2 to recover its answers. However, this approach does not work! The main reason for this is if the adversary finds triangles of category three for k_3 different salts, it is not necessary that the q_2 queries for the triangle are unique for each of the salts.

However, notice that if $\mathcal{A}_1$ finds triangles of category three for two salts a, a' that share the q_2 query, the following must have happened: It must have made queries $h(a, M) = y$, $h(a, M') = y'$, $h(a', M'') = y$, $h(a', M''') = y'$ for some y, y', M, M', M'', M''' such that $y \neq a, y' \neq a, y \neq y'$. If this happens, we say $\mathcal{A}_1$ has found a *zig-zag* for the salt pair (a, a'). We illustrate the structure of a zigzag in the query graph in Fig. 12. We define the event $\mathsf{off\text{-}zz}\text{-}k$ as $\mathcal{A}_1$ finding a *zig-zag* for k different salt pairs $\{\{a_{1,i}, a_{2,i}\}_{i \in k}\}$ where all the salts $a_{j,i}$'s are distinct. We have that for any k_4

$$\Pr\left[\mathsf{off\text{-}3\text{-}tri}\text{-}k_3\right] \leqslant \Pr\left[\mathsf{off\text{-}3\text{-}tri}\text{-}k_3 \wedge \neg\mathsf{off\text{-}zz}\text{-}k_4\right] + \Pr\left[\mathsf{off\text{-}zz}\text{-}k_4\right] \ . \tag{56}$$

We prove the following lemmas.

Lemma 12. *For $k \in \mathbb{N}_{>0}$,*

$$\Pr\left[\mathsf{off\text{-}zz}\text{-}k\right] \leqslant \left(\frac{e^3 T_1^4}{k^3 2^{2n}}\right)^k \ .$$

Lemma 13. *For any $k, k' \in \mathbb{N}_{>0}$ such that $k \geqslant 2k'$, k is a multiple of 2,*

$$\Pr\left[\mathsf{off\text{-}3\text{-}tri}\text{-}k \wedge \neg\mathsf{off\text{-}zz}\text{-}k'\right] \leqslant \left(\frac{4eT_1^2}{(k-2k')^2 2^n}\right)^{(k-2k')/2} \ .$$

Since k_3 is a multiple of 6, $k_3/3$ is an integer. By setting $k_4 = k_3/3$, putting together (57) with Lemmas 12 and 13, we have

$$\Pr\left[\mathsf{off\text{-}3\text{-}tri}\text{-}k_3\right] \leqslant \left(\frac{12eT_1^2}{k_3^2 2^n}\right)^{k_3/6} + \left(\frac{27e^3T_1^4}{k_3^3 2^{2n}}\right)^{k_3/3} \ . \tag{57}$$

□

We need to prove the Lemmas 12 and 13. We first prove Lemma 12.

Proof (Lemma 12). We upper bound $\Pr\left[\mathsf{off\text{-}zz}\text{-}k\right]$ via a compression argument.

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $\mathsf{off\text{-}zz}\text{-}k$ does not happen, it outputs $\varnothing$
3. It finds a set of k salt pairs $\{\{a_{i,1}, a_{i,2}\}_{i \in [k]}\}$ such that all the $a_{i,j}$'s are distinct and $\mathcal{A}_1$ found a zig-zag for each of these pairs.
4. For each of the salt pairs $a_{i,1}, a_{i,2}$, it isolates the four queries $h(a_{i,1}, M) = y$, $h(a_{i,1}, M') = y'$, $h(a_{i,2}, M'') = y$, $h(a_{i,2}, M''') = y'$. It labels the query that was the last among these four as q_4. It labels the query which has the same answer as q_4 but came earlier as q_3. It labels the earlier of the remaining two queries q_1, and later as q_2.

5. It initializes lists $\mathsf{L}_1, \mathsf{L}_2$ to empty lists, $\mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4$ to empty sets
6. It starts running $\mathcal{A}_1$ again
7. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_2 query for some zigzag salt pair, it adds the index of the query to the set T_2, inserts the index of the corresponding q_1 in front of the list L_1
   (b) If the query is marked as a q_4 query for some zigzag salt pair, it adds the index of the query to the set T_4, adds the index of the corresponding q_3 in the set T_3
   (c) Otherwise it adds $h(a, M)$ to L_2.
8. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_2 in the lexicographical order of the inputs.
9. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T_2, it removes the element i on the front of list L_1, and locates the query q which has index i among. It answers with the answer of q.
   (b) Otherwise if the index of the query is in T_4, it locates the query q that has its index in T_2 or L_1 that had input salt a. If such q is not found or more than one such q is found it aborts.
      – If q is in T_2, it finds the query q' in L_1 which was located to answer q – it then finds a query q'' in the set T_3 which has the same input salt as q'. If none or more than one q' is found, it aborts. Otherwise it answers with the answer of q'
      – If q is in L_1, it finds the query q' in T_2 which located q when being answered earlier in the decoding – it then finds a query q'' in the set T_3 which has the same input salt as q'. If none or more than one q' is found, it aborts. Otherwise it answers with the answer of q'
   (c) Otherwise it removes the element in front of L_2 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_1

Correctness of decoding: First off, observe that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}zz\text{-}}k$ to happen, the encoding algorithm will never return $\varnothing$. Further, we argue that whenever the decoding algorithm receives an output of the encoding algorithm that is not $\varnothing$, the decoding algorithm will decode correctly. This is because the encoding algorithm removes the answer of two of the queries from the pair of zig-zag salts which the decoding algorithm recovers using $\mathsf{L}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4$. It is easy to see decoding answers the queries whose answers were in L_1 and the ones whose answers were not in L_1 but indices were put in T_2 correct. What is left is verifying that the queries whose answers were not in L_1 and whose indices were put in T_4 were answered correctly.

This can be verified by inspecting line 3b. For an output of the encoding algorithm that is not $\varnothing$, the decoding algorithm will never abort here – this is because for the q_4 queries removed, there is exactly one other query among all the queries whose indices are in $\mathsf{L}_1, \mathsf{T}_2$ whose salt is same as q_4, by the definition of the event $\mathsf{off\text{-}zz\text{-}}k$. And this step successfully recovers that value and returns the correct answer. Therefore, whenever $\mathcal{A}_1$ causes $\mathsf{off\text{-}zz\text{-}}k$, the encoding produces an outputs that decodes correctly.

Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}zz\text{-}}k] \ .$$

Observe that the encoding algorithm removes $2 \cdot k$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were queries for different salts. It instead adds an unordered sets $\mathsf{T}_2, \mathsf{T}_2, \mathsf{T}_4$ of k values in $[T_1]$, and an ordered list of k values in $[T_1]$. Using the compression lemma, we have that

$$\Pr[\mathsf{off\text{-}zz\text{-}}k] \leqslant \Pr[\text{ Decoding is correct }] \leqslant \frac{\left(\binom{T_1}{k}\right)^3 T_1^k}{2^{n2k}} \leqslant \left(\frac{e^3 T_1^4}{k^3 2^{2n}}\right)^k \ .$$

□

We next prove Lemma 13.

Proof (Lemma 12). We need to upper bound $\Pr\left[\mathsf{off\text{-}3\text{-}tri\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k'\right]$. Note that if $\mathcal{A}_1$ does not cause $\mathsf{off\text{-}zz\text{-}}k'$ and finds triangles of category three for at least k salts, it means that there is a set of at least $k-2k'$ salts for which $\mathcal{A}_1$ finds triangles of category three such that no two triangles in the set share any query. We use this to intuition to build a compression argument that upper bounds $\Pr\left[\mathsf{off\text{-}3\text{-}tri\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k'\right]$.

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $\mathsf{off\text{-}zz\text{-}}k'$ happens or $\mathsf{off\text{-}3\text{-}tri\text{-}}k$ does not happen, it outputs $\varnothing$
3. It finds a set of salts S of size $(k-2k')/2$ such that for each $a \in S$
   - $\mathcal{A}_1$ found a category 3 triangle for a
   - The salt a has no more than $2T_1/(k-2k')$ queries on it
4. If no such S is found, return $\varnothing$
5. It marks the q_3, q_2 queries corresponding to the triangle for each salt in S
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_2 query for a salt $a \in S$, then it adds the index of this query in T, and it adds the index of q_3 in the list L_2
   (b) Otherwise it adds $h(a, M)$ to L_1.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_1 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{T}$.

The decoding procedure works as follows.

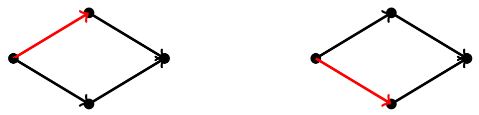
1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T, it removes the element i on the front of list L_2, and locates the query q which has index i. It answers with the answer of q
   (b) Otherwise it removes the element in front of L_1 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_1

Correctness of decoding: First off, we argue that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}3\text{-}tri\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k'$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Note that the encoding algorithm reaches this line only if the event $\mathsf{off\text{-}2\text{-}tri\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k'$ was caused by running $\mathcal{A}_1$, which means there is a set of k salts S' such that $\mathcal{A}_1$ found category three triangles for them. Moreover there does not exist a set of k' salt pairs for which $\mathcal{A}_1$ found a zig-zag. Meaning one can find a set of at least $k-2k'$ salts such that $\mathcal{A}_1$ found a category three triangle for all of them but no two of these triangles share a q_2 query. Now, since $\mathcal{A}_1$ makes at most T_1 queries, at least half of these salts are such that the adversary makes at most $2Tp/(k-2k')$ queries on them. Therefore, such a set S of size $(k-2k')/2$ always exists.

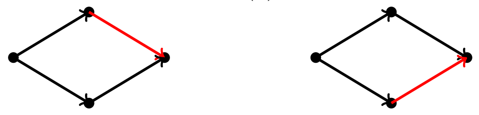
If the encoding algorithm does not return $\varnothing$, it is easy to see the decoding algorithm decodes correctly, because the query answers that the encoding algorithm does not add in the list L_1 can be recovered by the decoding algorithm correctly using S, L_2. Therefore, we have that

$$\Pr\left[\text{ Decoding is correct }\right] \geqslant \Pr\left[\mathsf{off\text{-}3\text{-}tri\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k'\right] .$$

Observe that the encoding algorithm removes $(k-2k')/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were q_1 queries for different salts. It instead adds an



(a) Category 1 diamonds.



(b) Category 2 diamonds.

Fig. 13: The different categories of diamonds. The edge colored red is the last of the four queries made.

unordered set T of $(k-2k')/2$ values in $[T_1]$, and an ordered list of $(k-2k')/2$ values in $[T_1]$. Using the compression lemma, we have that

$$\Pr[\text{ Decoding is correct }] \leqslant \frac{(2T_1/(k-2k'))^{(k-2k')/2}\binom{T_1}{(k-2k')/2}}{2^{n(k-2k')/2}} \leqslant \left(\frac{4eT_1^2}{(k-2k')^2 2^n}\right)^{(k-2k')/2} .$$

Therefore,

$$\Pr\left[\textsf{off-3-tri-}k \wedge \neg\textsf{off-zz-}k'\right] \leqslant \left(\frac{4eT_1^2}{(k-2k')^2 2^n}\right)^{(k-2k')/2} . \tag{58}$$

□

5.10 Proof of Lemma 11

Proof (Lemma 11). We say that $\mathcal{A}_1$ has found a triangle for a salt a if it has made queries $h(a,M)=y$, $h(y,M')=z$, $h(a,M'')=z$ for some M,M',M'',y,z. We define the event $\textsf{off-dia-}k$ as $\mathcal{A}_1$ finding diamonds for at least k salts. We have that for any k,

$$\Pr\left[\textsf{dia}_{\textsf{off}}\right] \leqslant \Pr\left[\textsf{dia}_{\textsf{off}} \,\middle|\, \neg\textsf{off-dia-}k\right] + \Pr\left[\textsf{off-dia-}k\right] . \tag{59}$$

We claim that $\Pr\left[\textsf{dia}_{\textsf{off}} \,\middle|\, \neg\textsf{off-dia-}k\right] \leqslant k/2^n$ – this is because if $\mathcal{A}_1$ finds diamonds for at most k different salts, since a is sampled uniformly at random, $\textsf{dia}_{\textsf{off}}$ happens with probability at most $k/2^n$.

