

Security Properties of Iterated Random Functions

Xiaoqi Duan , Kristina Hostáková 

{xiaoqi.duan,kristina.hostakova}@inf.ethz.ch
ETH Zurich, Switzerland.

Abstract. Iteration of a cryptographic hash function is a common practice in many applications, typically used to enhance resistance against various attacks (e.g., to slow down dictionary attacks in password hashing). While the security properties of (non-iterated) cryptographic hash functions are well studied and understood by the cryptographic community, their iterated counterparts have received comparatively little attention. Even in idealized models such as the random oracle model, preimage and collision resistance of an iterated hash function do not seem to be fully characterized.

The security of an iterated random function has mainly been studied through its indistinguishability from the (non-iterated) random function. Bhaumik et al. (ASIACRYPT 2017) analyse the collision resistance of an iterated random function, but in a model where the adversary has oracle access to the iterated random function (and not the random function itself). Kogan et al. (CCS 2017) then study the preimage resistance of an iterated random function in the random oracle model, but focus only on finding preimages for the last iteration step.

In this paper we complete the picture by providing attacks and matching upper bounds for preimage and collision resistance of an iterated random function H^k , where $H: [n] \rightarrow [n]$ is modelled as a random oracle. While collision resistance is essentially unaffected by iterations, we prove that the situation is very different in the case of preimage resistance. Specifically, we present a concrete attack on preimage resistance of H^k with advantage $\Omega(\frac{qk}{n})$ assuming $q = \Omega(k)$, where q denotes the number of H -oracle queries made by the adversary. We complement our attack with an upper bound $\mathcal{O}(\frac{qk+k^2}{n})$, which is tight in the $q = \Omega(k)$ regime. Finally, we show that iteration weakens preimage resistance only for random functions: when H is a permutation, preimage resistance remains essentially unaffected.

1 Introduction

Cryptographic hash functions are fundamental primitives in modern cryptography. They map inputs of arbitrary length to fixed-length outputs while being designed to satisfy properties such as preimage resistance and collision resistance. Hash functions are used across cryptographic systems, including digital signatures (e.g., [35,8,4,26,29]), zero-knowledge proof systems [14], message authentication codes [1], key-derivation functions [21], and play a central role in

practical security systems such as password storage [30,33,7] or blockchain protocols [31,9].

In some of these applications, hash functions are not used in isolation but are instead iterated multiple times to amplify security guarantees or to adapt hash functions to specific threat models. For instance, many password-based schemes use hash iterations to increase the computational cost of dictionary attacks [30,33,7]. Iteration is sometimes also used to avoid structural weaknesses such as length-extension attacks (for instance, the Bitcoin blockchain applies $\text{SHA256}(\text{SHA256}(\text{data}))$ during the Bitcoin mining process [31]). S/KEY system [16] uses hash iterations to produce one-time passwords. Winternitz one-time signature scheme (W-OTS) [26,13] applies direct hash iterations to improve efficiency of Lamport’s one-time signature scheme [25]. The improved variant W-OTS+ [18] is then used in some modern signature schemes including SPHINCS+ [5].

Notably, modern constructions avoid iterating the same hash function many times, and for example use a different salt at every iteration. Indeed, apart from the double-hashing in Bitcoin, direct iterations appear mainly in legacy constructions such as W-OTS [26,13], PBKDF1 (PKCS #5 v2.0 [30]) or S/KEY [16]. Hence, it is well established that iterating the same hash function introduces structural (and hence security) weaknesses. Nevertheless, surprisingly few works have precisely characterized the resulting preimage and collision resistance in this setting and how it compares with the non-iterated case.

Our goal. In this work, we focus on the most basic (and idealized) setting, where we consider a hash function $H: [n] \rightarrow [n]$ modelled as a random oracle, and where by iteration we understand the direct application of H to the input k times, i.e.,

$$H^k(x) = H(H(\dots H(x))).$$

By preimage resistance of H^k we understand the task of finding H^k -preimages, i.e., given $H^k(x)$ for a random x , find some x' such that $H^k(x') = H^k(x)$. And by collision resistance we understand the task of finding H^k collision, i.e., find some $x \neq x'$ such that $H^k(x') = H^k(x)$. In both preimage and collision resistance, we assume the adversary has access to a random oracle H and time complexity of the adversary is measured in the number of H -queries. Our goal is to characterize the preimage and collision resistance of H^k by providing attacks and matching advantage upper bounds, and compare the results to the bound in the non-iterated case.

1.1 Existing work

Iterated Random Function Problem. The security of iterated random permutations/functions has primarily been studied through their indistinguishability from non-iterated random permutations/functions (e.g., [37,36,2,32,3,27,6]). More precisely, in the iterated random function problem, a distinguisher is given oracle access either to a random function or to an *iterated* random function and

must distinguish between the two settings. For the iterated random permutation problem, Minaud and Seurin [27] showed that the distinguishing advantage upper bound using q oracle queries is $\mathcal{O}(\frac{q^k}{n})$ (their bound is almost tight as they provide a matching attack). For the iterated random function problem, Bhaumik et al. [6] showed a corresponding upper bound $\mathcal{O}(\frac{(\log k)^3 q^2 k}{n})$ (which is tight up to polylog factors).

While bounds on collision or preimage resistance of H^k may be helpful when constructing and analyzing distinguishers in the random function problem (see, e.g., [6]), the converse direction appears much less clear. Distinguishers can exploit global structural properties of H^k , such as decrease in image size or cycle statistics, whereas collision and preimage attacks require computing explicit witnesses and therefore exploiting local structure. Consequently, there does not seem to be an obvious black-box reduction translating distinguishing advantage into bounds on collision or preimage resistance, although we are unaware of any formal separation result.

Collision resistance. Collision resistance of H^k has been studied by Bhaumik et al. [6] as a stepping stone towards bounding the distinguishing advantage of the iterated random function problem. They proved an advantage upper bound $\mathcal{O}(\frac{q^{2k}(\log k)^3}{n})$ for any adversary making up to q queries to H^k (and no access to H). Together with the birthday attack, which has advantage $\Omega(\frac{q^2 k}{n})$ (see, e.g., Preneel and van Oorschot [34]), their bound is tight up to polylog factors.

The considered modelling (i.e., giving oracle access to H^k but not H) is well-motivated through the end goal of finding bounds for the iterated random function problem. But if one looks at the question of collision resistance of H^k on its own, it is less clear why the adversary should have access to the iterate H^k and not the random function H itself. Recall that one of the reasons why hash functions are often iterated in applications is to increase the computational cost for brute-force attacks. Hence, we believe that it is more natural to study the security properties of iterated hash functions in a model, where the cost of computing $H^k(x)$ is k -times more expensive than a single computation $H(x)$.

Preimage resistance. Kogan et al. [24] (based on ideas from [17]) have studied preimage resistance of H^k in a model where the adversary has oracle access to a random oracle H . However, their notion of preimage resistance is slightly different than ours. Concretely, they study the problem of finding an H -preimage of $H^k(x)$ for a randomly chosen x , i.e., finding x' such that $H(x') = H^k(x)$. For this problem, they provide a concrete attack with advantage $\Omega(\frac{q^k}{n})$, where q denotes the number of H queries. They assume $k \leq \sqrt{n}$ and $2k \leq q \leq \frac{n}{k}$. Their result demonstrates a loss in security by a multiplicative factor k compared to the non-iterated case and motivates them to study the security of hash-chains when each iteration is performed with a different random function.

Let us stress the difference between H -preimage resistance and H^k -preimage resistance of iterated random functions on a concrete example. Consider a password hashing scheme in which a password pw is iteratively hashed k times to

produce a public commitment $y = H^k(pw)$. A formal security proof for hash-chain-based constructions such as S/KEY therefore requires arguing about the hardness of recovering the initial password pw from the public commitment y , i.e., establishing the H^k -preimage resistance of the iterated hash function. Note that H -preimage resistance would correspond to finding intermediate values in the hash-chain and not the initial password.

It is not clear how to translate the bound on H -preimage resistance of H^k to a bound on H^k -preimage resistance of H^k in a black-box way. Note that for x' such that $H(x') = H^k(x)$, we have no guarantee that $x' = H^{k-1}(x)$. In fact, there might not even exist a H^{k-1} -preimage of x' . Instead of a black-box reduction, one may therefore want to study the concrete attack on H -preimage resistance that the authors present and see whether it can be extended to find H^k -preimages. As we argue in the technical overview, this is unfortunately also not the case. Intuitively, the way the attacker searches for H -preimages does not provide any guarantee of being able to “trace back” $k - 1$ steps and hence also find a H^k -preimage.

1.2 Our results

Let us assume an adversary that can make up to q queries to a random oracle H . As a main result of this work, we provide a concrete attack on H^k -preimage resistance of H^k with advantage lower bounded by $\Omega(\frac{qk}{n})$ assuming $q \geq 3k$. Additionally, we prove an advantage upper bound of $O(\frac{qk+k^2}{n})$, which matches the lower bound in the $q = \Omega(k)$ regime. To complete the picture, we also study attacks and upper bounds on collision resistance of H^k , and the preimage resistance of H^k for the special case of H being a random permutation. Our asymptotic bounds, on which we comment below, are summarized in Table 1.

	CR	PR	PR-P
Attack Advantage	$\Omega\left(\frac{q^2}{n}\right)$	$\Omega\left(\frac{qk}{n}\right)$, for $q \geq 3k$	$\Omega\left(\frac{q}{n}\right)$, for $q \geq 2k$
Security Bound	$O\left(\frac{(q+k)^2}{n}\right)$	$O\left(\frac{qk+k^2}{n}\right)$	$O\left(\frac{q+k}{n}\right)$

Table 1. Asymptotic bounds for: (PR) finding a H^k preimage, cf. Theorem 1 and Theorem 2; (CR) finding an H^k collision, cf. Theorem 8 and Theorem 9; and (PR-P) finding an H^k preimage when H is a permutation, cf. Theorem 6 and Theorem 7.

Let us briefly interpret and comment on the bounds presented in Table 1. For collision resistance, both the iterated hash function H^k and the non-iterated hash function H share the same attack advantage and security bound $\mathcal{O}(\frac{q^2}{n})$, when $q = \Omega(k)$. The reason is two-fold: On the one hand, any collision for H

leads to a collision for H^k . On the other hand, from a collision for H^k , we can also construct a collision for H (with slightly more oracle queries, see Section 5.2 for details). As a result, iterating hash functions does not help or harm the security of collision resistance (as long as $q = \Omega(k)$).

The situation is very different in the preimage resistance game. In the non-iterated case, it is known that no attack can achieve advantage better than $\mathcal{O}(\frac{q}{n})$ (see, e.g., [19,28]). However, for the iterated function H^k , we put forward an attack with advantage $\Omega(\frac{qk}{n})$ (see Section 3.1 for details). This means iterations reduce (both H and H^k) preimage resistance of random functions by a multiplicative factor k .

Finally, we consider the special case of H being a random permutation, and show that the security gap between non-iterated and iterated settings disappears. One interpretation of this result is that if H is a permutation, then so is H^k . Specifically, for any $y \in [n]$, there exists exactly one preimage x satisfying $H^k(x) = y$. Therefore, the adversary should not gain an advantage compared to the non-iterated version H .

1.3 Technical Overview

Let us elaborate on how we achieve our results stated in the previous section. Since preimage resistance of an iterated hash function yields the most interesting results, we mainly focus on this property here and discuss the remaining two results briefly at the end of this technical overview.

Attack on H -preimage resistance. Let us start by explaining the attack presented by Kogan et al. [24, Theorem 4.1] on H -preimage resistance of H^k . On input $y = H^k(x)$, their adversary defines $x_0 := y$ and computes $x_j = H(x_{j-1})$ until either **Case 1:** $x_j = y$ or **Case 2:** $x_j = x_i$ for some $1 \leq i < j$. In case 2, the adversary samples a fresh x_{j+1} and continues hashing as before.

If such an adversary succeeds without ever applying case 2 (i.e., the adversary finds a cycle that includes y), this adversary then also finds an H^k preimage of y by tracing k steps back and hence the attack translates to our setting. The situation is more complicated when the adversary has to apply case 2 at least once. This means that an H -preimage is found through a chain of ℓ hash queries such that $H^\ell(x') = y$ for some x' never seen before (and in particular $x' \neq y$). Obviously, an H^k -preimage of y is then found only if $\ell \geq k$.

This motivates our new attack on H^k -preimage resistance of H^k , where we want to ensure that y is always found in a chain of hash queries of length at least k , which significantly simplifies the probability calculations.

Our attack for the preimage resistance game. On input $y \in [n]$, our adversary $\mathcal{A}$ samples a random $x_0 \leftarrow_{\$} [n]$ and iteratively queries the random oracle H $3k$ -times. That is, for $i \in [3k]$, the adversary learns $x_i = H(x_{i-1})$. The hope of the adversary is that one of the x_i values actually equals the challenge value y . If this is the case and the adversary is able to trace k steps back, they find an H^k -preimage of y . More precisely, if there exists $i^* \in [k, 3k]$ such that $x_{i^*} = y$,

the adversary outputs x_{i^*-k} as an H^k -preimage of y . The adversary repeats this process $r = q/3k$ times, where q is the total number of oracle queries (for the purpose of this technical overview, we assume q is a multiple of $3k$). We call a sequence of $3k$ oracle calls described above an *attack round*. Our attack hence consists of r attack rounds.

In order to analyse the advantage of $\mathcal{A}$, we split the $3k$ oracle queries made in each attack round into three sets, each of size k . The first k queries are $\{x_0, \dots, x_{k-1}\}$, the middle k queries are $\{x_k, \dots, x_{2k-1}\}$, and the last k queries are $\{x_{2k}, \dots, x_{3k-1}\}$, see Figure 1 for an example. Let x be the random value sampled by the challenger to generate $y = H^k(x)$.

It is not difficult to see that if *any* of the k middle queries equals *any* of the following values

$$x, H(x), H^2(x), \dots, H^k(x), \quad (1)$$

our adversary finds a preimage of $y = H^k(x)$ successfully.¹ This is because (i) there are at least k preceding queries (indeed, even if $x_k = y$, the adversary can still trace k steps back and output x_0); and (ii) there are at least k follow-up queries and hence the adversary reaches y (indeed, even if $x_{2k-1} = x$, the adversary makes sufficiently many queries to reach y , i.e., $H(x_{3k-1}) = y$). Let us denote by E_i the event that one of the middle queries equals a value in Equation (1) in attack round $i \in [r]$.

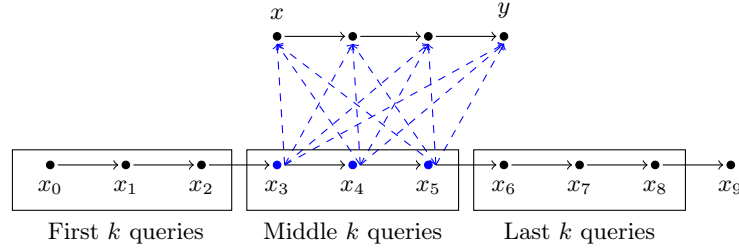


Fig. 1. This example depicts one round of queries of $\mathcal{A}$ when $k = 3$. $\mathcal{A}$ makes 9 queries $x_0, x_1, \dots, x_8$ in this round. As long as the response of any of the middle part queries x_3, x_4, x_5 equals any of the points in the computation of $H^k(x) = y$ (all the 12 possibilities have been indicated by blue dashed arrows), the adversary will reach y in this query round, and is able to trace back and obtain a preimage of y .

Hence, in order to find a lower bound for the advantage of our adversary, we study the probability of the event E_i .

For the purpose of this technical overview, assume all the $k + 1$ values in Equation (1) are different, and that our chain of $3k$ queries will not run into

¹ Note that the adversary might find a preimage in other ways too, but we do not need to capture all possibilities as we are interested in a lower bound on the advantage.

loops. Now since there are k queries in the middle part, we have k chances to hit any of the $k + 1$ values in Equation (1). Since H is a random oracle, we have $\Pr[E_i] \geq \frac{k^2}{n}$. For simplicity, assume independence of the events $\{E_i\}_{i \in [r]}$. Then we can bound the advantage of our adversary as follows:

$$\text{Adv}(\mathcal{A}) \geq \Pr\left[\bigvee_{i \in [r]} E_i\right] = 1 - \prod_{i \in [r]} \Pr[\overline{E_i}] \geq \left(1 - \left(1 - \frac{k^2}{n}\right)^r\right) \geq \frac{k^2 \cdot r}{2n} = \frac{kq}{6n}.$$

Let us stress that the actual analysis presented in Section 3.1 is more involved as this rough sketch makes a number of simplifying assumptions and omits some details. For instance, we assume here that there are no loops in the queries of the adversary, or assume independence of the event E_i . The purpose of this simplified sketch is to provide some intuition why we reach the $\Omega(\frac{kq}{n})$ bound.

Advantage upper bound. We show that the advantage of any adversary making up to q queries to the random oracle H in the preimage resistance game is $\mathcal{O}(\frac{kq}{n})$. The main idea is to utilize the incompressibility of random oracles; concretely, the compression lemma from [10]. At a high level, the compression lemma (formally recalled in Lemma 2) says that, if a pair of encoding and decoding algorithms can compress a random function by at least $\log 1/\epsilon$ bits, then correctness of decoding can hold with probability at most ϵ .

Our goal is to design a pair of encoding and decoding algorithms (Enc, Dec) which compress the random oracle $H: [n] \rightarrow [n]$. Since H can implement any random function from $[n]$ to $[n]$, we need $n \log n$ bits to store the whole function table of H . However, suppose we have an adversary $\mathcal{A}$ that can win the preimage resistance game with high probability. We then show that we can utilize $\mathcal{A}$ to compress H . Namely, we design encoding and decoding algorithms, such that as long as $\mathcal{A}$ wins the game, our encoding algorithm can output fewer than $n \log n$ bits, while our decoding algorithm can still correctly restore H .

An adversary $\mathcal{A}$ on input $y = H^k(x)$ wins the preimage resistance game if they find some x' such that $H^k(x') = y$. We first argue that the probability that $x' = x$ is low and focus on the case when $x' \neq x$. We argue that in this case we can find an H -collision in the random oracle queries made. That is, there are two different random oracle queries z_i, z_j ($i < j$) such that $H(z_i) = H(z_j)$. The two oracle queries can either be made by $\mathcal{A}$, or by the challenger when computing $y := H^k(x)$ or verifying $H^k(x') = y$. Now, we do *not* store the value $H(z_j)$ in the function table of H ; this would normally require $\log n$ bits. Instead, we store two indices i and j which require fewer than $\log n$ bits. When our decoding algorithm tries to recover the value $H(z_j)$, it uses the indices and the value $H(z_i)$ instead.

To conclude, we can use fewer bits to store the function table of H (i.e. compress H). Now, the compression theorem gives a probability upper bound for such an event, which also gives an upper bound for the advantage of $\mathcal{A}$ in the preimage resistance game.

Other results. For the special case that H is a permutation, our attack is similar to the attack for H being a random oracle. We also sample a random $x_0 \leftarrow_{\$}$

$[n]$ and iteratively query H on x_0 . However, now we only obtain advantage $\Omega\left(\frac{q}{n}\right)$ due to properties of permutations. In particular, the only way for the adversary to win the preimage resistance game is to find the exact value x sampled by the challenger. For the advantage upper bound, we use the properties of a random permutation, and, through a game hopping proof, show that no adversary can achieve advantage $\Omega\left(\frac{q+k}{n}\right)$ in this new game. The details can be found in Section 4.1 and Section 4.2.

For the collision resistance property, our attack is very similar to the traditional birthday attack. The intuition is that a collision (x_1, x_2) for H (where $H(x_1) = H(x_2)$) is essentially a collision for H^k , because we have $H^k(x_1) = H^{k-1}(H(x_1)) = H^{k-1}(H(x_2)) = H^k(x_2)$. For the advantage upper bound, we again utilize the incompressibility of random oracles, and design encoding and decoding algorithms to compress H . Specifically, if an adversary $\mathcal{A}$ finds a collision x_1, x_2 where $H^k(x_1) = H^k(x_2)$, we can again find a H -collision and use fewer bits to store the collision point. The details can be found in Section 5.1 and Section 5.2.

1.4 More on related work

From the long line of works focusing on the iterated random permutation/function problem (e.g., [37, 36, 2, 32, 3, 27]), the work most closely related to ours is that of Bhaumik et al. [6]. As already discussed, they provide an upper bound $\mathcal{O}\left(\frac{q^2 k (\log k)^3}{n}\right)$ on collision resistance of an iterated random function H^k . One may wonder why the bound is different than what we present here. The crucial difference is in the considered model, as Bhaumik et al. give the adversary oracle access to H^k (and not H), while we consider the setting in which the adversary gets oracle access to H (and not H^k). The difference in modelling also explains the difference in the provided bounds.

Kogan et al. [24] study the security of hash-chains in the context of time-based one-time password system called T/Key (extending the classical S/KEY [16]). In addition to the already mentioned H -preimage resistance of H^k , they also study the preimage-resistance of a hash-chain of length k , where every iteration is performed with a different random function. In this context, they prove that an adversary that has oracle access to all random functions used to compute the hash-chain and making q oracle queries in total has advantage of finding a preimage of the last iterate of at most $\frac{2q+3}{n}$, and of the whole hash-chain at most $\frac{2q+2k+1}{n}$.

The Merkle-Damgård construction also utilizes the idea of iterating hash functions, and there have been many works studying its security properties. Specifically, many attacks have been proposed, including the herding attack [22], the multi-collision attack [20], and the second-preimage attack [23]. The key intuition behind these attacks is that the adversary can partially control the intermediate results in the Merkle-Damgård construction by carefully choosing the message for each input block. However, for the iterated hash function H^k , the adversary can only provide one input message at the beginning, and is not

able to control the intermediate results. As a result, all these attacks become invalid in the H^k setting.

Security of hash functions is also analysed through their indifferentiability from a random function. At a high level, the indifferentiability notion requires a hash function construction h^f that is built from an ideal primitive f to be indistinguishable from a random oracle R , even if the adversary has oracle access to the ideal primitive f . This notion is formalized through a real/ideal world paradigm by requiring the existence of a simulator S^R such that iteration with h^f , f is indistinguishable from iteration with R , S^R . It has been shown by Dodis et al. [12] that iterated random functions perform poorly in terms of indifferentiability. Intuitively, this is because in the real world, the adversary interacts with H^k and H for a random function H . And in the ideal world, the simulator needs to interpret outputs of a random oracle R as k -th iterates of a random function.

2 Definitions and Preliminaries

2.1 Notations

Let $[n]$ denote the set $\{1, 2, \dots, n\}$. We write $x \leftarrow_{\$} [n]$ to indicate that x is sampled uniformly at random from the set $[n]$. For two pairs of integers (i, j) and (i', j') , we write $(i, j) < (i', j')$, if either (1) $i < i'$, or (2) $i = i'$ but $j < j'$. Similarly, we write $(i, j) \leq (i', j')$ if either (1) $i < i'$, or (2) $i = i'$ but $j \leq j'$.

We work in the random oracle model, where the adversary gets *oracle access* to a random function H (which is sampled uniformly from all possible functions mapping $[n]$ to $[n]$). Our adversaries are constrained by the number of H -oracle queries q , but otherwise *computationally unbounded*.

2.2 Security properties of iterated random functions

We now formally define preimage and collision resistance of an iterated random function H^k . We start with the preimage resistance experiment against H^k .

Definition 1. For a fixed function $H : [n] \rightarrow [n]$ and integer parameters k, q , for any adversary $\mathcal{A}$ that makes at most q oracle queries to H , the preimage resistance game for H^k , denoted by $PR^{k,q,H}(\mathcal{A})$, is defined in Figure 2:

The advantage of $\mathcal{A}$ in this game, denoted by $Adv_{k,q,H}^{PR}(\mathcal{A})$, is defined as the probability that the game $PR^{k,q,H}(\mathcal{A})$ returns 1, where the probability is taken over the choice of x and the randomness of $\mathcal{A}$.

Now we define the advantage of an adversary $\mathcal{A}$ when the hash function H is sampled randomly from some family. Specifically, we consider two different cases, where H is modeled as a random function (respectively a random permutation):

Game $\text{PR}^{k,q,H}(\mathcal{A})$	
1 :	$x \leftarrow_{\$} [n]$
2 :	$y \leftarrow H^k(x)$
3 :	$x' \leftarrow \mathcal{A}^H(k, q, n, y)$
4 :	if $(H^k(x') = y)$
5 :	return 1
6 :	else return 0

Fig. 2. Security game $\text{PR}^{k,q,H}(\mathcal{A})$. The adversary $\mathcal{A}$ is given parameters k, q, n and the challenge y as input, and can make at most q oracle queries to H . The computational power of $\mathcal{A}$ is unbounded.

Definition 2 (Preimage resistance for iterated random functions). *The advantage $\text{Adv}_{k,q,H}^{\text{PR-RAND}}(\mathcal{A})$ is defined as the probability that the preimage resistance game $\text{PR}^{k,q,H}(\mathcal{A})$ returns 1, where H is sampled uniformly at random from all functions mapping $[n]$ to $[n]$.*

Definition 3 (Preimage resistance for iterated random permutations). *The advantage $\text{Adv}_{k,q,H}^{\text{PR-PERM}}(\mathcal{A})$ is defined as the probability that the preimage resistance game $\text{PR}^{k,q,H}(\mathcal{A})$ returns 1, where H is sampled uniformly at random from all permutations mapping $[n]$ to $[n]$.*

Similarly, we can define the collision resistance game for H^k .

Definition 4. *For a fixed function $H : [n] \rightarrow [n]$ and integer parameters k, q , for any adversary $\mathcal{A}$ that makes at most q oracle queries to H , the collision resistance game for H^k , denoted by $\text{CR}^{k,q,H}(\mathcal{A})$, is defined in Figure 3:*

Game $\text{CR}^{k,q,H}(\mathcal{A})$	
1 :	$x_1, x_2 \leftarrow \mathcal{A}^H(k, q, n)$
2 :	if $(x_1 \neq x_2 \wedge H^k(x_1) = H^k(x_2))$
3 :	return 1
4 :	else return 0

Fig. 3. Security game $\text{CR}^{k,q,H}(\mathcal{A})$. The adversary $\mathcal{A}$ is given parameters k, q, n as input, and can make at most q oracle queries to H . The computational power of $\mathcal{A}$ is unbounded.