Towards upper bounding $\textsf{off-dia-}k$, we define a categorization of the diamonds. For a diamond on salt a, let the four queries forming the triangle be $q_1 = h(a,M) = y$, $q_2 = h(y,M') = z$, $q_3 = h(a,M'') = y'$, $q_4 = h(y',M''') = z$.

- We say that a diamond is of category 1, if the query q_1 or q_3 was the last among the three
- We say that a triangle is of category 2, if the query q_2 or q_4 was the last among the three

We illustrate the categories of diamonds in Fig. 13. We define the event $\textsf{off-1-dia-}k_1$ as $\mathcal{A}_1$ find diamonds of category 1 for k_1 different salts. We define the event $\textsf{off-2-dia-}k_2$ as $\mathcal{A}_1$ finds diamonds of category 2 for k_2 different salts.

We would have that for any k_1, k_2,

$$\Pr\left[\textsf{off-dia-}2\cdot\max(k_1,k_2)\right] \leqslant \Pr\left[\textsf{off-1-dia-}k_1\right] + \Pr\left[\textsf{off-2-dia-}k_2\right] .$$

We prove the three following claims.

Claim. For any $k_1 \in \mathbb{N}_{>0}$, such that k_1 is a multiple of 2

$$\Pr\left[\textsf{off-1-dia-}k_1\right] \leqslant \left(\frac{64e^3T_1^4}{k_1^2 2^{3n}}\right)^{k_1/2} + \left(\frac{4en^2T_1^2}{k_1^2 2^n}\right)^{k_1/2} + \frac{1}{2^n} .$$

Claim. For any $k_2 \in \mathbb{N}_{>0}$, such that k_2 is a multiple of 30

$$\Pr[\mathsf{off\text{-}2\text{-}dia\text{-}}k_2] \leqslant \left(\frac{1600eT_1^4}{(k_2)^2 2^{3n}}\right)^{k_2/10} + \left(\frac{100en^2T_1^2}{(k_2)^2 2^n}\right)^{k_2/10} + 3\left(\frac{32e(60e)^3T_1^6}{k_2^3 2^{4n}}\right)^{k_2/30} + 3\left(\frac{256\cdot 5^3\cdot enT_1^3}{k_2^3 2^n}\right)^{k_2/30} + \frac{4}{2^n}\ .$$

We let $k_1 = \max\left(\left(\frac{128e^3T_1^4}{2^{3n}}\right)^{1/2}, \left(\frac{8en^2T_1^2}{2^n}\right)^{1/2}, 2n\right)$. Then we have that

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1] \leqslant \frac{3}{2^n}\ .$$

We let

$$k_2 = \max\left(\left(\frac{3200eT_1^4}{2^{3n}}\right)^{1/2}, \left(\frac{200en^2T_1^2}{2^n}\right)^{1/2}, \left(\frac{64e(60e)^3T_1^6}{2^{4n}}\right)^{1/3}, \left(\frac{512\cdot 5^3\cdot enT_1^3}{2^n}\right)^{1/3}, 30n\right)\ .$$

Then we have that

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_2] \leqslant \frac{12}{2^n}\ .$$

Also,

$$\begin{aligned}\max(k_1,k_2) &\leqslant \left(\frac{128e^3T_1^4}{2^{3n}}\right)^{1/2} + \left(\frac{8en^2T_1^2}{2^n}\right)^{1/2} + \left(\frac{3200eT_1^4}{2^{3n}}\right)^{1/2} + \left(\frac{200en^2T_1^2}{2^n}\right)^{1/2} \\ &\quad + \left(\frac{64e(60e)^3T_1^6}{2^{4n}}\right)^{1/3} + \left(\frac{512\cdot 5^3\cdot enT_1^3}{2^n}\right)^{1/3} + 30n \\ &= \left(\frac{8(2)^{1/2}e^{3/2}T_1^2}{2^{3n/2}}\right) + \left(\frac{(8e)^{1/2}nT_1}{2^{n/2}}\right) + \left(\frac{40(2e)^{1/2}T_1^2}{2^{3n/2}}\right) + \left(\frac{10(2e)^{1/2}nT_1}{2^{n/2}}\right) \\ &\quad + \left(\frac{240e^{2/3}T_1^2}{2^{4n/3}}\right) + \left(\frac{40(en)^{1/3}T_1}{2^{n/3}}\right) + 30n\ .\end{aligned}$$

Let $k = 2\max(k_1,k_2)$. We have that

$$\Pr[\mathsf{off\text{-}dia\text{-}}k] \leqslant \frac{15}{2^n}\ .$$

Plugging this into (59) we have that

$$\begin{aligned}\Pr[\mathsf{dia_{off}}] &\leqslant \left(\frac{8(2)^{1/2}e^{3/2}T_1^2}{2^{5n/2}}\right) + \left(\frac{(8e)^{1/2}nT_1}{2^{3n/2}}\right) + \left(\frac{40(2e)^{1/2}T_1^2}{2^{5n/2}}\right) + \left(\frac{10(2e)^{1/2}nT_1}{2^{3n/2}}\right) \\ &\quad + \left(\frac{240e^{2/3}T_1^2}{2^{7n/3}}\right) + \left(\frac{40(en)^{1/3}T_1}{2^{4n/3}}\right) + \frac{30n}{2^n} + \frac{15}{2^n}\ . \\ &= \frac{30n+15}{2^n} + \frac{40(en)^{1/3}T_1}{2^{4n/3}} + \frac{(8e)^{1/2}nT_1 + 10(2e)^{1/2}nT_1}{2^{3n/2}} \\ &\quad + \frac{240e^{2/3}T_1^2}{2^{7n/3}} + \frac{8(2)^{1/2}e^{3/2}T_1^2 + 40(2e)^{1/2}T_1^2}{2^{5n/2}}\ .\end{aligned}$$

We now prove these claims one by one.

Upper bounding $\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1]$. We have that for any m,

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1] \leqslant \Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] + \Pr[m+1\text{-}\mathsf{col}] \ . \tag{60}$$

We upper bound $\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}]$ using a compression argument. Before giving the compression argument, we recall the notion of successor-hitting queries we introduced earlier in the proof: we say that a query $q = h(y, M) = z$ made by $\mathcal{A}_1$ is a successor-hitting query for a salt a if

- $z \neq a, y \neq a, y \neq z$
- there are queries $q' = h(a, M') = b$, $q'' = h(b, M'') = z$ with $b \neq a, z \neq b$ made by $\mathcal{A}_1$, for some M', M''

Informally, a query q is a successor-hitting query to salt a if there query $q' = h(y', M') = z$ such that q' is a successor query to a and $q' \neq q$.

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $m+1$-col happens or $\mathsf{off\text{-}1\text{-}dia\text{-}}k_1$ does not happen, it outputs $\varnothing$
3. It finds a set of salts S of size $k_1/2$ such that for each $a \in S$
   - $\mathcal{A}_1$ found a category 1 triangle for a
   - The salt a has no more than $2m^2T_1/k_1$ successor-hitting queries
4. If no such S is found, return $\varnothing$
5. It marks the q_2, q_3 queries corresponding to the triangle for each salt in S
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_3 query for a salt $a \in S$, then it adds the index of this query in T, and it adds the lexicographical index of q_2 among all *successor-hitting* queries for the salt a in the list L_2
   (b) Otherwise it adds $h(a, M)$ to L_1.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_1 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T, it removes the element i on the front of list L_2, and locates the query q which has lexicographic order i among all the successor-hitting queries of the salt a. It answers with the answer of q
   (b) Otherwise it removes the element in front of L_1 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_1

Correctness of decoding: First off, we argue that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Note that the encoding algorithm reaches this line only if the event $\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}$ was caused by running $\mathcal{A}_1$, which means there is a set of k_1 salts S' such that $\mathcal{A}_1$ found category one diamonds for them. Now, since $\mathcal{A}_1$ makes at most T_1 queries, and $\neg(m+1)\text{-}\mathsf{col}$ happens, there are at most m^2T_1 salt-successor-hitting query pairs. This is because if there were more than m^2T_1 salt-successor hitting query pairs, there would be at least one query which is a successor hitting query to m^2+1 salts. Given that there are no $m+1$-multi-collisions, this would be contradiction.

Therefore, for at least $k_1/2$ salts (note that since k_1 is a multiple of 2, $k_1/2$ is an integer) in S' such that it has at most $2T_1m^2/k_1$ successor queries (because otherwise there would be more than m^2T_1 salt, successor-hitting query pairs). Therefore, such a set $S \subseteq S'$ of size $k_1/2$ always exists.

If the encoding algorithm does not return $\varnothing$, it is easy to see the decoding algorithm decodes correctly, because the query answers that the encoding algorithm does not add in the list L_1 can be recovered by the decoding algorithm correctly using S, L_2. Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \;.$$

Observe that the encoding algorithm removes $k_1/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were q_3 queries for different salts. It instead adds an unordered set T of $k_1/2$ values in $[T_1]$, and an ordered list of $k_1/2$ elements where each value is at most $2m^2T_1/k_1$. Using the compression lemma, we have that

$$\Pr[\text{ Decoding is correct }] \leqslant \frac{(2m^2T_1/k_1)^{k_1/2}\binom{T_1}{k_1/2}}{2^{nk_1/2}} \leqslant \left(\frac{4em^2T_1^2}{k_1^2 2^n}\right)^{k_1/2} \;.$$

Therefore,

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{4em^2T_1^2}{k_1^2 2^n}\right)^{k_1/2} \;.$$

We know from previous analyses that $\Pr[(m+1)\text{-}\mathsf{col}] \leqslant 1/2^n$ for $m = \max\left(n, \frac{4eT_1}{2^n}\right)$. Plugging this value of m we have

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1 \wedge \neg(m+1)\text{-}\mathsf{col}] \leqslant \left(\frac{64e^3T_1^4}{k_1^2 2^{3n}}\right)^{k_1/2} + \left(\frac{4en^2T_1^2}{k_1^2 2^n}\right)^{k_1/2} \;.$$

Plugging this into (60), we get that

$$\Pr[\mathsf{off\text{-}1\text{-}dia\text{-}}k_1] \leqslant \left(\frac{64e^3T_1^4}{k_1^2 2^{3n}}\right)^{k_1/2} + \left(\frac{4en^2T_1^2}{k_1^2 2^n}\right)^{k_1/2} + \frac{1}{2^n} \;. \tag{61}$$

We next upper bound $\Pr[\mathsf{off\text{-}2\text{-}dia\text{-}}k_2]$. One could hope for an analysis similar to what we did for $\Pr[\mathsf{off\text{-}dia\text{-}}k_1]$– with the difference that in the compression argument, the encoding algorithm omits the answer of the last query and instead stores information like the set S, L_2 to recover its answers. However, this approach does not work! The main reason for this is if the adversary finds diamonds of category two for k_2 different salts, it is not necessary that the last queries for the diamond are unique for each of the salts. Therefore, we need to separately handle the case where a particular query is the last query for diamonds of category two of multiple salts.