Since a permutation does not contain collisions, for the advantage, we focus on the case that H is a random function here:

Definition 5 (Collision resistance of iterated random functions). *The advantage $\text{Adv}_{k,q,H}^{\text{CR}}(\mathcal{A})$ is defined as the probability that the game $\text{CR}_{k,q,H}(\mathcal{A})$ returns 1, where H is sampled uniformly at random from all functions mapping $[n]$ to $[n]$.*

2.3 Miscellaneous

During the analysis, it will be helpful to consider the functional graph [15] of H . Specifically, we treat $H : [n] \rightarrow [n]$ as a graph with n points indexed by $1, 2, \dots, n$, where an edge $x \rightarrow y$ exists if and only if $H(x) = y$. Now we can adapt the concept of a “chain” from graph theory:

Definition 6 (Chain). *Given a hash function $H : [n] \rightarrow [n]$ and $x, l \in [n]$, the points $x, H(x), H^2(x), \dots, H^l(x)$ form the length- l chain starting from x .*

Finally, the following are some lemmas we will use in our proofs. The first is a folklore analytic bound:

Lemma 1. *For any $0 \leq x_1, \dots, x_m \leq 1$, we have $\prod_{i=1}^m (1 - x_i) \geq 1 - \sum_{i=1}^m x_i$.*

Proof. Consider m independent Bernoulli random variables $X_1, \dots, X_m$ where $\Pr[X_i = 1] = x_i$ and $\Pr[X_i = 0] = 1 - x_i$. Consider event E : $X_1 = X_2 = \dots = X_m = 0$. We have $\Pr[E] = \prod_{i=1}^m (1 - x_i)$. On the other hand, consider the complementary event $\bar{E}$ (i.e. there exists some i such that $X_i = 1$). From the union bound we know $\Pr[\bar{E}] \leq \sum_{i=1}^m \Pr[X_i = 1] = \sum_{i=1}^m x_i$, so $\Pr[E] = 1 - \Pr[\bar{E}] \geq 1 - \sum_{i=1}^m x_i$, which finishes the proof. $\square$

When $x_1 = \dots = x_m = x$, we have the following corollary:

Corollary 1. *For any $0 \leq x \leq 1$ and integer $m \geq 1$, we have $(1 - x)^m \geq 1 - xm$.*

The compression theorem (originated from [10]) states the impossibility of compressing a random source. Intuitively, if there exists a pair of encoding and decoding algorithms (Enc, Dec) where Enc can compress the input length significantly, then the probability that Dec can recover the input will be relatively small. We will apply it to ROM outputs, and use the following lemma in the proof of our security bounds:

Lemma 2 ([11]). *Consider any pair (Enc, Dec) of encoding and decoding algorithms, where $\text{Enc} : \{0, 1\}^x \rightarrow \{0, 1\}^y$ and $\text{Dec} : \{0, 1\}^y \rightarrow \{0, 1\}^x$. If the recovery probability is at least ϵ , then $y \geq x - \log \frac{1}{\epsilon}$.*

3 Preimage Resistance Results for H^k in the Random Oracle Model

The goal of this section is to study the $\text{PR}^{k,q,H}(\mathcal{A})$ game, and look at both a concrete attack and the security bounds for this game. In Section 3.1, we give an adversary $\mathcal{A}$ that achieves advantage $\Omega(\frac{qk}{n})$ in the $\text{PR}^{k,q,H}(\mathcal{A})$ game. In

Section 3.2, we give a proof that no adversary $\mathcal{A}$ can achieve advantage better than $\mathcal{O}(\frac{qk}{n})$ in this game. The intuition of both the attack and the upper bound was provided in Section 1.3.

Throughout this section, let $H : [n] \rightarrow [n]$ be a random oracle, and k, q be the given parameters for the preimage resistance game. For convenience, suppose k, q, n are all powers of 2. (If not, simply round up k, q, n to the closest power of 2. After rounding, all of the bounds will only be influenced by a constant factor.)

3.1 Attack

Theorem 1. *Assume $q \geq 3k$ and $\frac{qk}{n} \leq \frac{1}{8}$. There is an adversary $\mathcal{A}$ that achieves advantage $\text{Adv}_{k,q}^{\text{PR-RAND}}(\mathcal{A}) = \Omega(\frac{qk}{n})$.*

We first introduce some notation. For $0 \leq i \leq k$, let $x_i = H^i(x)$ where x is the randomly sampled value by the challenger at the beginning of the $\text{PR}^{k,q,H}(\mathcal{A})$ game (notice that $x_k = H^k(x) = y$, and similarly $x_0 = H^0(x) = x$). Now the points $x_0, x_1, \dots, x_k$ will form a length- k chain starting from x_0 . Let $S_0 = \{x_0, x_1, \dots, x_k\}$ denote the set of points in this chain.

Now we give an overview of our attack. The attack is divided into r rounds with $3k$ queries in each round. (The intuition of the parameter $3k$ has been explained in Section 1.3). In the i_{th} round, we start by picking a random point $x_{i,0}$, and query out the length- $3k$ chain starting from $x_{i,0}$, namely $x_{i,1} = H(x_{i,0}), x_{i,2} = H(x_{i,1}), \dots, x_{i,3k} = H(x_{i,3k-1})$. Our target will be to hit y in the chain. If we find y somewhere in the chain, say $x_{i,j} = y$, then we can trace back k steps and find a preimage of y , namely $x_{i,j-k}$ (as long as $j \geq k$), because we know $H^k(x_{i,j-k}) = x_{i,j} = y$ from our construction of the chain.

Proof. Our constructed adversary $\mathcal{A}(k, q, n, y)$ is depicted in Figure 4.

$\mathcal{A}(k, q, n, y)$	
1 :	$r \leftarrow \lfloor \frac{q}{3k} \rfloor, z \leftarrow \perp$
2 :	for $i \in [1, r]$
3 :	$x_{i,0} \leftarrow_{\$} [n]$
4 :	for $j \in [1, 3k]$
5 :	$x_{i,j} \leftarrow H(x_{i,j-1})$
6 :	if $(x_{i,j} = y \wedge j \geq k)$
7 :	$z \leftarrow x_{i,j-k}$
8 :	return z

Fig. 4. Our adversary $\mathcal{A}(k, q, n, y)$ breaking preimage resistance of H^k where H is a random oracle.

For convenience, we denote by a “round” of $\mathcal{A}$ each execution of the loop in lines 3-7. $\mathcal{A}$ will execute r rounds before returning some preimage $z = x_{i,j}$ or the failure symbol $\perp$.

First, if the adversary returns some $z = x_{i,j}$, we claim that this $x_{i,j}$ will be a preimage of y . This is because according to line 5, we have $H^k(x_{i,j}) = H^{k-1}(x_{i,j+1}) = \dots = x_{i,j+k}$. Further, from the check in line 6 we know $x_{i,j+k} = y$. Therefore, $x_{i,j}$ is indeed a preimage of y .

Now we look at the winning probability of our adversary. Recall that $S_0 = \{x_0, x_1, \dots, x_k\}$ is the length- k chain starting from x_0 . Denote by G the event: $\forall 0 \leq i < j \leq k, x_i \neq x_j$ (i.e. all points in set S_0 are different, or equivalently $|S_0| = k+1$). We first show that G will happen with good probability, then show that our adversary can achieve good advantage given that G happens.

Lemma 3. *Let G be the event defined above. We have $\Pr[G] \geq \frac{1}{2}$.*

Proof. To help the analysis, denote by G_i the event: $\forall j < i, x_i \neq x_j$, which says that x_i does not coincide with any of the previous points $x_1, \dots, x_{i-1}$. Then G happens if and only if all the events G_i ($1 \leq i \leq k$) happen. Hence we have:

$$\begin{aligned} \Pr[G] &= \Pr[G_1 \wedge \dots \wedge G_k] = \prod_{i=1}^k \Pr[G_i | G_1 \wedge \dots \wedge G_{i-1}] = \prod_{i=1}^k \left(1 - \frac{i}{n}\right) \\ &\stackrel{\text{L.1}}{\geq} 1 - \sum_{i=1}^k \frac{i}{n} = 1 - \frac{k(k+1)}{2n} \geq \frac{1}{2}. \end{aligned}$$

The third equality is due to the lazy sampling property of H . Since x_i is the response for the query x_{i-1} to H , given events $G_1 \wedge \dots \wedge G_{i-1}$, we know that x_i should be sampled uniformly at random from $[n]$. Therefore, given events $G_1 \wedge \dots \wedge G_{i-1}$, the event G_i should happen with probability $(1 - \frac{i}{n})$. The inequality is from Lemma 1, and the last inequality is since $\frac{k^2}{n} \leq \frac{qk}{3n} \leq \frac{1}{24} \leq \frac{1}{2}$ from our assumption. $\square$

Now we focus on the advantage of our adversary given event G . We assume event G happens in the rest of the proof. To help the proof, we first give the definition of a “fresh” query, which literally says that neither the challenger nor the adversary has made the same query before:

Definition 7 (Fresh queries). *We call a query $x_{i,j}$ to be **fresh**, if both the following conditions hold:*

1. $x_{i,j} \notin S_0$.
2. $x_{i,j} \neq x_{i',j'}$ for $(i',j') < (i,j)$.

Now we give some intuition of our analysis. If the adversary returns some $z = x_{i,j}$, then the length- k chain starting from $x_{i,j}$ will collide with the chain S_0 at some point (this is because we know $x_{i,j+k} = y$ and $y \in S_0$, the length- k chain will always collide with S_0). Therefore, we want to focus on the first point of the chain that collides with S_0 . Specifically, we want to find some pair (i,j) that satisfies the following requirements:

Definition 8 (Event $E_{i,j}$). For a pair (i, j) , denote by $E_{i,j}$ the event that all of the following conditions are satisfied:

1. The queries $x_{i,0}, \dots, x_{i,j-1}$, are all **fresh**. (It also means that none of them coincide with the chain S_0 , because otherwise they will not be fresh.)
2. The response $x_{i,j} = H(x_{i,j-1})$ of the query $x_{i,j-1}$ should coincide with some point in S_0 .
3. We also require $k < j \leq 2k$ (i.e. we require the collision point to be in the middle part of the $3k$ queries).

We first claim that, if $E_{i,j}$ happens, then in the i_{th} round the adversary will always find some $x_{i,j'}$ that is a preimage of y :

Claim 1. Given that $E_{i,j}$ happens, there exists $0 \leq j' \leq k$ such that $x_{i,j-j'}$ will be stored in line 7 (as a preimage of y).

Proof. Since $E_{i,j}$ happens, according to condition 2 (i.e. $x_{i,j}$ coincides with S_0), there exists $0 \leq j' \leq k$ s.t. $x_{j'} = x_{i,j}$. Since $k < j \leq 2k$ (by condition 3), we have $0 < j - j' \leq 2k$ and $k < j - j' + k \leq 3k$, so the adversary must have encountered both $x_{i,j-j'}$ and $x_{i,j-j'+k}$.

Now we only need to show $x_{i,j-j'+k} = y$. Since $x_{i,j} = x_{j'}$, we know that

$$x_{i,j-j'+k} = x_{i,j+(k-j')} = x_{j'+(k-j')} = x_k = y. \quad \square$$

Figure 5 gives an example of the relationship between event $E_{i,j}$ and the obtained preimage $x_{i,j-j'}$.

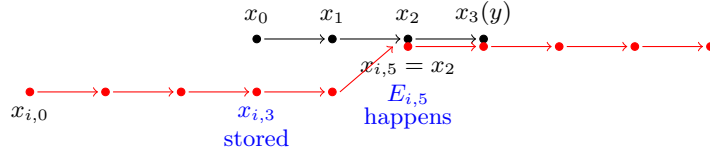


Fig. 5. An example of the relationship between $E_{i,j}$ and the stored preimage $x_{i,j-j'}$. Here we have $k = 3, j = 5$ and $j' = 2$. The event $E_{i,5}$ happens because $x_{i,5} = H(x_{i,4})$ is the first point that coincides with S_0 . Specifically, it collides with x_2 so we have $j' = 2$, and this indicates $x_{i,j-j'} = x_{i,3}$ to be a valid preimage.

As a result, if the adversary triggers some $E_{i,j}$, then it will definitely win the $\text{PR}^{k,q,H}(\mathcal{A})$ game. To give a bound for its advantage, we focus on the probability that **no** event $E_{i,j}$ is triggered for all pairs (i, j) . Specifically:

Lemma 4. Let $\overline{E_{i,j}}$ be the complementary event of the event $E_{i,j}$. We have

$$\Pr \left[\left(\bigwedge_{(i,j)} \overline{E_{i,j}} \right) \middle| G \right] \leq e^{-\frac{qk}{12n}}.$$

Proof. We begin with an observation. Throughout the attack, it is possible that the event $E_{i,j}$ happens for more than one pair of (i, j) . However, in each round i there can be at most one index j that triggers $E_{i,j}$:

Claim 2. For each i , the event $E_{i,j}$ can happen for at most one index j .

Proof. Suppose there are $j' < j''$ such that both $E_{i,j'}$ and $E_{i,j''}$ happen. Then we know $x_{i,j'} \in S_0$ by condition 2. However, this will imply that the query $x_{i,j'}$ is not fresh. Since $j' < j''$, this contradicts condition 1 for event $E_{i,j''}$. $\square$

Now, we want to know what the requirement is for some event $E_{i,j}$ to be triggered. Specifically, we fix some $1 \leq i \leq r$ and $k < j \leq 2k$, and look at the probability that (i, j) is the first pair to trigger event $E_{i,j}$. (Notice that $E_{i,j}$ can only happen if $k < j \leq 2k$, so we do not need to consider other values of j .) To make sure (i, j) is the first pair, we have the following requirements:

1. No event $E_{i',j'}$ happens for some $i' < i$ (i.e. no event $E_{i',j'}$ happens in the previous $i - 1$ rounds).
2. Event $E_{i,j}$ happens.

(Notice that from Claim 2, as long as $E_{i,j}$ happens, no event $E_{i,j'}$ can happen for any $j' < j$ in the i_{th} round.)