Now, consider how two salts with category two diamonds can share the last query. There are two possibilities here: either the diamonds share both the queries q_2, q_4 queries or they share one of the queries q_2, q_4. If they share both the queries q_2, q_4, we have that the pair of salts form a zig-zag, and we can re-use our analysis from Lemma 8. The other possibility is both the category two diamonds share one query that was the last query for both of them. For this to happen for salts a, a', before the shared query is made, there must have been queries

- $q_1 := h(a, M_1) = w$ for some w, M_1 such that $w \neq a, a'$
- $q_2 := h(a, M_2) = y$ for some y, M_2 such that $y \neq w, a, a'$
- $q_3 := h(y, M_3) = z$ for some z, M_3 such that $z \neq a, a', w, y$
- $q_4 := h(a', M_4) = w$ for some M_4
- $q_5 := h(a', M_5) = y'$ for some y', M_5 such that $y' \neq w, a, a', y, z$
- $q_6 := h(y', M_6) = z$ for some M_6

We refer to such a structure as a *hexagon* because this structure consists of a total of six queries and two of these six collide. If for some salt pair $\{a, a'\}$, $\mathcal{A}_1$ has made the above queries, we say that it has found a hexagon for the salt pair. We illustrate the structure of a hexagon in a query graph in Fig. 14.

We define the event $\mathsf{off\text{-}hex\text{-}}k$ as follows: there is a set of k salt pairs $\{\{a_{i,1}, a_{i,2}\}_{i\in[k]}\}$ such that

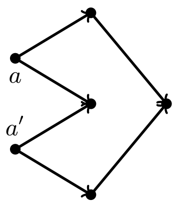


Fig. 14: The hexagon structure for a salt pair (a', a') in the query graph.

1. $| \cup_{i\in[k]} \{a_{i,1}, a_{i,2}\}| = 2k$
2. $\mathcal{A}_1$ has found a hexagon for each pair $\{a_{i,1}, a_{i,2}\}$ for all $i \in [k]$.

We have that for all k_3, k_4, m

$$\Pr[\textsf{off-2-dia-}k_2] \leqslant \Pr[\textsf{off-2-dia-}k_2 \wedge \neg\textsf{off-zz-}k_3 \wedge \neg\textsf{off-hex-}k_4 \wedge \neg(m+1)\textsf{-col}] \tag{62}$$
$$+ \Pr[\textsf{off-zz-}k_3] + \Pr[\textsf{off-hex-}k_4] + \Pr[m+1\textsf{-col}] \ . \tag{63}$$

We prove the following lemmas.

Lemma 14. *For $k \in \mathbb{N}_{>0}$ such that k is a multiple of 6*

$$\Pr[\textsf{off-hex-}k] \leqslant 3\left(\frac{32e(12e)^3T_1^6}{k^3 2^{4n}}\right)^{k/6} + 3\left(\frac{256enT_1^3}{k^3 2^n}\right)^{k/6} + \frac{3}{2^n} \ . \tag{64}$$

Lemma 15. *For any $k, k', k'', m \in \mathbb{N}_{>0}$ such that k is a multiple of 2, $k > 2k' + 2k''$,*

$$\Pr\left[\textsf{off-2-dia-}k \wedge \neg\textsf{off-zz-}k' \wedge \neg\textsf{off-hex-}k'' \wedge \neg(m+1)\textsf{-col}\right]$$
$$\leqslant \left(\frac{4em^2T_1^2}{(k-2k'-2k'')^2 2^n}\right)^{(k-2k'-2k'')/2} .$$

From Lemma 12 we had that for $k \in \mathbb{N}_{>0}$,

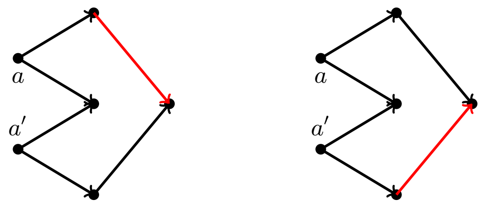
$$\Pr[\textsf{off-zz-}k] \leqslant \left(\frac{e^3T_1^4}{k^3 2^{2n}}\right)^k .$$

Plugging this into (62), we have that for any k_2, k_3, k_4, m such that k_4 is a multiple of 6, k_2 is a multiple of 2, $k > 2k_3 + k_4$

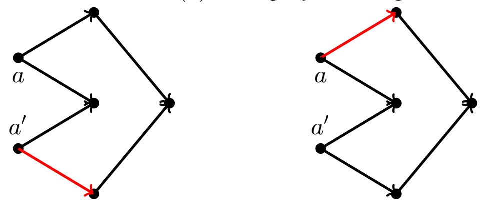
$$\Pr[\textsf{off-2-dia-}k_2] \leqslant \left(\frac{4em^2T_1^2}{(k_2-2k_3-2k_4)^2 2^n}\right)^{(k_2-2k_3-2k_4)/2}$$
$$+ 3\left(\frac{32e(12e)^3T_1^6}{k_4^3 2^{4n}}\right)^{k_4/6} + 3\left(\frac{256enT_1^3}{k_4^3 2^n}\right)^{k_4/6}$$
$$+ \frac{3}{2^n} + \Pr[(m+1)\textsf{-col}] \ .$$

Setting $k_4 = k_3 = k_2/5$, we have

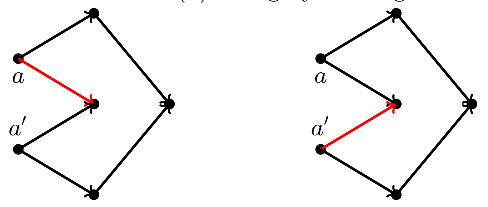
$$\Pr[\textsf{off-2-dia-}k_2] \leqslant \left(\frac{100em^2T_1^2}{(k_2)^2 2^n}\right)^{k_2/10} + 3\left(\frac{32e(60e)^3T_1^6}{k_2^3 2^{4n}}\right)^{k_2/30}$$
$$+ 3\left(\frac{256\cdot 5^3\cdot enT_1^3}{k_2^3 2^n}\right)^{k_2/30} + \frac{3}{2^n} + \Pr[(m+1)\textsf{-col}] \ .$$



(a) Category 1 hexagons.



(b) Category 2 hexagons.



(c) Category 3 hexagons.

Fig. 15: The different categories of hexagons. The edge colored red is the last of the six queries made.

Further, in the proof of Lemma 7 we had that for $m = \max(n, 4eT_1/2^n)$, $\Pr\left[(m+1)\text{-}\mathsf{col}\right] \leqslant 1/2^n$. Setting m to this value we have that for any k_2 such that k_2 is a multiple of 30,

$$\Pr\left[\mathsf{off\text{-}2\text{-}dia\text{-}}k_2\right] \leqslant \left(\frac{1600eT_1^4}{(k_2)^2 2^{3n}}\right)^{k_2/10} + \left(\frac{100en^2T_1^2}{(k_2)^2 2^n}\right)^{k_2/10}$$
$$+ 3\left(\frac{32e(60e)^3T_1^6}{k_2^3 2^{4n}}\right)^{k_2/30} + 3\left(\frac{256\cdot 5^3\cdot enT_1^3}{k_2^3 2^n}\right)^{k_2/30} + \frac{4}{2^n}\ .$$

We prove Lemma 14.

Proof (Lemma 14). For the adversary to query a hexagon for salts a, a', there must have been queries

- $q_1 := h(a, M_1) = w$ for some w, M_1 such that $w \neq a, a'$
- $q_2 := h(a, M_2) = y$ for some y, M_2 such that $y \neq w, a, a'$
- $q_3 := h(y, M_3) = z$ for some z, M_3 such that $z \neq a, a', w, y$
- $q_4 := h(a', M_4) = w$ for some M_4
- $q_5 := h(a', M_5) = y'$ for some y', M_5 such that $y' \neq w, a, a', y, z$
- $q_6 := h(y', M_6) = z$ for some M_6

We categorize hexagons into three categories based on the query that was the last one among the six

1. category 1 hexagons: q_3 or q_6 is the last query
2. category 2 hexagons: q_2 or q_5 is the last query
3. category 3 hexagons: q_1 or q_4 is the last query

We illustrate the different categories of hexagons in Figure 15.

We define these three events.

1. $\textsf{off-1-hex-}k_1$: $\mathcal{A}_1$ has found hexagons of category 1 for a set of k_1 salt pairs such that none of the two salt pairs in the set share a salt
2. $\textsf{off-2-hex-}k_2$: $\mathcal{A}_1$ has found hexagons of category 2 for a set of k_1 salt pairs such that none of the two salt pairs in the set share a salt
3. $\textsf{off-3-hex-}k_3$: $\mathcal{A}_1$ has found hexagons of category 3 for a set of k_1 salt pairs such that none of the two salt pairs in the set share a salt

Since if $\mathcal{A}_1$ finds $3k$ hexagons, it must have found k hexagons of one of the categories, we have that for any k

$$\Pr[\textsf{off-hex-}3k] \leqslant \Pr[\textsf{off-1-hex-}k] + \Pr[\textsf{off-2-hex-}k] + \Pr[\textsf{off-3-hex-}k] \ .$$

We prove the three following claims.

Claim. For any $k \in \mathbb{N}_{>0}$ such that k is a multiple of 2

$$\Pr[\textsf{off-1-hex-}k] \leqslant \left(\frac{32e(4e)^3T_1^6}{k^3 2^{4n}}\right)^{k/2} + \left(\frac{32enT_1^3}{k^3 2^n}\right)^{k/2} + \frac{1}{2^n} \ .$$

Claim. For any $k \in \mathbb{N}_{>0}$ such that k is a multiple of 2

$$\Pr[\textsf{off-2-hex-}k] \leqslant \left(\frac{32e(4e)^3T_1^6}{k^3 2^{4n}}\right)^{k/2} + \left(\frac{32enT_1^3}{k^3 2^n}\right)^{k/2} + \frac{1}{2^n} \ .$$

Claim. For any $k \in \mathbb{N}_{>0}$ such that k is a multiple of 2

$$\Pr[\textsf{off-3-hex-}k] \leqslant \left(\frac{32e(4e)^3T_1^6}{k^3 2^{4n}}\right)^{k/2} + \left(\frac{32enT_1^3}{k^3 2^n}\right)^{k/2} + \frac{1}{2^n} \ .$$

Putting this all together we have for any k that is a multiple of 2

$$\Pr[\textsf{off-hex-}3k] \leqslant 3\left(\frac{32e(4e)^3T_1^6}{k^3 2^{4n}}\right)^{k/2} + 3\left(\frac{32enT_1^3}{k^3 2^n}\right)^{k/2} + \frac{3}{2^n} \ .$$

Equivalently, we have that for any k that is a multiple of 6

$$\Pr[\textsf{off-hex-}k] \leqslant 3\left(\frac{32e(12e)^3T_1^6}{k^3 2^{4n}}\right)^{k/6} + 3\left(\frac{256enT_1^3}{k^3 2^n}\right)^{k/6} + \frac{3}{2^n} \ .$$

Upper bounding $\Pr[\textsf{off-1-hex-}k]$. We have that for any m,

$$\Pr[\textsf{off-1-hex-}k] \leqslant \Pr[\textsf{off-1-hex-}k \wedge \neg(m+1)\textsf{-col}] + \Pr[(m+1)\textsf{-col}] \ . \tag{65}$$

We upper bound $\Pr[\textsf{off-1-hex-}k \wedge \neg(m+1)\textsf{-col}]$ using a compression argument. The encoding procedure encodes the random oracle h as follows.

1. It runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $m+1\textsf{-col}$ happens or $\textsf{off-1-hex-}k$ does not happen, it outputs $\varnothing$
3. it finds a set of salt pairs S of size $k/8$ such that for each $\{a, a'\} \in S$
   (a) $\mathcal{A}_1$ found a hexagon for $\{a', a''\}$
   (b) $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a'
   (c) $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a''
   (d) The salt a has no more than $4mT_1/k$ successor queries
   (e) The salt a has no more than $4mT_1/k$ successor queries
4. If no such S is found, it returns $\varnothing$

5. For every $\{a', a''\}$ pair in S it picks one type 1 hexagon formed by the queries of $\mathcal{A}_1$ (if there is more than one, it picks one arbitrarily). It lets the queries forming the hexagon be $q_1 = h(a', M_1) = w, q_2 = h(a, M_2) = y, q_3 = h(y, M_3) = z, q_4 = h(a'', M_4) = w, q_5 = h(a', M_5) = y', q_6 = h(y', M_6) = z$ for some $M_1, M_2, M_3, M_4, M_5, M_6, w, y, y', z$ such that $|\{a', a'', w, y, y', z\}| = 6$ such that q_6 was the last of the six queries made by $\mathcal{A}_1$.
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_6 query for a salt pair $\{a, a'\} \in S$
      i. it adds the index of this query in T
      ii. it adds the lexicographical index of the corresponding q_5 query among all queries with answer a to the list L_1
      iii. Let the input salt of the corresponding q_5 query be a''. It adds the lexicographical index of the corresponding q_4 query among all queries with input salt a'' to the list L_2
      iv. Let the answer of the corresponding q_4 query be w. It adds the lexicographical index of the corresponding q_1 query among all queries with answer w to the list L_3
      v. Let the input salt of the corresponding q_1 query be a'. It adds the lexicographical index of the corresponding q_3 query among all queries which are successor queries of the salt a' to the list L_4.
   (b) Otherwise it adds $h(a, M)$ to L_5.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_5 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T,
      i. it removes the element i on the front of list L_1, and locates the query q_5 which has lexicographic order i among all queries with answer a.
      ii. Let the input of q_5 be a''. It removes the element i on the front of list L_2, and locates the query q_4 which has lexicographic order i among all queries with input salt a''.
      iii. Let the answer of q_4 be w. It removes the element i on the front of list L_3, and locates the query q_1 which has lexicographic order i among all queries with answer w.
      iv. Let the input salt of q_1 be a'. It removes the element i on the front of list L_4, and locates the query q_3 which has lexicographic order i among all successor queries of the salt a'

      It answers with the answer of query q_3.
   (b) Otherwise it removes the element in front of L_5 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_5

Correctness of decoding: First off, we argue that for an adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}1\text{-}hex\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Since $\mathsf{off\text{-}1\text{-}hex\text{-}}k$ happens, there is a set S' of k salt pair such that no two pairs share a salt and $\mathcal{A}_1$ has found a hexagon for each pair. We first claim that at most $k/4$ of the pairs in S' are such that $\mathcal{A}_1$ has made more than $4T_1/k$ queries on one or both the salts in the pair- this is because otherwise there would need to be more than T_1 queries which is a contradiction. Further, since $\neg(m+1)\text{-}\mathsf{col}$ happens, we have that there are at most mT_1 salt-successor query pairs – this was argued in the proof of 8. We next claim that at most $k/4$ of the pairs in S' are such that for one or both salts in the pair- there are more than $4mT_1/k$ successor queries – this is because otherwise there would be more than mT_1 salt-successor query pairs.

Therefore, we have that

– The subset of S' such that at least one of the salts is such that there are more than $4T_1/k$ queries on the salt is of size at most $k/4$
– The subset of S' such that at least one of the salts is such that there are more than $4mT_1/k$ successor queries for the salt is of size at most $k/4$

Therefore, this implies the subset of S' such that for at least one of the salts **a)** there are more $4T_1/k$ queries on the salt or, **b)** there are more than $4mT_1/k$ queries that are successor query to the salt is of size at most $k/2$. This in turn means the subset of S' such that for both of the salts **a)** there are at most $4T_1/k$ queries on the salt and, **b)** there are at most $4mT_1/k$ queries that are successor query to the salt is of size at least $k/2$. Therefore, the encoding algorithm will always find such a set S, and never produce $\varnothing$ in line 4.

Further, we argue that whenever the decoding algorithm receives an output of the encoding algorithm that is not $\varnothing$, the decoding algorithm will decode correctly. It is easy to see decoding answers the queries whose answers were in L_5. What is left is verifying that the queries whose answers were not in L_5 and whose indices were put in T were answered correctly.

This can be verified by inspecting line 8a in the encoding algorithm and line 3a in the decoding algorithm. It can be checked that the decoding algorithm always returns the same answer as the encoding algorithm. Therefore, whenever $\mathcal{A}_1$ causes $\mathsf{off\text{-}1\text{-}hex\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}$, the encoding produces an outputs that decodes correctly.

Therefore, we have that

$$\Pr[\text{ Decoding is correct }] \geqslant \Pr[\mathsf{off\text{-}1\text{-}hex\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}] \ .$$

Observe that the encoding algorithm removes $k/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were queries for different salts. It instead adds

1. T: a set of $k/2$ distinct values in $[T_1]$
2. L_1: a sequence of $k/2$ values that are each at most m
3. L_2: a sequence of $k/2$ values that are each at most $4T_1/k$
4. L_3: a sequence of $k/2$ values that are each at most m
5. L_4: a sequence of $k/2$ values that are each at most $4mT_1/k$

Using the compression lemma, we have that

$$\begin{aligned}\Pr[\mathsf{off\text{-}1\text{-}hex\text{-}}k \wedge \neg(m+1)\text{-}\mathsf{col}] &\leqslant \Pr[\text{ Decoding is correct }]\\ &\leqslant \frac{\binom{T_1}{k/2} m^{k/2} \left(\frac{4T_1}{k}\right)^{k/2} m^{k/2} \left(\frac{4mT_1}{k}\right)^{k/2}}{2^{nk/2}}\\ &\leqslant \left(\frac{(2eT_1)(4mT_1)(4T_1)m^2}{k^3 2^n}\right)^{k/2}\\ &= \left(\frac{32eT_1^3 m^3}{k^3 2^n}\right)^{k/2} \ .\end{aligned}$$

We know from proof of Lemma 7, that for $m = \max(4eT_1/2^n, n)$, $\Pr[(m+1)\text{-}\mathsf{col}] \leqslant 1/2^n$. We set m to this value and plug this into (65).

$$\Pr[\mathsf{off\text{-}1\text{-}hex\text{-}}k] \leqslant \left(\frac{32e(4e)^3 T_1^6}{k^3 2^{4n}}\right)^{k/2} + \left(\frac{32enT_1^3}{k^3 2^n}\right)^{k/2} + \frac{1}{2^n} \ .$$

Upper bounding $\Pr[\mathsf{off\text{-}2\text{-}hex\text{-}}k]$. The analysis for this is identical to that for $\Pr[\mathsf{off\text{-}1\text{-}hex\text{-}}k]$ with a slightly modified compression strategy (but one that gives identical amount of compression). For the sake of completeness, we just give the modified encoding and decoding algorithms here.

The encoding procedure encodes the random oracle h as follows.

1. It runs $\mathcal{A}_1^h$, answering all its queries using h.

2. If $m + 1$-col happens or off-hex-k does not happen, it outputs $\varnothing$
3. it finds a set of salt pairs S of size $k/8$ such that for each $\{a, a'\} \in S$
   - $\mathcal{A}_1$ found a hexagon for $\{a', a''\}$
   - $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a'
   - $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a''
   - The salt a has no more than $4mT_1/k$ successor queries
   - The salt a has no more than $4mT_1/k$ successor queries
4. If no such S is found, it returns $\varnothing$
5. For every $\{a', a''\}$ pair in S it picks one type 1 hexagon formed by the queries of $\mathcal{A}_1$ (if there is more than one, it picks one arbitrarily). It lets the queries forming the hexagon be $q_1 = h(a', M_1) = w, q_2 = h(a, M_2) = y, q_3 = h(y, M_3) = z, q_4 = h(a'', M_4) = w, q_5 = h(a', M_5) = y', q_6 = h(y', M_6) = z$ for some $M_1, M_2, M_3, M_4, M_5, M_6, w, y, y', z$ such that $|\{a', a'', w, y, y', z\}| = 6$ such that q_5 was the last of the six queries made by $\mathcal{A}_1$.
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_5 query for a salt pair $\{a, a'\} \in S$
      i. it adds the index of this query in T
      ii. it adds the lexicographical index of the corresponding q_5 query among all queries with input a to the list L_1
      iii. Let the answer of the corresponding q_4 query be w. It adds the lexicographical index of the corresponding q_1 query among all queries with answer w to the list L_2
      iv. Let the input salt of the corresponding q_1 query be a'. It adds the lexicographical index of the corresponding q_3 query among all successor queries of a' to the list L_3
      v. Let the answer of the corresponding q_3 query be z. It adds the lexicographical index of the corresponding q_6 query among all queries with answer z to the list L_4.
   (b) Otherwise it adds $h(a, M)$ to L_5.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_5 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T,
      i. it removes the element i on the front of list L_1, and locates the query q_4 which has lexicographic order i among all queries with input salt a.
      ii. Let the answer of q_4 be w. It removes the element i on the front of list L_2, and locates the query q_1 which has lexicographic order i among all queries with answer w.
      iii. Let the input salt of q_1 be a'. It removes the element i on the front of list L_3, and locates the query q_3 which has lexicographic order i among all successor queries of a'.
      iv. Let the answer of q_3 be z. It removes the element i on the front of list L_4, and locates the query q_6 which has lexicographic order i among all queries with answer z.

      It answers with the input of query q_6.
   (b) Otherwise, it removes the element in front of L_5 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_5

Upper bounding $\Pr[\mathsf{off\text{-}3\text{-}hex\text{-}}k]$. The analysis for this is identical to that for $\Pr[\mathsf{off\text{-}1\text{-}hex\text{-}}k]$ with a slightly modified compression strategy (but one that gives identical amount of compression). For the sake of completeness, we just give the modified encoding and decoding algorithms here.

The encoding procedure encodes the random oracle h as follows.