Now, fix $1 \leq i \leq r$ and $k < j \leq 2k$, and we focus on the conditional probability that $E_{i,j}$ happens given the first requirement (and also the event G). We will later use the chain rule to obtain the final results. In short, we want to look at the conditional probability $\Pr[E_{i,j} | G \wedge (\bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}})]$. To this end, we only need to verify the first two conditions for $E_{i,j}$ according to Definition 8 (Notice that condition 3 is already satisfied by our choice of j):

1. First, for $j' < j$, all the previous queries $x_{i,j'}$ in the i_{th} round should be fresh. It means that
 - (a) The randomly sampled $x_{i,0}$ should not coincide with S_0 or any previous queries $x_{i'',j''}$ where $i'' < i$. Given event G , we know $|S_0| = k + 1$. Further, the adversary can make at most q queries in total. Therefore, this happens with probability at least $1 - \frac{q+k+1}{n}$.
 - (b) For $0 \leq j' < j - 1$ in increasing order, given that all the previous queries $x_{i,0}, \dots, x_{i,j'}$ are fresh, we want that the next query $x_{i,j'+1} = H(x_{i,j'})$ is also fresh. Notice that $x_{i,j'+1} = H(x_{i,j'})$ is the response of a fresh query $x_{i,j'}$. Therefore, this response should not coincide with S_0 or any previous queries $x_{i'',j''}$ where $(i'', j'') \leq (i, j')$. (This includes the queries in previous $i - 1$ rounds, as well as the previous queries $x_{i,0}, \dots, x_{i,j'}$ in the i_{th} round.) Notice that due to the lazy sampling property of H , the response $H(x_{i,j'})$ is sampled randomly from $[n]$. Therefore, with the same reasoning, this happens with probability at least $1 - \frac{q+k+1}{n}$.
2. Next, given that $x_{i,j-1}$ is a fresh query, the response $x_{i,j} = H(x_{i,j-1})$ should be in S_0 . Given event G , we know $|S_0| = k + 1$. Therefore, according to the lazy sampling property of H , this happens with probability $\frac{k+1}{n}$.

Combining these conditions, for any fixed $1 \leq i \leq r$ and $k < j \leq 2k$, the following holds:

$$\begin{aligned} \Pr \left[E_{i,j} \middle| G \wedge \left(\bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \right) \right] &\geq \frac{k+1}{n} \cdot \left(1 - \frac{q+k+1}{n} \right)^j \\ &\geq \frac{k}{n} \left(1 - \frac{q+2k}{n} \right)^{2k} \\ &\geq \frac{k}{n} \left(1 - \frac{2k(q+2k)}{n} \right) \geq \frac{k}{2n}. \end{aligned}$$

The first line is from the above analysis, the second line is since $q+k+1 \leq q+2k$ and $j' \leq j \leq 2k$, the last line is from Corollary 1 and the assumption $\frac{qk}{n} \leq \frac{1}{8}$ and $\frac{k^2}{n} \leq \frac{qk}{3n} \leq \frac{1}{24}$.

Now, according to Claim 2 (i.e. at most one j can trigger $E_{i,j}$), for any fixed i , all the events $E_{i,k+1}, E_{i,k+2}, \dots, E_{i,2k}$ are mutually exclusive. Hence we have:

$$\begin{aligned} \Pr \left[\exists j : E_{i,j} \middle| \bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \wedge G \right] &= \Pr \left[\bigvee_{k < j \leq 2k} E_{i,j} \middle| \bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \wedge G \right] \\ &= \sum_{j=k+1}^{2k} \Pr \left[E_{i,j} \middle| \bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \wedge G \right] \\ &\geq k \cdot \frac{k}{2n} = \frac{k^2}{2n}. \end{aligned}$$

The inequality is directly from the bound above. And therefore, given i , the probability that **no** event $E_{i,j}$ is triggered in round i will be:

$$\Pr \left[\bigwedge_j \overline{E_{i,j}} \middle| \left(\bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \right) \wedge G \right] \leq 1 - \frac{k^2}{2n}.$$

Finally, according to the chain rule, the probability that **no** event $E_{i,j}$ is triggered, for all rounds $1 \leq i \leq r$, will be:

$$\Pr \left[\bigwedge_{(i,j)} \overline{E_{i,j}} \middle| G \right] = \prod_{i=1}^r \Pr \left[\bigwedge_j \overline{E_{i,j}} \middle| \bigwedge_{(i',j'): i' < i} \overline{E_{i',j'}} \wedge G \right] \leq \left(1 - \frac{k^2}{2n} \right)^r \leq e^{-\frac{qk}{12n}},$$

where the last inequality is since $1 - x \leq e^{-x}$ and $r = \lfloor \frac{q}{3k} \rfloor \geq \frac{q}{6k}$. $\square$

This completes the proof of Lemma 4. Now we can finally bound the advantage of our adversary from Figure 4.

Continuing the proof of Theorem 1. As argued before, as long as event G happens and our adversary triggers $E_{i,j}$ for some pair (i, j) , from Claim 1 we will win the $\text{PR}^{k,q,H}(\mathcal{A})$ game. Therefore, we have:

$$\begin{aligned} \text{Adv}_{k,q}^{\text{PR-RAND}}(\mathcal{A}) &\geq \Pr[(\exists i, j : E_{i,j}) \wedge G] = \Pr[(\exists i, j : E_{i,j}) | G] \cdot \Pr[G] \\ &= \left(1 - \Pr\left[\bigwedge_{(i,j)} \overline{E_{i,j}} | G\right]\right) \cdot \Pr[G] \geq \left(1 - e^{-\frac{qk}{12n}}\right) \cdot \frac{1}{2} \geq \frac{qk}{48n}, \end{aligned}$$

where the second-to-last inequality is from Lemma 3 and Lemma 4, while the last inequality is from the fact $1 - x \geq e^{-2x}$ for $0 \leq x \leq \frac{1}{2}$. (Notice that $\frac{qk}{n} \leq \frac{1}{8}$, so the condition $\frac{qk}{24n} \leq \frac{1}{2}$ is satisfied.) $\square$

3.2 Security Bound

In this section, we prove the advantage upper bound of any adversary $\mathcal{A}$ in the preimage resistance game $\text{PR}^{k,q,H}(\mathcal{A})$ for H^k .

Since we are interested in the advantage upper bound of $\mathcal{A}$, without loss of generality, we can make the following assumptions:

Assumption 1. $\mathcal{A}$ is deterministic.

Otherwise, we can always fix the best randomness that maximizes $\text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A})$.

Assumption 2. $\mathcal{A}$ does not make repeated queries.

Otherwise, we can construct an adversary that has the same advantage while making fewer queries.

Assumption 3. $\mathcal{A}$ explicitly queries H on the submitted answer x' .

Otherwise, we can construct an adversary that always does so by making one extra query on x' before submitting x' as the answer. Since $q + 1 \leq 2q$, this will at most add a constant factor of 2 to our bounds.

Now we state our main theorem:

Theorem 2. Suppose $\frac{qk}{n} \leq \frac{1}{5}$ and $\frac{k^2}{n} \leq 1$. For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we have $\text{Adv}_{k,q,H}^{\text{PR-RAND}}(\mathcal{A}) \leq \frac{5qk+k^2}{n}$.

If an adversary $\mathcal{A}$ wins the preimage resistance game $\text{PR}^{k,q,H}(\mathcal{A})$, at least one of the following events will be triggered (recall that x is the randomly sampled value by the challenger at the beginning of the $\text{PR}^{k,q,H}(\mathcal{A})$ game):

- Event T_1 : $\mathcal{A}$ submits exactly the same $x' = x$ as the preimage.
- Event T_2 : $\mathcal{A}$ submits some different $x' \neq x$ as the preimage (but it still holds that $H^k(x') = H^k(x)$).

We will treat the two events separately.

Bounding $\Pr[T_1]$

Theorem 3. Assume $\frac{q+k-1}{n} \leq \frac{1}{2}$. For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we have $\Pr[T_1] \leq \frac{2(q+k-1)}{n}$.

Recall that for T_1 to happen, the adversary must submit exactly the same $x' = x$ in the $\text{PR}^{k,q,H}(\mathcal{A})$ game. We first do a game hopping by considering a different version of the $\text{PR}^{k,q,H}(\mathcal{A})$ game, and we call it $\text{PR1}^{q,H}(\mathcal{A})$. This new game is very similar to the traditional preimage resistance game for a random oracle H , where the adversary $\mathcal{A}$ is given $H(x)$ (instead of $H^k(x)$) and needs to submit a preimage, but here we require that the submitted preimage must be the same x :

Definition 9. Given a random oracle $H : [n] \rightarrow [n]$ and integer parameter q , for any adversary $\mathcal{A}$ that makes at most q oracle queries to H , the game $\text{PR1}^{q,H}(\mathcal{A})$ is defined as follows:

Game $\text{PR1}^{q,H}(\mathcal{A})$	
1 :	$x \leftarrow_{\$} [n]$
2 :	$y \leftarrow H(x)$
3 :	$x' \leftarrow \mathcal{A}^H(y)$
4 :	if $(x' = x)$
5 :	return 1
6 :	else return 0

Fig. 6. Security game $\text{PR1}^{q,H}(\mathcal{A})$.

The advantage of $\mathcal{A}$ in this game, denoted by $\text{Adv}_{q,H}^{\text{PR1}}(\mathcal{A})$, is defined as the probability that the game $\text{PR1}^{q,H}(\mathcal{A})$ returns 1, where the probability is taken over the choice of H and x .

We first bound $\text{Adv}_{q,H}^{\text{PR1}}(\mathcal{A})$ for any $\mathcal{A}$:

Claim 3. Assume $\frac{q}{n} \leq \frac{1}{2}$. For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we have $\text{Adv}_{q,H}^{\text{PR1}}(\mathcal{A}) \leq \frac{2q}{n}$.

The proof of this claim can be found in Appendix A.1. It is very similar to the proof of preimage resistance property of random oracles, e.g. see [28].

This claim gives an advantage upper bound for any $\mathcal{A}$ in the game $\text{PR1}^{q,H}(\mathcal{A})$. Now, for any $\mathcal{A}$ in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, we relate $\Pr[T_1]$ of $\mathcal{A}$ to the advantage of some adversary $\mathcal{B}$ in the $\text{PR1}^{q+k-1,H}(\mathcal{B})$ game (notice that we allow $k-1$ extra queries for $\mathcal{B}$):

Lemma 5. *For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, we can construct an adversary $\mathcal{B}$ also satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game, such that $\Pr[T_1] \leq \text{Adv}_{q+k-1,H}^{\text{PR}1}(\mathcal{B})$.*

Proof. We construct $\mathcal{B}$ by simulating the challenger of the $\text{PR}^{k,q,H}(\mathcal{A})$ game. However, it does not sample x itself. Instead, it (implicitly) uses the same x in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game.

The construction of $\mathcal{B}$ consists of two parts. In the first part, $\mathcal{B}$ receives y in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game at the beginning, and sets $x_1 = y$. Then, it makes at most $k-1$ queries to the challenger: For $1 \leq i < k$, if x_i is a fresh query (i.e. $x_i \notin \{x_1, \dots, x_{i-1}\}$), it queries x_i , and receives $x_{i+1} = H(x_i)$ from the challenger. Otherwise, it sets $x_{i+1} = H(x_i)$ according to the previous query on x_i . After these queries, it knows the length- $(k-1)$ chain starting from x_1 , namely $x_1, x_2 = H(x_1), \dots, x_k = H^{k-1}(x_1)$.

In the second part, $\mathcal{B}$ starts to simulate the challenger of the $\text{PR}^{k,q,H}(\mathcal{A})$ game. It first forwards x_k as the challenge to $\mathcal{A}$. (Notice that $x_k = H^{k-1}(x_1) = H^k(x)$, so it indeed has simulated the $\text{PR}^{k,q,H}(\mathcal{A})$ game where the challenger samples x and forwards $H^k(x)$ as the challenge.)

Now, for the j_{th} query z_j of $\mathcal{A}$:

- If $\mathcal{B}$ has not queried z_j before, it forwards z_j to the challenger, receives the response $H(z_j)$, and forwards the response $H(z_j)$ back to $\mathcal{A}$.
- Otherwise, $\mathcal{B}$ directly forwards $H(z_j)$ to $\mathcal{A}$.

Finally, $\mathcal{B}$ forwards the answer x' submitted by $\mathcal{A}$ to the challenger.

Now we focus on some properties of $\mathcal{B}$. First, $\mathcal{B}$ still satisfies Assumptions 1, 2 and 3, because given that $\mathcal{A}$ is deterministic and that $\mathcal{A}$ does not make repeated queries, $\mathcal{B}$ is also deterministic and does not make repeated queries. Further, since $\mathcal{A}$ has queried its submission x' in one of its queries, and that $\mathcal{B}$ also submits x' as the answer, it means $\mathcal{B}$ will also have queried it in one of the previous queries.

Secondly, if $\mathcal{A}$ triggers event T_1 in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, then it will correctly return $x' = x$, and $\mathcal{B}$ will submit this x' as well. By our construction, this x remains the same in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game, so we will also win the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game in this case.

To summarize, we have constructed an adversary $\mathcal{B}$ that makes at most $q+k-1$ queries, and wins the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game whenever $\mathcal{A}$ triggers event T_1 in the $\text{PR}^{k,q,H}(\mathcal{A})$ game. Hence, we immediately have $\Pr[T_1] \leq \text{Adv}_{q+k-1,H}^{\text{PR}1}(\mathcal{B})$. $\square$

Now we can prove Theorem 3.

Proof of Theorem 3. First, given any adversary $\mathcal{A}$ in the $\text{PR}^{k,q,H}(\mathcal{A})$ game that satisfies Assumptions 1, 2 and 3, from Lemma 5 we can construct an adversary $\mathcal{B}$ (also satisfying Assumptions 1, 2 and 3) in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game such that $\Pr[T_1] \leq \text{Adv}_{q+k-1,H}^{\text{PR}1}(\mathcal{B})$. Now, according to Claim 3, we know that

$\text{Adv}_{q+k-1, H}^{\text{PR1}}(\mathcal{B}) \leq \frac{2(q+k-1)}{n}$ for any $\mathcal{B}$ satisfying Assumptions 1, 2 and 3. Therefore, we immediately have $\Pr[T_1] \leq \frac{2(q+k-1)}{n}$ for any $\mathcal{A}$ satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{k, q, H}(\mathcal{A})$ game. $\square$

Bounding $\Pr[T_2]$ Now we focus on event T_2 . The intuition is as follows. Recall that if event T_2 happens, the adversary will submit $x' \neq x$ (where x is sampled by the challenger at the beginning of the preimage resistance game) as the preimage. Now consider the two length- k chains starting from x and x' : They end at the same point y , so they will “collide” somewhere (see Claim 4 for a rigorous definition). As a result, we can utilize this collision to design a pair of encoding and decoding algorithms that can successfully compress the random oracle H . Hence from Lemma 2, this event should not happen with high probability.

We begin with the following claim:

Claim 4. If T_2 happens (i.e. the adversary $\mathcal{A}$ submits $x' \neq x$ as the preimage in the $\text{PR}^{k, q, H}(\mathcal{A})$ game), then there exists $1 \leq i \leq k$ such that

$$H^{i-1}(x') \neq H^{i-1}(x) \quad \wedge \quad H^i(x') = H^i(x),$$

where we define $H^0(x) = x$.

In other words, this claim states that if T_2 happens, we can always find a collision on $H^{i-1}(x')$ and $H^{i-1}(x)$.

Proof. Find the smallest i ($1 \leq i \leq k$) such that $H^i(x') = H^i(x)$. Such i must exist because we know $H^k(x') = H^k(x)$ (which is guaranteed by event T_2). If $i > 1$, then from the minimality of i we immediately know $H^{i-1}(x') \neq H^{i-1}(x)$. If $i = 1$, then $H^{i-1}(x') = H^0(x') = x'$ and $H^{i-1}(x) = H^0(x) = x$, but from event T_2 we know that $x' \neq x$, so $H^{i-1}(x') \neq H^{i-1}(x)$ still holds. In either case, this i will always satisfy both requirements above. $\square$

From the observation above, we can use $\mathcal{A}$ to help us compress H , by describing the collision point with fewer than $\log n$ bits.

To help the analysis, we first define the game $\text{PR2}^{k, q, H}(\mathcal{A}, x)$. It is very similar to the original $\text{PR}^{k, q, H}(\mathcal{A})$ game, except that the value x is provided as input instead of being sampled by the challenger. However, the adversary still does not know x :

Definition 10. Given a random oracle $H : [n] \rightarrow [n]$ and integer parameters k, q , for any adversary $\mathcal{A}$ that makes at most q oracle queries to H , and any integer $x \in [n]$, the game $\text{PR2}^{k, q, H}(\mathcal{A}, x)$ is defined as follows: The advantage of $\mathcal{A}$ in this game, denoted by $\text{Adv}_{k, q, H}^{\text{PR2}}(\mathcal{A}, x)$, is defined as the probability that the game $\text{PR2}^{k, q, H}(\mathcal{A}, x)$ returns 1, where the probability is taken over the choice of H .

Similarly, the event T_2 in the $\text{PR2}^{k, q, H}(\mathcal{A}, x)$ game is defined as the event that the adversary $\mathcal{A}$ submits some $x' \neq x$ such that $H^k(x') = H^k(x)$.

Game $\text{PR2}^{k,q,H}(\mathcal{A}, x)$	
1 :	$y \leftarrow H^k(x)$
2 :	$x' \leftarrow \mathcal{A}^H(y)$
3 :	if ($H^k(x') = y$)
4 :	return 1
5 :	else return 0

Fig. 7. Security game $\text{PR2}^{k,q,H}(\mathcal{A}, x)$.

We will first give a bound for $\Pr[T_2]$ in the $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ game for any fixed x , then derive the bound for $\Pr[T_2]$ in the original game $\text{PR}^{k,q,H}(\mathcal{A})$ where x is sampled randomly by the challenger.

Specifically, we will construct a pair of encoding and decoding algorithms in which we simulate the preimage resistance game $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ for some fixed x and fixed parameters k, q, n . If $\mathcal{A}$ wins this game by triggering event T_2 , then we can successfully compress the random oracle H . Therefore, if the winning probability of $\mathcal{A}$ is too high, our constructed algorithms will have a high probability of compressing H , which will violate Lemma 2. As a result, for any fixed x , the winning probability of $\mathcal{A}$ by triggering T_2 (in the $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ game) will be bounded.

Theorem 4. *For any adversary $\mathcal{A}$ satisfying assumptions 1, 2 and 3, and any $x \in [n]$, in the $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ game, we have $\Pr[T_2] \leq \frac{(q+k)k}{n}$.*

Proof. Given $\mathcal{A}, x$, we will construct a pair of encoding and decoding algorithms $(\text{Enc}_{\mathcal{A},k,q,n,x}, \text{Dec}_{\mathcal{A},k,q,n,x})$, such that as long as $\mathcal{A}$ triggers T_2 in the $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ game, we can always successfully compress H . (Notice that both algorithms know $\mathcal{A}$ and x at the beginning, and will also explicitly use $\mathcal{A}$ and x in their code.)

Our encoding algorithm $\text{Enc}_{\mathcal{A},k,q,n,x}(H)$ takes H as input, and will output some bits σ (as the compression of H).

1. From x and H , compute the length- k chain starting from x , i.e. compute $x_i = H^i(x)$ for $0 \leq i \leq k$ (where we define $x_0 = H^0(x) = x$). Let $S_0 = \{x_0, x_1, \dots, x_{k-1}, x_k\}$. Now we want to store the function values of H that have occurred in S_0 . We will use a bit $b \in \{0, 1\}$ to distinguish two cases, depending on whether the chain contains repeating elements. To this end, find the first index i such that $x_i = x_j$ for some $0 \leq j < i$ (i.e. find the first index that repeats in the chain).
 - If such i exists, we set $b = 1$. In this case, there is a collision within the chain. Now, to store the chain S_0 , we only store the first $i - 1$ entries $x_1, \dots, x_{i-1}$. For the value x_i , we do not store it directly, but use $2 \log k$ bits to store the indices i and j . (With these indices, later we can restore

the value of x_i by looking at the value x_j .) Now skip the following steps and directly jump to step 6.

- If such i does not exist, we set $b = 0$. Now, to store S_0 , we simply use $k \log n$ bits to store $x_1, x_2, \dots, x_k$. (Notice we have $x_k = H^k(x) = y$ in the $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ game.) Now continue with the following steps.
- 2. Feed y to $\mathcal{A}$, and simulate the game $\text{PR2}^{k,q,H}(\mathcal{A}, x)$ for $\mathcal{A}$ based on H . Specifically, for each query z_i made by $\mathcal{A}$, if we have not stored the entry for z_i before, we append the value $H(z_i)$ to the table that we stored in the previous step, and respond to $\mathcal{A}$ with this $H(z_i)$. Otherwise we already know the value $H(z_i)$ and can respond directly.
- 3. Get the submitted answer x' from $\mathcal{A}$.
- 4. For $1 \leq i \leq k$ in increasing order, check if we have stored the entry for $H^{i-1}(x')$. If not, we append the entry $H^i(x')$ to the end of the table. Let q' be the total number of entries added to the table in steps 2 and 4. By construction we have $q' \leq q + k$.
- 5. Now, let us assume that $\mathcal{A}$ wins this specific game by triggering event T_2 . (If not, claim failure and abort.) According to Claim 4, there exists i s.t. $H^{i-1}(x')$ and $H^{i-1}(x)$ form a collision. Notice that we must have stored the entry $H^i(x')$ in our table (in either step 2 or 4). Suppose it is the j_{th} ($1 \leq j \leq q'$) entry. Hence we can use $\log(q+k)$ bits to store the index j , and $\log k$ bits to store the offset i . Now we can remove the entry for $H^{i-1}(x')$ from the table (Looking ahead, since we know $H^i(x') = H^i(x)$, we can use i and j together with the stored value x_i in S_0 to recover it as $H(H^{i-1}(x')) = x_i$ in our decoding algorithm).
- 6. The final output σ of the encoding algorithm will contain the following parts in order:
 - 1 bit for the bit b (from step 1).
 - If $b = 0$, $\log(q+k) + \log k$ bits for i and j (from step 5).
 - If $b = 1$, $2 \log k$ bits for i and j (from step 1).
 - The (partially removed) table maintained through steps 1-5. (This table contains some function values of H .)
 - The remaining entries of the function table of H in lexicographic order.

Our decoding algorithm $\text{Dec}_{\mathcal{A},x}(\sigma)$ takes the bits σ generated by our encoding algorithm as input, and needs to recover the whole function table of H . Specifically, it starts with an empty function table for H . For each entry $l \in [n]$, it will fill in its function value $H(l)$ at some time in the algorithm. The decoding algorithm is as follows:

1. From σ , read the first bit b (i.e. the bit b generated in encoding step 1).
2. If $b = 0$:
 - (a) Read the first $\log(q+k) + \log k$ bits to get i and j .
 - (b) Read the next $k \log n$ bits to restore the chain S_0 and fill in the corresponding hash values. Specifically:
 - i. Denote $x_0 = x$ (Recall that x is the known parameter in our decoding algorithm $\text{Dec}_{\mathcal{A},x}(\sigma)$). Set $S_0 = \{x_0\}$ in the beginning.

- ii. For $1 \leq i \leq k$:
 - Read the next $\log n$ bits to get x_i . Set $H(x_{i-1}) = x_i$ in the table. Add x_i to S_0 .
- (c) Now we know $S_0 = \{x_0, x_1, \dots, x_k\}$. Simulate the game $\text{PR}2^{k,q,H}(\mathcal{A}, x)$ to $\mathcal{A}$ by first feeding $y(=x_k)$ to $\mathcal{A}$. For the l_{th} query z_l made by $\mathcal{A}$:
 - If $l \neq j$:
 - If the function value of z_l has not been filled, we read out the next entry y_l from the stored function table and answer correspondingly. Meanwhile, we use it to fill the function value of z_l (i.e. set $H(z_l) = y_l$).
 - Otherwise, we directly answer with $H(z_l)$.
 - If $l = j$, we know that we have removed this entry in encoding step 5, and that this query forms a collision with $H^{i-1}(x)$. Hence, we do not read the next entry, and answer directly with x_i (and fill in the function table correspondingly).
- (d) Get the answer x' submitted by $\mathcal{A}$. Set a counter $i' = q$.
- (e) For $1 \leq l \leq k$ in increasing order:
 - If the value for $H^{l-1}(x')$ has been filled, skip this l .
 - Otherwise, if $i' \neq j$, then set $i' \leftarrow i' + 1$, read the next entry y_l from the table, and fill in $H(H^{l-1}(x')) = y_l$.
 - Otherwise ($i' = j$), from event T_2 and our encoding algorithm, we know that $H^l(x') = H^i(x) = x_i$. Thus we don't read the next entry, and directly fill in the table as $H(H^{l-1}(x')) = x_i$.
- 3. If $b = 1$:
 - (a) Read the next $2 \log k$ bits to get i and j .
 - (b) Read the next $(i-1) \log n$ bits to get $x_1, \dots, x_{i-1}$ in the chain S_0 and fill in the corresponding hash values (similar to step 2b).
 - (c) Fill in $H(x_{i-1}) = x_j$. (By our encoding algorithm, it is guaranteed that $H(x_{i-1}) = x_i = x_j$.)
- 4. Continue reading out the remaining entries of the function table of H , and fill in the function table in lexicographic order.

From the construction, as long as $\mathcal{A}$ wins by triggering event T_2 , our algorithm can successfully recover the whole function table of H .

Now we look at the number of bits saved. To store the whole function table of H , normally we need to store $\log n$ bits for each entry. In fact, in our encoding algorithm, for all but one entry of the function table, we use $\log n$ bits to describe them plainly (but possibly in a different order). Further, notice that our algorithm will never store the same entry twice. Therefore, our algorithm does not save (or waste) any bits for these entries.

Our saving comes from the fact that we removed one entry from the function table of H (which happens either in encoding step 1 or 5). This entry would normally require $\log n$ bits to store, but instead we stored extra bits for i and j , which require $\log(q+k) + \log k$ or $2 \log k$ bits. Notice that $2 \log k \leq \log(q+k) + \log k$. To conclude, we can save at least $\log n - (\log(q+k) + \log k)$ bits.

To summarize, as long as $\mathcal{A}$ triggers T_2 in the $\text{PR}2^{k,q,H}(\mathcal{A}, x)$ game, we can successfully compress the random oracle H by at least $\log n - (\log(q+k) + \log k)$

bits. From Lemma 2, this can happen with probability at most $\frac{(q+k)k}{n}$. Therefore, it immediately means $\Pr[T_2] \leq \frac{(q+k)k}{n}$ in the $\text{PR}^{k,q,H}(\mathcal{A}, x)$ game. $\square$

The above theorem gives an advantage upper bound for the $\text{PR}^{k,q,H}(\mathcal{A}, x)$ game for any x . Now we can easily derive the same bound for the original game $\text{PR}^{k,q,H}(\mathcal{A})$:

Theorem 5. *For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, we have $\Pr[T_2] \leq \frac{(q+k)k}{n}$.*

Proof. We have

$$\Pr[T_2] = \sum_{i=1}^n \Pr[T_2|x=i] \Pr[x=i] = \frac{1}{n} \sum_{i=1}^n \Pr[T_2|x=i] \leq \frac{1}{n} \sum_{i=1}^n \left(\frac{(q+k)k}{n} \right) = \frac{(q+k)k}{n}.$$

The inequality is directly from Theorem 4, because in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, given $x=i$, the event T_2 is essentially the event T_2 in the $\text{PR}^{k,q,H}(\mathcal{A}, i)$ game. $\square$

Now we can finally prove Theorem 2.