1. It runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $m+\mathsf{1\text{-}col}$ happens or $\mathsf{off\text{-}hex\text{-}}k$ does not happen, it outputs $\varnothing$
3. it finds a set of salt pairs S of size $k/8$ such that for each $\{a, a'\} \in S$
   - $\mathcal{A}_1$ found a hexagon for $\{a', a''\}$
   - $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a'
   - $\mathcal{A}_1$ made no more than $4T_1/k$ queries with salt a''
   - The salt a has no more than $4mT_1/k$ successor queries
   - The salt a has no more than $4mT_1/k$ successor queries
4. If no such S is found, it returns $\varnothing$
5. For every $\{a', a''\}$ pair in S it picks one type 1 hexagon formed by the queries of $\mathcal{A}_1$ (if there is more than one, it picks one arbitrarily). It lets the queries forming the hexagon be $q_1 = h(a', M_1) = w, q_2 = h(a, M_2) = y, q_3 = h(y, M_3) = z, q_4 = h(a'', M_4) = w, q_5 = h(a', M_5) = y', q_6 = h(y', M_6) = z$ for some $M_1, M_2, M_3, M_4, M_5, M_6, w, y, y', z$ such that $|\{a', a'', w, y, y', z\}| = 6$ such that q_4 was the last of the six queries made by $\mathcal{A}_1$.
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_4 query for a salt pair $\{a, a'\} \in S$
      i. it adds the index of this query in T
      ii. it adds the lexicographical index of the corresponding q_6 query among all successor queries of a to the list L_1
      iii. Let the answer of the corresponding q_6 query be z. It adds the lexicographical index of the corresponding q_3 query among all queries with answer z the list L_2
      iv. Let the input salt of the corresponding q_3 query be y'. It adds the lexicographical index of the corresponding q_2 query among all queries with answer y' to the list L_3
      v. Let the input salt of the corresponding q_2 query be a'. It adds the lexicographical index of the corresponding q_1 query among all queries with input salt a' to the list L_4.
   (b) Otherwise, it adds $h(a, M)$ to L_5.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_5 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{L}_4, \mathsf{L}_5, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T,
      i. it removes the element i on the front of list L_1, and locates the query q_6 which has lexicographic order i among all successor queries to a.
      ii. Let the answer q_6 be z. It removes the element i on the front of list L_2, and locates the query q_3 which has lexicographic order i among all queries with answer z.
      iii. Let the input salt of q_3 be y. It removes the element i on the front of list L_3, and locates the query q_2 which has lexicographic order i among all queries with answer y.
      iv. Let the input salt of q_2 be a'. It removes the element i on the front of list L_4, and locates the query q_1 which has lexicographic order i among all queries with input salt a'

      It answers with the answer of query q_1.
   (b) Otherwise, it removes the element in front of L_5 and answers with that.

4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_5

□

We finally prove Lemma 15.

Proof. We want to upper bound

$$\Pr\left[\mathsf{off\text{-}2\text{-}dia\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k' \wedge \neg\mathsf{off\text{-}hex\text{-}}k'' \wedge \neg(m+1)\text{-}\mathsf{col}\right] .$$

Note that if $\mathcal{A}_1$ does not cause $\mathsf{off\text{-}zz\text{-}}k', \mathsf{off\text{-}hex\text{-}}k''$ and finds diamonds of category two for at least k salts, it means that there is a set of at least $k - 2k' - 2k''$ salts for which $\mathcal{A}_1$ finds diamonds of category two such that no two diamonds in the set share the last query completing the diamond. We will use this to intuition to build a compression argument that upper bounds this probability.

The encoding procedure encodes the random oracle h as follows.

1. It first runs $\mathcal{A}_1^h$, answering all its queries using h.
2. If $\mathsf{off\text{-}zz\text{-}}k'$ happens or $\mathsf{off\text{-}hex\text{-}}k''$ happens or $(m+1)$-col happens or $\mathsf{off\text{-}2\text{-}dia\text{-}}k$ does not happen, it outputs $\varnothing$
3. It finds a set of salts S of size $(k - 2k' - 2k'')/2$ such that for each $a \in S$
   - $\mathcal{A}_1$ found a category 2 diamond for a
   - The salt a has no more than $2T_1 m/(k - 2k' - 2k'')$ successor queries
   - The category 2 diamond for no two salts in the set share the last query of the diamond
4. If no such S is found, return $\varnothing$
5. For every a' in S it picks the category 2 diamond that does not share the last query with the diamond of any other salt in S found by $\mathcal{A}_1$ (if there is more than one, it picks one arbitrarily). It lets the queries forming the diamond be $q_1 = h(a', M_1) = y, q_2 = h(y, M_2) = z, q_3 = h(a', M_3) = y', q_4 = h(y', M_4) = z$ for some $M_1, M_2, M_3, M_4, y, y', z$ such that $|\{a', y, y', z\}| = 4$ such that q_4 was the last of the four queries made by $\mathcal{A}_1$.
6. It initializes lists $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3$ to empty lists, T to empty set
7. It starts running $\mathcal{A}_1$ again
8. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the query is marked as a q_4 query for a salt $a \in S$, then
      i. it adds the index of this query in T
      ii. it adds the lexicographical order of q_3 among all the queries made by $\mathcal{A}_1$ with answer a in the list L_1
      iii. Let the input salt of q_3 be a'. It adds the lexicographical order of q_2 among all the successor queries of the salt a' in the list L_2
   (b) Otherwise it adds $h(a, M)$ to L_3.
9. It appends the evaluation of h on the points not queried by $\mathcal{A}_1$ to L_3 in the lexicographical order of the inputs.
10. It outputs $\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3, \mathsf{T}$.

The decoding procedure works as follows.

1. If the encoding is $\varnothing$ it aborts.
2. It runs $\mathcal{A}_1^h$.
3. For every query $h(a, M)$ made by $\mathcal{A}_1$ it does the following:
   (a) If the index of the query is in T,
      i. it removes the element i on the front of list L_1, and locates the query q_3 which has lexicographical order i among the queries with answer a
      ii. Let a' be the input salt of q_3. It removes the element i on the front of list L_2, and locates the query q_2 which has lexicographical order i among all the successor query of salt a'. It answers with the answer of q_2

   (b) Otherwise it removes the element in front of L_3 and answers with that.
4. It populates h on the points not queried by $\mathcal{A}_1$ in the lexicographical order by the remaining entries of L_3

Correctness of decoding: First off, we argue that for adversary $\mathcal{A}_1$ that causes the event $\mathsf{off\text{-}2\text{-}dia\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k' \wedge \neg\mathsf{off\text{-}hex\text{-}}k'' \wedge \neg(m+1)\text{-}\mathsf{col}$ to happen, the encoding algorithm will never return $\varnothing$. To argue this it suffices to show that the encoding algorithm never returns $\varnothing$ from line 4. Note that the encoding algorithm reaches this line only if the event $\mathsf{off\text{-}2\text{-}dia\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k' \wedge \neg\mathsf{off\text{-}hex\text{-}}k'' \wedge \neg(m+1)\text{-}\mathsf{col}$ was caused by running $\mathcal{A}_1$, which means there is a set of k salts S' such that $\mathcal{A}_1$ found category two diamonds for them. Moreover, there does not exist a set of k' salt pairs for which $\mathcal{A}_1$ found a zig-zag. Also, there does not exist a set of k' salt pairs for which $\mathcal{A}_1$ found a hexagon. Now let P be the maximally large set of salt pairs such that

1. each salt in the salt pairs of P is in S'
2. no two salt pairs in P share a salt
3. $\mathcal{A}_1$ found a hexagon for each salt pair in P

Since $\neg\mathsf{off\text{-}hex\text{-}}k''$ happens we have that $|\mathsf{P}| < k''$. Let S'' be all the salts in the salt pairs in P. We have that $|\mathsf{S}'\backslash\mathsf{S}''| = |\mathsf{S}'| - 2|\mathsf{P}| > k - 2k''$. Now let P' be the maximally large set of salt pairs such that

1. each salt in the salt pairs of P' is in $\mathsf{S}'\backslash\mathsf{S}''$
2. no two salt pairs in P' share a salt
3. $\mathcal{A}_1$ found a zigzag for each salt pair in P'

Since $\neg\mathsf{off\text{-}zz\text{-}}k'$ happens we have that $|\mathsf{P}'| < k'$. Let S''' be all the salts in the salt pairs in P. We have that $|\mathsf{S}'\backslash\mathsf{S}''\backslash\mathsf{S}'''| = |\mathsf{S}'\backslash\mathsf{S}'| - 2|\mathsf{P}'|| > k - 2k'' - 2k'$. Let Notice that for every salt in a in $\mathsf{S}'\backslash\mathsf{S}''\backslash\mathsf{S}'''$, $\mathcal{A}_1$ found a category 2 diamond for a and category 2 diamond for no two salts in the set share the last query of the diamond by the maximality of $\mathsf{S}'', \mathsf{S}'''$. Further, since $\neg(m+1)\text{-}\mathsf{col}$ happens, the total number of salt-successor query pairs are at most mT_1. Therefore, at least $(k - 2k'' - 2k')/2$ of the salts (note that $(k - 2k' - 2k'')/2$ is an integer because k is a multiple of 2) in $\mathsf{S}'\backslash\mathsf{S}''\backslash\mathsf{S}'''$ are such that there are at most $2mT_1/(k - 2k' - 2k'')$ successor queries for the salt. Therefore, such a set S always exists.

If the encoding algorithm does not return $\varnothing$, it is easy to see the decoding algorithm decodes correctly, because the query answers that the encoding algorithm does not add in the list L_3 can be recovered by the decoding algorithm correctly using $\mathsf{S}, \mathsf{L}_1, \mathsf{L}_2$. This can be verified by inspection. Therefore, we have that

$$\begin{aligned}&\Pr[\text{ Decoding is correct }]\\ &\geqslant \Pr\left[\mathsf{off\text{-}2\text{-}dia\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k' \wedge \neg\mathsf{off\text{-}hex\text{-}}k'' \wedge \neg(m+1)\text{-}\mathsf{col}\right] .\end{aligned}$$

Observe that the encoding algorithm removes $(k - 2k' - 2k'')/2$ answers of the random oracle from the encoding, and all the removed answers are distinct because those were q_1 queries for different salts. It instead adds an set T of $(k - 2k' - 2k'')/2$ distinct values in $[T_1]$, and sequence L_1 of $(k - 2k' - 2k)/2$ values that are at most m, and sequence L_2 of $(k - 2k' - 2k)/2$ values that are at most $2mT_1/(k - 2k' - 2k'')$. Using the compression lemma, we have that

$$\begin{aligned}&\Pr[\text{ Decoding is correct }]\\ &\leqslant \frac{\binom{T_1}{(k-2k'-2k'')/2} m^{(k-2k'-2k'')/2}(2mT_1/(k-2k'-2k''))^{(k-2k'-2k'')/2}}{2^{n(k-2k'-2k'')/2}}\\ &\leqslant \left(\frac{4em^2T_1^2}{(k-2k'-2k'')^2 2^n}\right)^{(k-2k'-2k'')/2} .\end{aligned}$$

Therefore,

$$\begin{aligned}&\Pr\left[\mathsf{off\text{-}2\text{-}dia\text{-}}k \wedge \neg\mathsf{off\text{-}zz\text{-}}k' \wedge \neg\mathsf{off\text{-}hex\text{-}}k'' \wedge \neg(m+1)\text{-}\mathsf{col}\right]\\ &\leqslant \left(\frac{4em^2T_1^2}{(k-2k'-2k'')^2 2^n}\right)^{(k-2k'-2k'')/2} .\end{aligned}$$

□

Acknowledgements

This research was partially supported by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

ABD$^+$15. David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 5–17. ACM Press, October 2015. 2

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, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 157–186. Springer, Heidelberg, August 2020. 1, 2, 3

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, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 192–221. Springer, Heidelberg, August 2022. 1, 2, 3

BL13. Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2013. 1

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, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 693–721. Springer, Heidelberg, August 2018. 1

CDGS18. Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Heidelberg, April / May 2018. 1, 2

CK18. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 415–447. Springer, Heidelberg, April / May 2018. 1, 4, 7, 61

Dam90. Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990. 3

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, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 473–495. Springer, Heidelberg, April / May 2017. 1

DTT10. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Heidelberg, August 2010. 1, 5

GK22. Ashrujit Ghoshal and Ilan Komargodski. On time-space tradeoffs for bounded-length collisions in merkle-Damgård hashing. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 161–191. Springer, Heidelberg, August 2022. 1, 2, 3

GPR14. Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact PRF-security of NMAC and HMAC. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 113–130. Springer, Heidelberg, August 2014. 1

Hel80. Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980. 1

JT20. Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 414–443. Springer, Heidelberg, November 2020. 2, 3, 8, 9, 10

KM12. Neal Koblitz and Alfred Menezes. Another look at HMAC. Cryptology ePrint Archive, Report 2012/074, 2012. `https://eprint.iacr.org/2012/074`. 1

Mer90. Ralph C. Merkle. A fast software one-way hash function. *Journal of Cryptology*, 3(1):43–58, January 1990. 3

Oec03. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 617–630. Springer, Heidelberg, August 2003. 2

Rog06. Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 211–228. Springer, Heidelberg, September 2006. 1
Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 5, 7
Unr07. Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Heidelberg, August 2007. 1

A Attacks

In this section we add the proof of the various theorems that show the tightness of the bounds we obtained for security against offline-online adversaries.