Proof of Theorem 2. If $\mathcal{A}$ wins the $\text{PR}^{k,q,H}(\mathcal{A})$ game, then at least one of the events T_1 or T_2 will happen. Notice that since $\frac{qk}{n} \leq \frac{1}{5}$, the condition $\frac{q+k-1}{n} \leq \frac{1}{2}$ in Theorem 3 is satisfied. By applying the union bound on Theorem 3 and Theorem 5, we immediately have

$$\begin{aligned} \text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A}) &\leq \Pr[T_1 \vee T_2] \leq \Pr[T_1] + \Pr[T_2] \leq \frac{2(q+k-1)}{n} + \frac{(q+k)k}{n} \\ &\leq \frac{2q}{n} + \frac{2k}{n} + \frac{qk}{n} + \frac{k^2}{n} \leq \frac{5qk}{n} + \frac{k^2}{n}. \end{aligned} \quad \square$$

4 Preimage Resistance Results for H^k in the Random Permutation Model

In this section, we focus on the case that H is a random permutation in the $\text{PR}^{k,q,H}(\mathcal{A})$ game. Throughout this section, let $H : [n] \rightarrow [n]$ be a random permutation in which the adversary has oracle access to.² In Section 4.1, we provide an adversary with advantage $\Omega(\frac{q}{n})$. In Section 4.2, we show that no adversary $\mathcal{A}$ can achieve advantage better than $O(\frac{q+k}{n})$.

² For some previous works on iterated random permutations, e.g. [27], the inverse oracle H^{-1} is also provided to the adversary. This is reasonable in the context of distinguishing H and H^k . However, since we directly focus on security properties of H^k , there is no reason to give oracle access to H^{-1} (otherwise the preimage resistance game would be trivial).

$\mathcal{A}(k, q, n, y)$	
1 :	$S \leftarrow \emptyset, r \leftarrow 0, q_{cnt} \leftarrow 0$
2 :	while ($q_{cnt} < q$)
3 :	$r \leftarrow r + 1, j \leftarrow 0, x_{r,0} \leftarrow \min\{[n] \setminus S\}$
4 :	while ($q_{cnt} < q \wedge x_{r,j} \notin S$)
5 :	$S \leftarrow S \cup \{x_{r,j}\}$
6 :	$j \leftarrow j + 1, q_{cnt} \leftarrow q_{cnt} + 1$
7 :	$x_{r,j} \leftarrow H(x_{r,j-1})$
8 :	$t_r \leftarrow j$
9 :	for $i \in [1, r]$
10 :	for $j \in [1, t_i]$
11 :	if ($x_{i,j} = y \wedge (i < r \vee j \geq k)$)
12 :	return $x_{i,(j-k) \bmod t_i}$

Fig. 8. Our adversary $\mathcal{A}(k, q, n, y)$ against breaking preimage resistance of H^k for random permutation H .

4.1 Attack

We first provide an adversary that has advantage $\Omega(\frac{q}{n})$.

Theorem 6. *Assume $q \geq 2k$. There is an adversary $\mathcal{A}$ that achieves advantage $\text{Adv}_{k,q}^{\text{PR-PERM}}(\mathcal{A}) \geq \frac{q}{2n}$ in the preimage resistance game.*

We first give an intuition for our attack, which is depicted in Figure 8. We will query the chain $1, H(1), H^2(1), \dots$ until we enter a loop. Since H is a permutation, this loop must be a cycle. Now we pick the smallest unvisited value x and query the chain $x, H(x), H^2(x), \dots$ until we enter another loop. Repeat until we have used up all q queries. Now we have obtained a number of cycles and a chain (or a cycle) of H . If the challenge y lies on one of the cycles, then we can easily find its preimage in the cycle. Otherwise, if y lies on the chain, we can also trace back k steps and find a preimage as long as there are at least k values in the chain before y . As we prove later, this strategy guarantees that we win on $\Omega(q)$ challenges in total.

Proof of Theorem 6. We first show the correctness of $\mathcal{A}$:

Lemma 6. *Suppose the above adversary $\mathcal{A}(k, q, n, y)$ finds and outputs some $x_{i',j'}$ as the answer. Then $H^k(x_{i',j'}) = y$.*

Proof of Lemma 6. We will reuse the variables r, t_i (where $1 \leq i \leq r$), and $x_{i,j}$ (where $1 \leq i \leq r, 1 \leq j \leq t_i$) in our adversary. Intuitively, r is the number of rounds executed in the while loop (lines 2-9), t_i is the number of queries $\mathcal{A}$ made in the i_{th} round. Notice that we always have $x_{i,j} = H^j(x_{i,0})$.

Consider when $\mathcal{A}$ will jump out of the loop in lines 5-8. The condition $q_{cnt} = q$ can only happen at the last round. Hence, for the first $i (i < r)$ rounds, we have $x_{i,t_i} \in S$. Since H is a permutation, each value in $[n]$ should have exactly one preimage in H . Therefore, the only remaining possibility is that $x_{i,t_i} = x_{i,0}$ (because by the construction of S , all the other elements in S already have a preimage). Formally, we can prove the following claim:

Claim 5. For any $1 \leq i < r$, we have $x_{i,t_i} = x_{i,0}$.

Proof. We prove the claim via induction on i .

- When $i = 1$, before execution of line 9, the set S consists of $x_{1,0}, \dots, x_{1,t_1-1}$ satisfying $x_{1,i+1} = H(x_{1,i})$ for all $0 \leq i < t_1$, and since $x_{1,t_1} \in S$, we know there exists some $0 \leq j < t_1$ s.t. $x_{1,t_1} = x_{1,j}$. Suppose on the contrary we have $x_{1,t_1} = x_{1,j}$ for some $j \neq 0$. It means that $H(x_{1,t_1-1}) = x_{1,t_1} = x_{1,j} = H(x_{1,j-1})$. Since $t_1 \neq j$, we have found a collision on H , which is a contradiction. Hence we must have $x_{1,t_1} = x_{1,0}$.
- Suppose the claim holds for all $i' < i$. For the i th round, suppose on the contrary $x_{i,t_i} = x_{i',j'}$ for some pair (i', j') where $i' < i$ or $j' > 0$.
 - If $j' > 0$, we can find the following collision on H :

$$H(x_{i,t_i-1}) = x_{i,t_i} = x_{i',j'} = H(x_{i',j'-1}).$$

- If $j' = 0$ (which means $i' < i$), by induction hypothesis we know $x_{i',0} = x_{i',t_{i'}}$. Hence we can find the following collision on H :

$$H(x_{i',t_{i'}-1}) = x_{i',t_{i'}} = x_{i',0} = x_{i,t_i} = H(x_{i,t_i-1})$$

Therefore, the only possibility is that $x_{i,t_i} = x_{i,0}$, completing the induction. $\square$

With this claim, we now can prove Lemma 6. Consider the output $x_{i',j'}$ generated by $\mathcal{A}(k, q, n, y)$.

1. If $j' \geq k$, then $j' - k \geq 0$ and we immediately have $H^k(x_{i',j'-k}) = x_{i',j'} = y$.
2. Otherwise, since $j' < k$, we must have $i' < r$. Hence, from the above claim we know $x_{i',t_{i'}} = x_{i',0}$. Therefore, we have

$$\begin{aligned} H^k(x_{i',(j'-k) \bmod t_{i'}}) &= H^k(x_{i',j'-k+t_{i'}}) \\ &= H^{j'}(x_{i',t_{i'}}) \\ &= H^{j'}(x_{i',0}) \\ &= x_{i',j'} \\ &= y. \end{aligned} \quad \square$$

Now let's look at the advantage of $\mathcal{A}$. Notice that $\mathcal{A}$ is deterministic and only uses the challenge y in the last step. Specifically, for any fixed permutation H ,

the queries $\mathcal{A}$ made will be the same regardless of y . Therefore, it remains to count the number of x values (recall that x is the randomly sampled value by the challenger at the beginning of the $\text{PR}^{k,q,H}(\mathcal{A})$ game) that $\mathcal{A}$ can win given $y = H^k(x)$.

Intuitively, consider the cycles (and possibly one chain) $\mathcal{A}$ generates during execution. As long as x falls in one of the cycles, or the “front part” of the chain (i.e. there are at least k elements after x in the chain), then $\mathcal{A}$ can “trace back” k steps from y to find a valid preimage and win the game. In total, there are $q - k$ different x ’s that $\mathcal{A}$ can win under this H , which gives an advantage $\Omega(\frac{q}{n})$. As it holds for any H , the advantage also holds in the original $\text{PR}^{k,q,H}(\mathcal{A})$ game where H is a random permutation.

Formally, we will prove the following lemma:

Lemma 7. *Suppose $q \geq 2k$. Let $\mathcal{A}$ be the adversary defined above. For any permutation H mapping $[n]$ to $[n]$, we have $\text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A}) \geq \frac{q}{2n}$.*

Proof. From our construction, since $\mathcal{A}$ is deterministic, we know that for any fixed permutation H , the queries $\mathcal{A}$ made will be the same regardless of the challenge y . Denote by $x_{i,j}$ (where $1 \leq i \leq r$ and $0 \leq j < t_i$) the queries $\mathcal{A}$ made. Denote by x the random value that the challenger samples at the beginning. As a result, we only need to count the number of x ’s that $\mathcal{A}$ can win for this H . According to Lemma 6, $\mathcal{A}$ can win if one of the following happens:

1. $x = x_{r,j}$ for some $j < t_r - k$. Then immediately we have $y = H^k(x) = x_{r,j+k}$, and there exists $(i', j') = (r, j+k)$ satisfying case 1 and $\mathcal{A}$ can win the game.
2. $x = x_{i,j}$ for some $i < r$. In that case, we have $y = H^k(x) = H^k(x_{i,j}) = H^k(x_{i,(j+k) \bmod t_i})$ (from Claim 5). Therefore, there exists $(i', j') = (i, (j+k) \bmod t_i)$ satisfying case 2 and $\mathcal{A}$ can win the game.

Now we count the number of x ’s satisfying the above requirements. Notice that $\sum_{i=1}^r t_i = q$ (because $\mathcal{A}$ makes q queries in total). Further, all the values $x_{i,j}$ are different (due to our construction of $\mathcal{A}$). Therefore, there are in total

$$(t_r - k) + \sum_{i=1}^{r-1} t_i = \left(\sum_{i=1}^r t_i \right) - k = q - k$$

different x ’s that $\mathcal{A}$ can win under this fixed H .

Finally, we can compute the advantage $\text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A})$. Since $\mathcal{A}$ is deterministic, the randomness completely comes from the permutation H and the sampled value x :

$$\begin{aligned} \text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A}) &= \sum_{x=1}^n \frac{1}{n} \cdot \Pr \left[\mathcal{A} \text{ wins the } \text{PR}^{k,q,H}(\mathcal{A}) \text{ game} \mid H, x \right] \\ &\geq \sum_{x=1}^n \frac{1}{n} \cdot \frac{q - k}{n} = \frac{q - k}{n} \geq \frac{q}{2n}. \end{aligned}$$

where the last inequality is since $q \geq 2k$. □

Now we can finally prove Theorem 6. Notice that Lemma 7 holds for any permutation H . Therefore, we immediately have $\text{Adv}_{k,q}^{\text{PR-PERM}}(\mathcal{A}) \geq \frac{q}{2n}$ for H sampled uniformly at random over all permutations mapping $[n]$ to $[n]$. $\square$

4.2 Security Bound

We show the following theorem:

Theorem 7. *Suppose $\frac{q+k}{n} \leq \frac{1}{2}$. For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we have $\text{Adv}_{k,q,H}^{\text{PR-PERM}}(\mathcal{A}) \leq \frac{2(q+k-1)}{n}$.*

We first give an intuition for our proof. Notice that since H is a permutation, the only possibility for $\mathcal{A}$ to win on challenge $y = H^k(x)$ is to submit exactly the same x . Therefore, we can use ideas similar to the proof of Theorem 3. Specifically, we first reduce the $\text{PR}^{k,q,H}(\mathcal{A})$ game to a new game $\text{PR}^{3^{q+k-1},H}(\mathcal{A})$, and then show that no adversary can have good advantage in the $\text{PR}^{3^{q+k-1},H}(\mathcal{A})$ game.

Definition 11. *Given a random permutation $H : [n] \rightarrow [n]$ and integer parameter q , for any adversary $\mathcal{A}$ that makes at most q oracle queries to H , and any integer $x \in [n]$, the game $\text{PR}^{3^q,H}(\mathcal{A}, x)$ is defined as follows: The advantage of $\mathcal{A}$*

Game $\text{PR}^{3^q,H}(\mathcal{A})$	
1 :	$y \leftarrow_{\$} [n]$
2 :	$x' \leftarrow \mathcal{A}^H(y)$
3 :	if $(H(x') = y)$
4 :	return 1
5 :	else return 0

Fig. 9. Security game $\text{PR}^{3^q,H}(\mathcal{A})$.

in this game, denoted by $\text{Adv}_{q,H}^{\text{PR}^3}(\mathcal{A})$, is defined as the probability that the game $\text{PR}^{3^q,H}(\mathcal{A})$ returns 1, where the probability is taken over the choice of H and x .