A.1 Attacks for pre-2-CR-MD_n^h (Proofs of Theorems 8 and 9)

We first prove Theorem 9.

Proof (Theorem 9). Let $T_1/2^{n/2+2} = k$. We start by describing $\mathcal{A}$. Its offline phase $\mathcal{A}_1$ does the following.

1. Let S be a set of k distinct salts.
2. For each salt a in S, it makes $2^{n/2+2}$ distinct queries with the salt
3. For each salt in $\{0,1\}^n \backslash \mathsf{S}$, it makes $T_1/2^{n+1}$ distinct queries

Its online phase $\mathcal{A}_2$ does the following: on getting input salt a, it checks if $\mathcal{A}_1$ had made queries that led to a two-block MD-collision for a. If so it outputs the collision.

Clearly $\mathcal{A}_1$ makes $T_1/2^{n/2+2} \cdot 2^{n/2+1} + T_1/2^{n+1} \cdot (2^n - T_1/2n/2+2) < T_1$ queries, and $\mathcal{A}_2$ makes no queries. So $\mathcal{A}$ is a $(T_1, 0)$-adversary.

Now, we analyze the advantage of $\mathcal{A}$. We say that a salt a' in S is *good* if the $\mathcal{A}_1$ made two queries for a' that collided. For each $a' \in \mathsf{S}$, we have that

$$\Pr\left[a' \text{ is good}\right] = 1 - \Pr\left[a' \text{ is not good}\right] .$$

Observe that a' is not good if and only if all the $2^{n/2+1}$ queries made on it produce distinct answers

$$\begin{aligned}\Pr\left[a' \text{ is not good}\right] &= \prod_{i=1}^{2^{n/2+1}} \left(1 - \frac{i-1}{2^n}\right) \\ &\leqslant \prod_{i=1}^{2^{n/2+1}} e^{-\frac{(i-1)}{2^n}} \\ &= e^{-\frac{(2^{n/2+1})(2^{n/2+1}-1)}{2^n}} \leqslant 1/e .\end{aligned}$$

Let X be the random variable that denotes the number of salts in S that are not good. We have that $\mathsf{E}[X] \leqslant k/e$. Using Markov's equality we have that

$$\Pr\left[X \geqslant 2k/e\right] \leqslant \frac{\mathsf{E}[X]}{2k/e} \leqslant \frac{1}{2} .$$

Therefore,

$$\Pr\left[X \leqslant 2k/e\right] \geqslant \frac{1}{2} .$$

Let Q_a be the event that for a salt a, $\mathcal{A}_1$ made a query on a in line 3 such that one of the queries had an answer a' such that a' is a good salt in S. Let $\mathsf{win}_{\mathcal{A}}$ be the event that the a that was sampled before the online phase was such that

1. $a \notin \mathsf{S}$
2. Q_a happens

We have that

$$\begin{aligned}\Pr[\mathsf{win}_{\mathcal{A}}] &= \Pr[a \notin \mathsf{S} \wedge Q_a] \\ &\geqslant \Pr[a \notin \mathsf{S} \wedge Q_a \wedge X \leqslant 2k/e] \\ &\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] \Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \ .\end{aligned}$$

Since a is sampled independent of $\mathcal{A}_1$ we have that

$$\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] = \Pr[a \notin \mathsf{S}] \cdot \Pr[X \leqslant 2k/e] \ .$$

$\Pr[a \notin \mathsf{S}]\ 1 - k/2^n$ since a is sampled uniformly at random. Since $T_1 \leqslant 2^{5n/4}$, and $n \geqslant 2$, $\Pr[a \notin \mathsf{S}] \geqslant 1/2$. Further we have $\Pr[X \leqslant 2k/e] \geqslant 1/2$. Therefore,

$$\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] \geqslant 1/4 \, .$$

If $a \notin \mathsf{S}$, Q_a does not happen only if none queries made on $\mathcal{A}_1$ have answers that are not among the good salts in S. Therefore, we have that

$$\Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \geqslant 1 - \left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \ .$$

We have that

$$\begin{aligned}\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} &= \left(1 - \frac{k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \\ &\leqslant e^{-\frac{k(1-2/e)}{2^n} \cdot \frac{T_1}{2^{n+1}}} \ .\end{aligned}$$

Note that

$$\frac{k(1-2/e)}{2^n} \cdot \frac{T_1}{2^{n+1}} = \frac{(1-2/e)T_1^2}{2^{5n/2+3}} \leqslant 1 \, .$$

Above we used the fact that $T_1 \leqslant 2^{5n/4}$. We have that $e^{-x} \leqslant 1 - x/2$ for $0 \leqslant x \leqslant 1.5$, therefore,

$$\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \leqslant 1 - \frac{(1-2/e)T_1^2}{2^{5n/2+4}} \ .$$

This implies

$$\Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \geqslant \frac{(1-2/e)T_1^2}{2^{5n/2+4}} \ .$$

Therefore,

$$\Pr[\mathsf{win}_{\mathcal{A}}] \geqslant (1/4) \cdot \frac{(1-2/e)T_1^2}{2^{5n/2+4}} = \frac{(1-2/e)T_1^2}{2^{5n/2+6}} \ .$$

Clearly if $\mathsf{win}_{\mathcal{A}}$ happens, $\mathcal{A}$ has won. Hence

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{n,\ell,n}} \geqslant \frac{(1-2/e)T_1^2}{2^{5n/2+6}} \ .$$

□

We next prove Theorem 8.

Proof (Theorem 8). Let $T_1/2^{n/2+1} = k$. We start by describing $\mathcal{A}$. Its offline phase $\mathcal{A}_1$ does the following.

1. Let S be a set of k distinct salts.
2. For each salt a in S, it makes $2^{n/2+1}$ distinct queries with the salt

Its online phase $\mathcal{A}_2$ does the following:

1. Let T be the set of salts for which $\mathcal{A}_1$ found a collision.
2. On getting salt a input it makes T_2 distinct queries with salt a. If any of the answers are in T – say the query $h(a, M)$ produced a' such that $\mathcal{A}_1$ had made queries $h(a', M') = z$ and $h(a', M'') = z$ for $M' \neq M''$. It outputs (M, M') and (M, M'') as the two-block collision.

Clearly $\mathcal{A}_1$ makes $T_1/2^{n/2+1} \cdot 2^{n/2+1} = T_1$ queries, and $\mathcal{A}_2$ makes T_2 queries. So $\mathcal{A}$ is a (T_1, T_2)-adversary.

Now, we analyze the advantage of $\mathcal{A}$. We say that a salt a' in S is *good* if the $\mathcal{A}_1$ made two queries for a' that collided. Let X be the random variable that denotes the number of salts in S that are not good.

In our analysis in 9, we showed that

$$\Pr\left[X \leqslant 2k/e\right] \geqslant \frac{1}{2} \,.$$

Let $\mathsf{win}_{\mathcal{A}}$ be the event that for a salt a, $\mathcal{A}_2$ made a query on a such that one of the queries had an answer a' such that a' is a good salt in S. We have that

$$\begin{aligned}\Pr\left[\mathsf{win}_{\mathcal{A}}\right] \geqslant \Pr\left[\mathsf{win}_{\mathcal{A}} \wedge X \leqslant (2k/e\right] \\ \Pr\left[X \leqslant 2k/e\right] \Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] \,.\end{aligned}$$

Note that $\mathsf{win}_{\mathcal{A}}$ does not happen only if none queries made on $\mathcal{A}_2$ have answers that are not among the good salts in S. Therefore, we have that

$$\Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] \geqslant 1 - \left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_2} \,.$$

We have that

$$\begin{aligned}\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_2} &= \left(1 - \frac{k(1-2/e)}{2^n}\right)^{T_2} \\ &\leqslant e^{-\frac{k(1-2/e)}{2^n}\cdot T_2} \,.\end{aligned}$$

Note that

$$\frac{k(1-2/e)}{2^n} \cdot T_2 = \frac{(1-2/e)T_1T_2}{2^{3n/2+1}} \leqslant 1 \,.$$

Above we used the fact that $T_1T_2 \leqslant 2^{3n/2}$. We have that $e^{-x} \leqslant 1 - x/2$ for $0 \leqslant x \leqslant 1.5$. Therefore,

$$\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_2} \leqslant 1 - \frac{(1-2/e)T_1T_2}{2^{3n/2+2}} \,.$$

This implies

$$\Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] \geqslant \frac{(1-2/e)T_1T_2}{2^{3n/2+2}} \,.$$

Therefore,

$$\Pr\left[\mathsf{win}_{\mathcal{A}}\right] \geqslant (1/2) \cdot \frac{(1-2/e)T_1T_2}{2^{3n/2+2}} = \frac{(1-2/e)T_1T_2}{2^{3n/2+3}} \,.$$

Clearly if $\mathsf{win}_{\mathcal{A}}$ happens, $\mathcal{A}$ has won. Hence

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}CR\text{-}MD}}_{H_{n,\ell,n}} \geqslant \frac{(1-2/e)T_1T_2}{2^{3n/2+3}} \,.$$

□

A.2 Attacks on pre-2-PR-MD^h_n (Proofs of Theorems 5 and 6)

We first prove Theorem 5.

Proof (Theorem 5). Let $k = T_1/2^n$. The offline phase of the adversary $\mathcal{A}$ does the following:

1. It chooses a set of k distinct salts S
2. For each salt $a \in \mathsf{S}$, it makes 2^{n+2} distinct queries

Its online phase $\mathcal{A}_2$ does the following: on getting input salt a, it makes T_2 distinct queries with salt a. If some query $h(a, M') = a'$ such that $a' \in \mathsf{S}$ and for $\mathcal{A}_1$ had made a query $h(a', M'') = y$ then $\mathcal{A}_2$ outputs the (M', M'').

Clearly $\mathcal{A}_1$ makes $T_1/2^n \cdot 2^n = T_1$ queries, and $\mathcal{A}_2$ makes T_2 queries. So $\mathcal{A}$ is a (T_1, T_2)-adversary.

Now, we analyze the advantage of $\mathcal{A}$. We say that a salt a' in S is *good* if the $\mathcal{A}_1$ made a query for a' that had answer y. For each $a' \in \mathsf{S}$, we have that

$$\Pr\left[a' \text{ is good}\right] = 1 - \Pr\left[a' \text{ is not good}\right] .$$

Observe that a' is not good if and only if all the 2^n queries made on it produce answers other than y

$$\begin{aligned}\Pr\left[a' \text{ is not good}\right] &= \left(1 - \frac{1}{2^n}\right)^{2^n} \\ &\leqslant 1/e .\end{aligned}$$

Let X be the random variable that denotes the number of salts in S that are not good. We have that $\mathsf{E}[X] \leqslant k/e$. Using Markov's equality we have that

$$\Pr[X \geqslant 2k/e] \leqslant \frac{\mathsf{E}[X]}{2k/e} \leqslant \frac{1}{2} .$$

Therefore,

$$\Pr[X \leqslant 2k/e] \geqslant \frac{1}{2} .$$

Let $\mathsf{win}_{\mathcal{A}}$ be the event that for a salt a, $\mathcal{A}_2$ made a query on a such that one of the query had an answer a' such that a' is a good salt in S. We have that

$$\begin{aligned}\Pr[\mathsf{win}_{\mathcal{A}}] &\geqslant \Pr[\mathsf{win}_{\mathcal{A}} \wedge X \leqslant 2k/e] \\ &= \Pr[X \leqslant 2k/e] \Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] .\end{aligned}$$

Note that $\mathsf{win}_{\mathcal{A}}$ does not happen only if none queries made on $\mathcal{A}_1$ have answers that are not among the good salts in S. Therefore, we have that

$$\Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] \geqslant 1 - \left(\frac{2^n - k(1 - 2/e)}{2^n}\right)^{T_2} .$$

We have that

$$\begin{aligned}\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} &= \left(1 - \frac{k(1-2/e)}{2^n}\right)^{T_2} \\ &\leqslant e^{-\frac{k(1-2/e)}{2^n} \cdot T_2} .\end{aligned}$$

Note that

$$\frac{k(1-2/e)}{2^n} \cdot T_2 = \frac{(1-2/e)T_1 T_2}{2^{2n}} \leqslant 1 .$$

Above we used the fact that $T_1T_2 \leqslant 2^{2n}$. We have that $e^{-x} \leqslant 1 - x/2$ for $0 \leqslant x \leqslant 1.5$, therefore,

$$\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_2} \leqslant 1 - \frac{(1-2/e)T_1T_2}{2^{2n}} \ .$$

This implies

$$\Pr\left[\mathsf{win}_{\mathcal{A}} \,\middle|\, X \leqslant 2k/e\right] \geqslant \frac{(1-2/e)T_1T_2}{2^{2n}} \ .$$

Therefore,

$$\Pr\left[\mathsf{win}_{\mathcal{A}}\right] \geqslant (1/2) \cdot \frac{(1-2/e)T_1T_2}{2^{2n}} = \frac{(1-2/e)T_1T_2}{2^{2n+1}} \ .$$

Clearly if $\mathsf{win}_{\mathcal{A}}$ happens, $\mathcal{A}$ has won. Hence,

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}} \geqslant \frac{(1-2/e)T_1T_2}{2^{2n+1}} \ .$$

□

We next prove Theorem 6.

Proof (Theorem 6). Let $k = T_1/2^{n+1}$. The offline phase of the adversary $\mathcal{A}$ does the following:

1. It chooses a set of k distinct salts S
2. For each salt $a \in \mathsf{S}$, it makes 2^{n+2} distinct queries
3. For every salt in $\{0,1\}^n \backslash \mathsf{S}$ it makes $T_1/2^{n+1}$ distinct queries

Its online phase $\mathcal{A}_2$ does the following: on getting input salt a, it checks the queries made by $\mathcal{A}_1$ to find a message M, at most two blocks long such that the $\mathsf{MD}^h(a, M) = y$. If so it outputs M.

Clearly $\mathcal{A}_1$ makes $T_1/2^{n+1} \cdot 2^n + T_1/2^{n+1} \cdot (2^n - T_1/2n/2 + 2) < T_1$ queries, and $\mathcal{A}_2$ makes no queries. So $\mathcal{A}$ is a $(T_1, 0)$-adversary.

Now, we analyze the advantage of $\mathcal{A}$. We say that a salt a' in S is *good* if the $\mathcal{A}_1$ made a query for a' that had answer y. For each $a' \in \mathsf{S}$, we have that

$$\Pr\left[a' \text{ is good}\right] = 1 - \Pr\left[a' \text{ is not good}\right] \ .$$

Observe that a' is not good if and only if all the 2^n queries made on it produce answers other than y

$$\begin{aligned}\Pr\left[a' \text{ is not good}\right] &= \left(1 - \frac{1}{2^n}\right)^{2^n} \\ &\leqslant 1/e \ .\end{aligned}$$

Let X be the random variable that denotes the number of salts in S that are not good. We have that $\mathsf{E}\left[X\right] \leqslant k/e$. Using Markov's equality we have that

$$\Pr\left[X \geqslant 2k/e\right] \leqslant \frac{\mathsf{E}\left[X\right]}{2k/e} \leqslant \frac{1}{2} \ .$$

Therefore,

$$\Pr\left[X \leqslant 2k/e\right] \geqslant \frac{1}{2} \ .$$

Let Q_a be the event that for a salt a, $\mathcal{A}_1$ made a query on a in line 3 such that one of the queries had an answer a' such that a' is a good salt in S. Let $\mathsf{win}_{\mathcal{A}}$ be the event that the a that was sampled before the online phase was such that

1. $a \notin \mathsf{S}$
2. Q_a happens

We have that

$$\begin{aligned}\Pr[\mathsf{win}_{\mathcal{A}}] &= \Pr[a \notin \mathsf{S} \wedge Q_a] \\ &\geqslant \Pr[a \notin \mathsf{S} \wedge Q_a \wedge X \leqslant 2k/e] \\ &\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] \Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \ .\end{aligned}$$

Since a is sampled independent of $\mathcal{A}_1$ we have that

$$\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] = \Pr[a \notin \mathsf{S}] \cdot \Pr[X \leqslant 2k/e] \ .$$

We have that $\Pr[a \notin \mathsf{S}] = 1 - k/2^n$ since a is sampled uniformly at random. Since $T_1 \leqslant 2^{3n/2}$, and $n \geqslant 2$, $\Pr[a \notin \mathsf{S}] \geqslant 1/2$. Further we have $\Pr[X \leqslant 2k/e] \geqslant 1/2$. Therefore,

$$\Pr[a \notin \mathsf{S} \wedge X \leqslant 2k/e] \geqslant 1/4 \ .$$

If $a \notin \mathsf{S}$, Q_a does not happen only if none queries made on $\mathcal{A}_1$ have answers that are not among the good salts in S. Therefore, we have that

$$\Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \geqslant 1 - \left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \ .$$

We have that

$$\begin{aligned}\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} &= \left(1 - \frac{k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \\ &\leqslant e^{-\frac{k(1-2/e)}{2^n} \cdot \frac{T_1}{2^{n+1}}}\end{aligned}$$

Note that

$$\frac{k(1-2/e)}{2^n} \cdot \frac{T_1}{2^{n+1}} = \frac{(1-2/e)T_1^2}{2^{3n+2}} \leqslant 1 \ .$$

Above we used the fact that $T_1 \leqslant 2^{3n}$. We have that $e^{-x} \leqslant 1 - x/2$ for $0 \leqslant x \leqslant 1.5$, therefore,

$$\left(\frac{2^n - k(1-2/e)}{2^n}\right)^{T_1/2^{n+1}} \leqslant 1 - \frac{(1-2/e)T_1^2}{2^{3n+2}} \ .$$

This implies

$$\Pr\left[Q_a \,\middle|\, a \notin \mathsf{S} \wedge X \leqslant 2k/e\right] \geqslant \frac{(1-2/e)T_1^2}{2^{3n+2}} \ .$$

Therefore,

$$\Pr[\mathsf{win}_{\mathcal{A}}] \geqslant (1/4) \cdot \frac{(1-2/e)T_1^2}{2^{3n+2}} = \frac{(1-2/e)T_1^2}{2^{3n+4}} \ .$$

Clearly if $\mathsf{win}_{\mathcal{A}}$ happens, $\mathcal{A}$ has won. Hence

$$\mathsf{Adv}^{\mathsf{pre\text{-}2\text{-}PR\text{-}MD}}_{H_{n,\ell,n}} \geqslant \frac{(1-2/e)T_1^2}{2^{3n+4}} \ .$$

□

A.3 Offline-only attack for s-pre-CR^h (Proof of Theorem 3)

In this section we prove Theorem 3.

Proof (Theorem 3). Let $k = T_1/2^{n/2+1}$. We describe the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Its offline phase $\mathcal{A}_1$ works as follows:

1. It picks a set S of k distinct salts
2. For each salt in S it makes $2^{n/2+1}$ distinct queries

The online phase on getting salt a as input simply checks whether $\mathcal{A}_1$ queried a collision for $\mathcal{A}$. If so it outputs the collision.

We say that a salt a' in S is *good* if the $\mathcal{A}_1$ made two queries for a' that collided. For each $a' \in \mathsf{S}$, we have that

$$\Pr\left[a' \text{ is good}\right] = 1 - \Pr\left[a' \text{ is not good}\right] \ .$$

Observe that a' is not good if and only if all the $2^{n/2+1}$ queries made on it produce distinct answers

$$\begin{aligned}\Pr\left[a' \text{ is not good}\right] &= \prod_{i=1}^{2^{n/2+1}} \left(1 - \frac{i-1}{2^n}\right) \leqslant \prod_{i=1}^{2^{n/2+1}} e^{-\frac{(i-1)}{2^n}} \\ &= e^{-\frac{(2^{n/2+1})(2^{n/2+1}-1)}{2^n}} \leqslant 1/e \ .\end{aligned}$$

Now, notice that $\mathcal{A}$ wins whenever a is sampled from the set of good salts in S. Therefore,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CR}}_{H_{s,m,n}}(\mathcal{A}) &\geqslant \Pr\left[a \in \mathsf{S} \wedge a \text{ is a good salt in } \mathsf{S}\right] \\ &= \Pr\left[a \in \mathsf{S}\right] \cdot \Pr\left[a \text{ is a good salt in } \middle| a \in \mathsf{S}\right] \\ &\geqslant (1 - 1/e)k/2^n = \frac{(1-1/e)T_1}{2^{s+n/2+1}} \ .\end{aligned}$$

The second inequality above follows since a is sampled at random and for all $a' \in \mathsf{S}$, $\Pr\left[a' \text{ is good}\right] \geqslant 1 - 1/e$. □

We note that there is a dual of this strategy for an offline-only adversary compared to the one we presented in this proof – instead of making $2^{n/2}$ queries for $T_1/2^{n/2}$ salts, it could make $T_1/2^s$ for all 2^s salts – in this case its advantage of be of the order $T_1^2/2^{2s+n}$. Note that if we care about constant advantage, then both the attack strategies need the same amount of queries. However, this latter attack is worse than the one we give in Theorem 3 since $T_1^2/2^{2s+n} \leqslant T_1/2^{s+n/2}$ when both the terms are at most one. In Appendix B, we formalize this difference between the two attack strategies for a general class of games.