Notice that in this game y is randomly sampled in $[n]$, and the adversary needs to output a preimage of y under the permutation H (instead of H^k).

To begin with, we first bound $\text{Adv}_{q,H}^{\text{PR}^3}(\mathcal{A})$ for any $\mathcal{A}$:

Claim 6. Assume $\frac{q}{n} \leq \frac{1}{2}$. For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we have $\text{Adv}_{q,H}^{\text{PR}^3}(\mathcal{A}) \leq \frac{2q}{n}$.

Proof of Claim 6. Denote by $z_1, z_2, \dots, z_q$ the queries made by $\mathcal{A}$. From Assumption 2, none of these queries repeat, so each query z_i is a fresh query

to the random permutation H . Further, since y is sampled at random, no information is known about H except these queries. Due to the lazy sampling property of H , the responses $H(z_i)$ should be sampled uniformly at random from $[n]$ without replacement. Specifically, it means that $H(z_i)$ is sampled uniformly from the set $[n] \setminus \{H(z_1), \dots, H(z_{i-1})\}$. If $y \notin \{H(z_1), \dots, H(z_{i-1})\}$, then $\Pr[H(z_i) = y] = \frac{1}{n-i+1}$. Otherwise, $\Pr[H(z_i) = y] = 0$. Therefore, for any $1 \leq i \leq q$, we have $\Pr[H(z_i) = y] \leq \frac{1}{n-i+1}$. From Assumption 3, if $\mathcal{A}$ wins the game, then there exists an index i such that $H(z_i) = y$. Therefore, from the union bound we have

$$\Pr[\mathcal{A} \text{ wins the } \text{PR}^{q,H}(\mathcal{A}) \text{ game}] \leq \sum_{i=1}^q \Pr[H(z_i) = y] \leq \sum_{i=1}^q \frac{1}{n-i+1} \leq \frac{q}{2n},$$

where the last inequality is from $\frac{q}{n} \leq \frac{1}{2}$ (so $n - q \geq \frac{n}{2}$). $\square$

Now, for any $\mathcal{A}$ against the $\text{PR}^{k,q,H}(\mathcal{A})$ game, we construct an adversary $\mathcal{B}$ against the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game:

Lemma 8. *For any adversary $\mathcal{A}$ satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{k,q,H}(\mathcal{A})$ game, we can construct an adversary $\mathcal{B}$ also satisfying Assumptions 1, 2 and 3 in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game, such that $\text{Adv}_{k,q,H}^{\text{PR-PERM}}(\mathcal{A}) \leq \text{Adv}_{q+k-1,H}^{\text{PR3}}(\mathcal{B})$.*

Proof. We construct $\mathcal{B}$ by simulating the challenger of the $\text{PR}^{k,q,H}(\mathcal{A})$ game. However, it does not sample x and compute the challenge $y = H^k(x)$, but utilizes the challenge y received in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game and passes $H^{k-1}(y)$ as the challenge to $\mathcal{A}$.

The construction of $\mathcal{B}$ consists of two parts. In the first part, $\mathcal{B}$ receives y in the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game at the beginning, and sets $x_1 = y$. Then, it makes at most $k-1$ queries to the challenger: For $1 \leq i < k$, if x_i is a fresh query (i.e. $x_i \notin \{x_1, \dots, x_{i-1}\}$), it queries x_i , and receives $x_{i+1} = H(x_i)$ from the challenger. After these queries, it knows the length- $(k-1)$ chain starting from x_1 , namely $x_1, x_2 = H(x_1), \dots, x_k = H^{k-1}(x_1)$.

In the second part, $\mathcal{B}$ starts to simulate the challenger of the $\text{PR}^{k,q,H}(\mathcal{A})$ game. It first forwards x_k as the challenge y to $\mathcal{A}$.

Now, for the j_{th} query z_j of $\mathcal{A}$:

- If $\mathcal{B}$ has not queried z_j before, it forwards z_j to the challenger, receives the response $H(z_j)$, and forwards the response $H(z_j)$ back to $\mathcal{A}$.
- Otherwise, $\mathcal{B}$ directly forwards $H(z_j)$ to $\mathcal{A}$.

Finally, $\mathcal{B}$ forwards the answer x' submitted by $\mathcal{A}$ to the challenger.

Now we focus on some properties of $\mathcal{B}$. First, $\mathcal{B}$ still satisfies Assumptions 1, 2 and 3, because given that $\mathcal{A}$ is deterministic and that $\mathcal{A}$ does not make repeated queries, $\mathcal{B}$ is also deterministic and does not make repeated queries. Further, since $\mathcal{A}$ has queried its submission x' in one of its queries, and that $\mathcal{B}$

also submits x' as the answer, it means $\mathcal{B}$ will also have queried it in one of the previous queries.

Secondly, $\mathcal{B}$ perfectly simulates the challenger of the $\text{PR}^{k,q,H}(\mathcal{A})$ game. To show this, it suffices to prove the following claim:

Claim 7. Fix any permutation $H : [n] \rightarrow [n]$. Then the following two distributions of y_1 and y_2 are the same:

- $y_1 \leftarrow H^k(x)$, where $x \leftarrow_{\$} [n]$.
- $y_2 \leftarrow H^{k-1}(y)$, where $y \leftarrow_{\$} [n]$.

Proof. We know $y_1 = H^k(x) = H^{k-1}(H(x))$. Hence it suffices to show that $H(x)$ (where $x \leftarrow_{\$} [n]$) and y (where $y \leftarrow_{\$} [n]$) have the same distribution.

Since H is a permutation, for any $i \in [n]$ there exists a unique $x \in [n]$ where $H(x) = i$. Now fix any $i \in [n]$, for randomly sampled $x \leftarrow_{\$} [n], y \leftarrow_{\$} [n]$, we immediately know $\Pr[H(x) = i] = \Pr[y = i] = \frac{1}{n}$. Since it holds for any $i \in [n]$, this finishes the proof. $\square$

Therefore, $\mathcal{B}$ indeed perfectly simulates the challenger in the $\text{PR}^{k,q,H}(\mathcal{A})$ game. Finally, we need to show the advantage of $\mathcal{B}$. We begin with the following claim on the iterated permutation H^k :

Claim 8. Fix a permutation $H : [n] \rightarrow [n]$ and an integer k . Then the function $\varphi(x) = H^k(x)$ is still a permutation mapping $[n]$ to $[n]$.

Proof of Claim 8. By definition, we only need to show that φ is bijective.

- φ is injective. Suppose there exists $\varphi(x_1) = \varphi(x_2)$ where $x_1 \neq x_2$. Then we claim that there exists $0 \leq i < k$ such that $H^i(x_1) \neq H^i(x_2)$ and $H^{i+1}(x_1) = H^{i+1}(x_2)$ (the proof is very similar to Claim 4). Therefore, we have found a collision on H , which is impossible since H is a permutation.
- φ is surjective. For any $y \in [n]$, we can compute its inverse as $x = H^{-k}(y)$, where H^{-1} is the inverse of the permutation H , and $H^{-k}(x)$ is the k_{th} iteration of H^{-1} on x . It is easy to verify that $H^k(x) = H^k(H^{-k}(y)) = y$.

$\square$

Now, if $\mathcal{A}$ wins our simulated $\text{PR}^{k,q,H}(\mathcal{A})$ game and returns some x' , we know

$$H^k(x') = H^{k-1}(y) \Leftrightarrow H^{k-1}(H(x')) = H^{k-1}(y) \Leftrightarrow H(x') = y,$$

where the last equivalence is since H^{k-1} is a permutation (from the claim above). Hence $\mathcal{B}$ will also win the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game.

To summarize, we have constructed an adversary $\mathcal{B}$ that makes at most $q+k-1$ queries, and wins the $\text{PR}^{q+k-1,H}(\mathcal{B})$ game whenever $\mathcal{A}$ wins the $\text{PR}^{k,q,H}(\mathcal{A})$ game. Hence, we immediately have $\text{Adv}_{k,q,H}^{\text{PR}}(\mathcal{A}) \leq \text{Adv}_{q+k-1,H}^{\text{PR3}}(\mathcal{B})$. $\square$

Now we can finally prove Theorem 7.

Proof of Theorem 7. First, given any adversary $\mathcal{A}$ in the $\text{PR}^{k,q,H}(\mathcal{A})$ game that satisfies Assumptions 1, 2 and 3, from Lemma 8 we can construct an adversary $\mathcal{B}$ (also satisfying Assumptions 1, 2 and 3) in the $\text{PR3}^{q+k-1,H}(\mathcal{B})$ game such that $\text{Adv}_{k,q,H}^{\text{PR-PERM}}(\mathcal{A}) \leq \text{Adv}_{q+k-1,H}^{\text{PR3}}(\mathcal{B})$. Now, since we have assumed $\frac{q+k}{n} \leq \frac{1}{2}$, we can apply Claim 6 and achieve $\text{Adv}_{q+k-1,H}^{\text{PR3}}(\mathcal{B}) \leq \frac{2(q+k-1)}{n}$. Therefore, for any $\mathcal{A}$ satisfying Assumptions 1, 2 and 3, we indeed have $\text{Adv}_{k,q,H}^{\text{PR-PERM}}(\mathcal{A}) \leq \frac{2(q+k-1)}{n}$. $\square$

5 Collision Resistance Results for H^k

In this section, we will focus on the collision resistance game $\text{CR}^{k,q,H}(\mathcal{A})$. In Section 5.1, we will give an adversary that achieves advantage $\Omega(\frac{q^2}{n})$. The attack is very similar to the traditional birthday attack. In Section 5.2, we show that no adversary can achieve advantage greater than $\mathcal{O}(\frac{(q+k)^2}{n})$. Under the assumption $q = \Omega(k)$, our attack matches the security bound.

Throughout this section, let $H : [n] \rightarrow [n]$ be a random oracle, and k, q be the given parameters for the collision resistance game. Similar to the preimage resistance case, we also assume k, q, n are all powers of 2.

5.1 Attack

Recall the traditional birthday attack against collision resistance for a hash function H , where the adversary makes q evaluations $H(x_1), \dots, H(x_q)$ and check if any of them form a collision (i.e. if there exists $x_i \neq x_j$ such that $H(x_i) = H(x_j)$). In fact, the same idea also works in the $\text{CR}^{k,q,H}(\mathcal{A})$ game, because if (x_i, x_j) form a collision in H , it will also form a collision in H^k :

$$H^k(x_i) = H^{k-1}(H(x_i)) = H^{k-1}(H(x_j)) = H^k(x_j).$$

Following this idea, we can give an attack with advantage $\mathcal{O}(\frac{q^2}{n})$.

Theorem 8. Assume $q \geq 4$ and $\frac{q^2}{n} \leq 1$. There exists an adversary $\mathcal{A}$ that achieves advantage $\text{Adv}_{k,q,H}^{\text{CR}}(\mathcal{A}) \geq \frac{q^2}{8n}$.

Proof. The adversary $\mathcal{A}$ works as depicted in Figure 10.

Denote by E the event that $\mathcal{A}$ returns some (i, j) . We first look at the probability of E . Here we give an upper bound for its complement event $\Pr[\overline{E}]$ instead. For $\overline{E}$ to happen, the response of each query should not coincide with any of the previous responses. From the lazy sampling property of H , each response is sampled uniformly from $[n]$. Therefore:

$$\Pr[\overline{E}] = \prod_{i=1}^q \left(1 - \frac{i-1}{n}\right) \leq \prod_{i=1}^q e^{-\frac{i-1}{n}} = e^{-\frac{q(q-1)}{2n}} \leq 1 - \frac{q(q-1)}{4n},$$

$\mathcal{A}(k, q, n)$	
1 :	for $i \in [1, q]$
2 :	$y_i \leftarrow H(i)$
3 :	for $i \in [1, q]$
4 :	for $j \in [i + 1, q]$
5 :	if $y_i = y_j$
6 :	return (i, j)
7 :	return $\perp$

Fig. 10. Our adversary $\mathcal{A}(k, q, n)$ breaking collision resistance of H^k where H is a random oracle.

where the first inequality holds because $1 - x \leq e^{-x}$, and the last inequality holds because $1 - x \geq e^{-2x}$ for $0 \leq x \leq \frac{1}{2}$ (notice that $\frac{q^2}{n} \leq 1$, so the condition $\frac{q(q-1)}{2n} \leq \frac{1}{2}$ is satisfied). As a result $\Pr[E] \geq \frac{q(q-1)}{4n} \geq \frac{q^2}{8n}$, since $q \geq 4$.

Next, given that E happens, we have

$$H^k(i_1) = H^{k-1}(H(i_1)) = H^{k-1}(H(i_2)) = H^k(i_2),$$

so the submitted (i_1, i_2) will indeed be accepted by the challenger, and $\mathcal{A}$ will win the game. Hence we have $\text{Adv}_{k,q,H}^{\text{CR}}(\mathcal{A}) \geq \Pr[E] \geq \frac{q^2}{8n}$ as required. $\square$

5.2 Security Bound

Now we give an upper bound for the advantage of any adversary $\mathcal{A}$. Notice that when $q = \Omega(k)$ (which is the case in most realistic scenarios), this bound matches the advantage of the attack given in the previous section.

Theorem 9. *For any adversary $\mathcal{A}$, we have $\text{Adv}_{k,q,H}^{\text{CR}}(\mathcal{A}) \leq \frac{(q+2k)^2}{n}$.*

The essence of the proof is similar to Theorem 4, where we design a pair of encoding and decoding algorithms that can compress the random oracle H whenever $\mathcal{A}$ wins the $\text{CR}_{k,q,H}^{\text{CR}}(\mathcal{A})$ game.