B Strategies for offline-only attacks

In this section we present the qualitative difference between two strategies behind offline only attacks.

Suppose game G is compatible with ideal distribution $\mathcal{I}$. Suppose there exists $d, N, T_0 \in \mathbb{N}$, $c \in [0,1]$ that depend on $\mathsf{G}, \mathcal{I}$ and satisfy the following.

– $T_0 \geqslant N^{1/d}$
– for all $T \leqslant T_0$, there exists a T-query adversary $\mathcal{A}_T$ such that $\mathsf{Adv}^{\mathsf{G}}_{\mathcal{I}}(\mathcal{A}_T) \geqslant c \cdot \frac{T^d}{N}$

$$\mathsf{Adv}^{\mathsf{G}}_{\mathcal{I}}(\mathcal{A}) \geqslant c \cdot \frac{T^d}{N} \ .$$

This property hold for several random oracle based games, e.g., for collision-resistance of random oracles with n bit outputs, $c = 1/2, d = 2, T_0 = 2^{n/2}$.

Let $T_1, s \in \mathbb{N}_{>0}$. Assume T_1 is a multiple of 2^s, and $N^{1/d}$, and $T_1 \leqslant N^{1/d} \cdot 2^s$. Consider the two following T_1-query offline-only adversaries $\mathcal{B}, \mathcal{C}$ against $s\text{-pre-}\mathsf{G}^\pi$.

– $\mathcal{B}_1$: It chooses a set S of $T_1/N^{1/d}$ salts. For each salt a in the set, it runs $\mathcal{A}_T$ for $T = N^{1/d}$ and simulates G^{π_a} to it.
– $\mathcal{C}_1$: For each salt $a \in \{0,1\}^s$, it runs $\mathcal{A}_{T'}$ for $T' = T_1/2^s$, simulating G^{π_a} to it.

The online phases of both adversaries do not make any queries themselves, and decide their output based on the queries made by their offline phase.

First we analyze the success probability of $\mathcal{B}$. We have that $\mathcal{B}$ wins if the sampled salt a was in S and $\mathsf{G}^{\pi_a}(\mathcal{A}_T)$ (where $T = N^{1/d}$) returns true. Therefore,

$$\begin{aligned}\mathsf{Adv}^{s\text{-pre-G}}_{\mathcal{I}_s}(\mathcal{B}) &= \Pr\left[\{\}\, a \in \mathsf{S}\right\} \Pr\left[\mathsf{G}^{\pi_a}(\mathcal{A}_T) \,\middle|\, a \in \mathsf{S}\right] \\ &\geqslant \frac{T_1}{N^{1/d}2^s} \cdot c = \frac{cT_1}{N^{1/d}2^s} \ .\end{aligned}$$

The inequality follows because a is sampled independently of $\mathcal{A}$, and for any salt a, $\mathsf{Adv}^{\mathsf{G}}_{\mathcal{I}}(\mathcal{A}_T) \geqslant c \cdot \frac{T^d}{N} = c$.

Next we analyze the success probability of $\mathcal{C}$. We have that $\mathcal{C}$ wins if $\mathsf{G}^{\pi_a}(\mathcal{A}_{T'})$ (where $T' = T_1/2^s$) returns true. Therefore,

$$\begin{aligned}\mathsf{Adv}^{s\text{-pre-G}}_{\mathcal{I}_s}(\mathcal{B}) &= \Pr\left[\mathsf{G}^{\pi_a}(\mathcal{A}_{T'})\right] \\ &\geqslant c \cdot \frac{T_1^d}{2^{sd}N} \ .\end{aligned}$$

Note that since $T_1 \leqslant N^{1/d} \cdot 2^s$, $\frac{cT_1}{N^{1/d}2^s} \geqslant \frac{T_1^d}{2^{sd}N}$. So for $d \geqslant 1$, the better offline only strategy is to trying to win on $T_1/N^{1/d}$ salts instead of distributing queries across all 2^s salts.

C Upper Bounding Offline-Online advantage via the Multi-Instance Approach

In [CK18], the authors remark that in personal communication with them, Dan Bernstein noted that a lower bound against multiple-discrete-log algorithms also yields lower bounds on the preprocessing time for discrete-log algorithms with preprocessing. In this section, we explore an example of using this approach to prove guarantees against offline-online adversaries against collision-resistance of a salted random oracle.

This approach upper bounds the advantage of a (T_1, T_2)-adversary $\mathcal{A}$ against the collision-resistance of a salted random oracle in terms of the advantage of an adversary $\mathcal{B}$ against a multi-instance version of the game that makes $T_1 + kT_2$ queries. The oracle game $s\text{-mi-}k\text{-}\mathsf{CR}^h$ in Fig 16 is the multi-instance version of the salted collision-resistance game: the adversary gets k distinct salts as input, and wins if it finds a collision for each of them. (Note that $k \leqslant 2^s$ since there are at most 2^s salts).

We prove the following theorem.

Theorem 10. *Let $s, \ell, n \in \mathbb{N}_{>0}$. Let $H_{s,\ell,n}$ be the uniform distribution over $\mathsf{Fcs}(\{0,1\}^s \times \{0,1\}^\ell, \{0,1\}^n)$. Let $\mathcal{A}$ be a (T_1, T_2) adversary such that*

$$\mathsf{Adv}^{s\text{-pre-CR}}_{H_{s,\ell,n}}(\mathcal{A}) = \epsilon \ .$$

Then for any $\delta \leqslant \epsilon$, and any $k \leqslant \delta \cdot 2^s$, there exists an adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{s\text{-mi-}k\text{-CR}}_{H_{s,\ell,n}}(\mathcal{B}) \geqslant (\epsilon - \delta)\frac{\binom{\delta \cdot 2^s}{k}}{\binom{2^s}{k}} \ .$$

Moreover $\mathcal{B}$ makes $T_1 + kT_2$ queries to h.

```
Game $s$-$\mathsf{mi}$-$k$-$\mathsf{CR}^h(\mathcal{A})$
$a_1, \ldots, a_k \leftarrow\!\!\$ \binom{\{0,1\}^s}{k}$
For $i \in k$:
$\{(M_i, M_i')_{i \in k}\} \leftarrow \mathcal{A}^h(a_1, \ldots, a_k)$
If for all $i \in k$, $M_i \neq M_i'$ and $h(a_i, M_i) = h(a_i, M_i')$
    Return true
Return false
```

Fig. 16: Oracle Game s-mi-k-CR^h capturing multi-instance salted collision-resistance of oracle h

Proof. We construct the following $T_1 + k \cdot T_2$-adversary $\mathcal{B}$ against s-mi-k-CR^h where:

- $\mathcal{B}$ first runs $\mathcal{A}_1^h$.
- It then runs $\mathcal{A}_2^h(a_i)$ for $i \in [k]$, and outputs all the values $\mathcal{A}_2$ outputs.

We next compute a lower bound on $\mathsf{Adv}^{s\text{-}\mathsf{mi}\text{-}k\text{-}\mathsf{CR}}_{H_{s,\ell,n}}(\mathcal{B})$. Let $0 \leqslant \delta \leqslant \epsilon$. Say h is "good" for $\mathcal{A}$ if $\mathcal{A}$ succeeds on finding collisions for at least $\delta \cdot 2^s$ salts for h. We claim that $\epsilon - \delta$ fraction of the h are good. This is true because if α fraction of h are good, we have that

$$\mathsf{Adv}^{\mathsf{CR}}_{H_{s,\ell,n}}(\mathcal{A}) \leqslant \Pr[h \text{ is good}] + \delta\ .$$

Therefore, $\alpha \geqslant \epsilon - \delta$, i.e., $\epsilon - \delta$ fraction of h are good. Observe that $\mathcal{B}$ always wins if h is good for $\mathcal{A}$, and $a_1, \ldots, a_k$ are sampled from the $\delta \cdot 2^s$ salts on which $\mathcal{A}$ wins. Therefore, for $k \leqslant \delta \cdot 2^s$, we have that

$$\begin{aligned}\mathsf{Adv}^{s\text{-}\mathsf{mi}\text{-}k\text{-}\mathsf{CR}}_{H_{s,\ell,n}}(\mathcal{B}) &\geqslant \Pr[h \text{ is good for } \mathcal{A}] \Pr\left[\mathcal{A} \text{ win on } a_1, \ldots, a_k \,\middle|\, h \text{ is good for } \mathcal{A}\right] \\ &\geqslant (\epsilon - \delta) \frac{\binom{\delta \cdot 2^s}{k}}{\binom{2^s}{k}}\end{aligned}$$

□

The hope is to then use the upper-bound on the advantage of any $(T_1 + kT_2)$-adversary against s-mi-k-CR to give an upper bound on the advantage of $\mathcal{A}$ against CR. We show an example where the left hand side and the right hand side of the advantage inequality in Theorem 10 are very far away, meaning we cannot hope for tight-bounds for collision-resistance of salted random oracles using this approach.

Consider a $(2^{n/2}, 0)$-adversary $\mathcal{A}$ that makes $2^{n/2}$ distinct queries for one salt in the offline phase. Let ϵ be the advantage of $\mathcal{A}$. It is east to see that ϵ is at most $1/2^s$.

Theorem 10 requires $k \leqslant \delta \times 2^s \leqslant \epsilon \cdot 2^s$. Since $\epsilon \leqslant 1/2^s$, we have that $k \leqslant 1$. Now note that there exists an adversary $2^{n/2}$ against s-mi-1-CR that succeeds with constant probability. On the other hand, for any choice of δ, the right hand side of the inequality is much smaller than 1. So, we cannot hope for tight bounds from this approach.

However, if we were only interested in adversaries that succeed with probability 1, we would have that the advantage of $\mathcal{B}$ would be 1 (since $\mathcal{A}$ wins on all inputs/random oracles). Then we could upper bound the advantage of $\mathcal{A}$ with the advantage of any $(T_1 + kT_2)$ adversary $\mathcal{B}$ against s-mi-k-CR. We can simply use our analysis in Theorem 2 of the event off-oneblk-k to upper bound this advantage. Namely off-oneblk-k was the probability that the adversary finds one-block collisions for k different salts using T_1 queries, and we showed that $\Pr[\mathsf{off}\text{-}\mathsf{oneblk}\text{-}k] \leqslant \frac{\binom{T_1}{2k}}{2^{kn}}$. Clearly the probability that any adversary finds collisions for salts $a_1, \ldots, a_k$ using $T_1 + kT_2$ queries is at least the probability that the adversary finds collisions for k different salts. Therefore, the advantage of such $\mathcal{B}$ is at most $\frac{\binom{T_1 + kT_2}{2k}}{2^{nk}}$ which is at most $\left(\frac{e(T_1 + kT_2)}{2^{n/2}k}\right)^{2k}$. Since we know $\mathcal{B}$ has advantage 1,we would have that

$$1 \leqslant \left(\frac{e(T_1 + kT_2)}{2^{n/2}k}\right)^{2k}\ .$$

Setting $k = 2^s$ and simplifying we get,

$$\frac{T_1}{2^{s-1}} + T_2 \geqslant \frac{2^{n/2}}{e} \ .$$

Therefore, for advantage 1 adversaries, we get the right guarantee using this approach.