Proof. Without loss of generality, as in Theorem 2 we assume $\mathcal{A}$ is deterministic and does not make repeated queries. We begin with an observation very similar to Claim 4, which says that if the adversary wins the game by submitting (α_1, α_2) , then the two length- k chains starting from α_1 and α_2 will collide somewhere:

Claim 9. If $\mathcal{A}$ wins the $\text{CR}_{k,q,H}^{\text{CR}}(\mathcal{A})$ game by submitting some $\alpha_1 \neq \alpha_2$, then there exists $1 \leq i' \leq k$ satisfying:

$$H^{i'-1}(\alpha_1) \neq H^{i'-1}(\alpha_2) \quad \wedge \quad H^{i'}(\alpha_1) = H^{i'}(\alpha_2).$$

In other words, we can find a collision on $H^{i'-1}(\alpha_1)$ and $H^{i'-1}(\alpha_2)$.

The proof is almost the same as Claim 4. With this observation, we can use $\mathcal{A}$ to help us compress the random oracle H , by using fewer than $\log n$ bits to describe the collision point. Specifically, given $\mathcal{A}$, we will construct a pair of encoding and decoding algorithms that uses $\mathcal{A}$ as a subroutine.

Our encoding algorithm $Enc_{\mathcal{A},k,q}(H)$ takes H as input, and outputs some bits σ (as the compression of H):

1. Maintain a table which is empty at first.
2. Simulate the game $CR^{k,q,H}(\mathcal{A})$ to $\mathcal{A}$ based on H . Specifically, for each query z_i made by $\mathcal{A}$, we answer it by $H(z_i)$, and append the entry $H(z_i)$ to the end of the table.
3. Get the submitted answer (α_1, α_2) from $\mathcal{A}$.
4. For $0 \leq i < k$ in increasing order, if $\mathcal{A}$ has not queried $H^i(\alpha_1)$ before, we append the entry $H^{i+1}(\alpha_1)$ to the end of the table.
5. For $0 \leq i < k$ in increasing order, if $\mathcal{A}$ has not queried $H^i(\alpha_2)$ before, we append the entry $H^{i+1}(\alpha_2)$ to the end of the table. Let q' be the total number of entries in our table after this step. We have $q' \leq q + 2k$.
6. From Claim 9, there exists $1 \leq i' \leq k$ s.t. $H^{i'}(\alpha_1) = H^{i'}(\alpha_2)$ but $H^{i'-1}(\alpha_1) \neq H^{i'-1}(\alpha_2)$. From the construction above, we must have stored both entries $H^{i'}(\alpha_1)$ and $H^{i'}(\alpha_2)$ in our table (in one of the steps 2, 4 or 5). Suppose the indices of these two entries are r_1 and r_2 (where $1 \leq r_1 < r_2 \leq q + 2k$). We use $2\log(q + 2k)$ bits to store r_1 and r_2 , and remove the entry of index r_2 in our table. (The intuition is, we can restore this entry by looking at the entry of index r_1 . See the decoding algorithm for more details.)
7. The final output σ of the encoding algorithm will contain the following parts in order:
 - $2\log(q + 2k)$ bits for r_1 and r_2 .
 - The (partially removed) table generated through the previous steps. (This table contains some function values of H .)
 - The remaining entries of function table of H in lexicographic order.

Our decoding algorithm $Dec_{\mathcal{A},k,q}(\sigma)$ takes the bits generated by our encoding algorithm as input, and will restore the function table of H :

1. From σ , read the first $2\log(q + 2k)$ bits to get r_1 and r_2 .
2. Simulate the game $CR^{k,q,H}(\mathcal{A})$ to $\mathcal{A}$. For the i_{th} query z_i made by $\mathcal{A}$:
 - If $i \neq r_2$, we read out the next entry y_i (i.e. read the next $\log n$ bits) from the stored table, and respond with y_i . Meanwhile, we know that $H(z_i) = y_i$, and can fill in the function table of H correspondingly.
 - If $i = r_2$, we know that $H(z_{r_2}) = H(z_{r_1})$ from our previous construction. As $r_1 < r_2$, we already know $H(z_{r_1}) = y_{r_1}$ when we respond the r_1 th query. Thus we do not read anything and directly respond with y_{r_1} . Meanwhile, we fill in the entry $H(z_{r_2}) = y_{r_1}$ in the function table.
3. Receive the response α_1, α_2 from $\mathcal{A}$. Initiate a counter $j \leftarrow q$.
4. For $0 \leq i < k$ in increasing order, if $\mathcal{A}$ has not queried $H^i(\alpha_1)$ before:
 - (a) Set $j \leftarrow j + 1$. Denote $z_j = H^i(\alpha_1)$.
 - (b) If $j \neq r_2$, we read the next entry y_j , and fill in the table with $H(z_j) = y_j$.

- (c) Otherwise ($j = r_2$), we know that $H^{i+1}(\alpha_1) = H^{i+1}(\alpha_2) = y_{r_1}$ (because $r_1 < r_2$, we already know y_{r_1} at this moment), so we do not read the next entry, and fill in the table with $H(z_j) = y_{r_1}$.
- 5. For $0 \leq i < k$ in increasing order, if $\mathcal{A}$ has not queried $H^i(\alpha_2)$ before:
 - (a) Set $j \leftarrow j + 1$. Denote $z_j = H^i(\alpha_2)$.
 - (b) If $j \neq r_2$, read the next entry y_j , and fill in the table with $H(z_j) = y_j$.
 - (c) Otherwise ($j = r_2$), we again know $H^{i+1}(\alpha_1) = H^{i+1}(\alpha_2) = y_{r_1}$, so we do not read the next entry, and fill in the table with $H(z_j) = y_{r_1}$.
- 6. Continue reading out the remaining entries of the table, and fill in the function table of H in lexicographic order.

As long as $\mathcal{A}$ wins the $\text{CR}^{k,q,H}(\mathcal{A})$ game, it is not hard to see that our algorithms will correctly restore the whole function table of H .

Now we look at the number of bits saved. For all but one entry in the function table, we use $\log n$. Hence we do not save anything compared to storing the table plainly. However, for the removed entry in encoding step 6, instead of using $\log n$ bits to store it plainly, we only use $2\log(q + 2k)$ bits (because we only stored r_1 and r_2). Hence, we can save $\log n - 2\log(q + 2k)$ bits. From Lemma 2, this can happen with probability at most $\frac{(q+2k)^2}{n}$, which also gives an upper bound on the advantage $\text{Adv}_{k,q,H}^{\text{CR}}(\mathcal{A})$ as well. $\square$

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Advances in Cryptology — CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18–22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer (1996). https://doi.org/10.1007/3-540-68697-5_1
2. Bellare, M., Pietrzak, K., Rogaway, P.: Improved security analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_32
3. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 312–329. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_19
4. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O’Hearn, Z.: SPHINCS: Practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 368–397. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_15
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS⁺ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2129–2146. ACM Press (Nov 2019). <https://doi.org/10.1145/3319535.3363229>
6. Bhaumik, R., Datta, N., Dutta, A., Mouha, N., Nandi, M.: The iterated random function problem. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 667–697. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70697-9_23

7. Biryukov, A., Dinu, D., Khovratovich, D.: Argon2: New generation of memory-hard functions for password hashing and other applications. In: 2016 IEEE European Symposium on Security and Privacy. pp. 292–302. IEEE Computer Society Press (Mar 2016). <https://doi.org/10.1109/EuroSP.2016.31>
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
9. Buterin, V.: A next-generation smart contract and decentralized application platform. <https://ethereum.org/en/whitepaper/> (2014)
10. De, A., Trevisan, L., Tulsiani, M.: Time space tradeoffs for attacks against one-way functions and PRGs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 649–665. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_35
11. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 473–495. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56614-6_16
12. Dodis, Y., Ristenpart, T., Steinberger, J.P., Tessaro, S.: To hash or not to hash again? (In)differentiability results for H^2 and HMAC. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 348–366. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_21
13. Dods, C., Smart, N.P., Stam, M.: Hash based digital signature schemes. In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 96–115. Springer, Berlin, Heidelberg (Dec 2005). https://doi.org/10.1007/11586821_8
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO’86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
15. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT ’89. pp. 329–354. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
16. Haller, N., Metz, C., Nesser, II, P.J., Straw, M.: RFC 2289: A One-Time Password System. Internet Activities Board (Feb 1998)
17. Hästad, J., Näslund, M.: Practical construction and analysis of pseudo-randomness primitives. *Journal of Cryptology* **21**(1), 1–26 (Jan 2008). <https://doi.org/10.1007/s00145-007-9009-3>
18. Hülsing, A.: W-OTS+ - shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 13. LNCS, vol. 7918, pp. 173–188. Springer, Berlin, Heidelberg (Jun 2013). https://doi.org/10.1007/978-3-642-38553-7_10
19. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Goldwasser, S. (ed.) Advances in Cryptology — CRYPTO ’88. pp. 8–26. Springer New York, New York, NY (1990)
20. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M. (ed.) Advances in Cryptology — CRYPTO 2004. pp. 306–316. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
21. Kaliski, B.: Pkcs 5: Password-based cryptography specification version 2.0. RFC 2898, IETF (September 2000). <https://doi.org/10.17487/RFC2898>, <https://www.rfc-editor.org>

22. Kelsey, J., Kohno, T.: Herding hash functions and the nostradamus attack. In: Vaudenay, S. (ed.) *Advances in Cryptology — EUROCRYPT 2006*. pp. 183–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
23. Kelsey, J., Schneier, B.: Second preimages on n -bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) *Advances in Cryptology — EUROCRYPT 2005*. pp. 474–490. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
24. Kogan, D., Manohar, N., Boneh, D.: T/key: Second-factor authentication from secure hash chains. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) *ACM CCS 2017*. pp. 983–999. ACM Press (Oct / Nov 2017). <https://doi.org/10.1145/3133956.3133989>
25. Lamport, L.: Constructing digital signatures from a one-way function. Technical report csl-98, SRI International, Computer Science Laboratory (1979)
26. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) *CRYPTO'89*. LNCS, vol. 435, pp. 218–238. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_21
27. Minaud, B., Seurin, Y.: The iterated random permutation problem with applications to cascade encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) *CRYPTO 2015, Part I*. LNCS, vol. 9215, pp. 351–367. Springer, Berlin, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_17
28. Mittelbach, A., Fischlin, M.: *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Springer (2021). <https://doi.org/10.1007/978-3-030-63287-8>
29. Moriarty, K., Kaliski, B., Jonsson, J., Rusch, A.: PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017 (Nov 2016). <https://doi.org/10.17487/RFC8017>, <https://www.rfc-editor.org/info/rfc8017>
30. Moriarty, K., Kaliski, B., Rusch, A.: PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018 (Jan 2017). <https://doi.org/10.17487/RFC8018>, <https://www.rfc-editor.org/info/rfc8018>
31. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
32. Nandi, M.: A simple and unified method of proving indistinguishability. In: Barua, R., Lange, T. (eds.) *INDOCRYPT 2006*. LNCS, vol. 4329, pp. 317–334. Springer, Berlin, Heidelberg (Dec 2006). https://doi.org/10.1007/11941378_23
33. Percival, C., Josefsson, S.: The scrypt Password-Based Key Derivation Function. RFC 7914 (Aug 2016). <https://doi.org/10.17487/RFC7914>, <https://www.rfc-editor.org/info/rfc7914>
34. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) *CRYPTO'95*. LNCS, vol. 963, pp. 1–14. Springer, Berlin, Heidelberg (Aug 1995). https://doi.org/10.1007/3-540-44750-4_1
35. Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology* 4(3), 161–174 (Jan 1991). <https://doi.org/10.1007/BF00196725>
36. Wagner, D., Goldberg, I.: Proofs of security for the Unix password hashing algorithm. In: Okamoto, T. (ed.) *ASIACRYPT 2000*. LNCS, vol. 1976, pp. 560–572. Springer, Berlin, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_43
37. Yao, F.F., Yin, Y.L.: Design and analysis of password-based key derivation functions. *IEEE Trans. Inf. Theory* 51(9), 3292–3297 (2005). <https://doi.org/10.1109/TIT.2005.853307>, <https://doi.org/10.1109/TIT.2005.853307>

A Deferred proofs

A.1 Preimage Resistance Advantage Upper Bound for Random Oracle

Proof of Claim 3. Denote by $z_1, z_2, \dots, z_q$ the queries made by $\mathcal{A}$. From Assumption 3, if $\mathcal{A}$ wins the game, then there exists an index i such that $z_i = x$. From Assumption 2, this index i is unique. We enumerate this index i where $z_i = x$. (It also means for all $i' < i$ we have $z_{i'} \neq x$.)

Now we focus on the distribution of x after the first $i - 1$ queries. Since x is randomly sampled from $[n]$ at the beginning, given that $z_{i'} \neq x$ for all $i' < i$, we claim that x is still uniformly distributed among the set $[n] \setminus \{z_1, \dots, z_{i-1}\}$ (and is independent of the challenge y and the previous responses $H(z_{i'})$). This is due to the lazy sampling property of the random oracle H : First, the challenge $y = H(x)$ is sampled uniformly in $[n]$, so y is independent of x . Next, for all $i' < i$, from the condition $z_{i'} \neq x$ and Assumption 2 that $\mathcal{A}$ does not make repeated queries, we know $z_{i'}$ is a fresh query to H , so its response $H(z_{i'})$ is sampled uniformly in $[n]$, and thus is independent of x .

As a result, to achieve $z_i = x$, $\mathcal{A}$ has to correctly guess x among these $n - i + 1$ numbers in $[n] \setminus \{z_1, \dots, z_{i-1}\}$. Therefore, we immediately have

$$\Pr \left[(z_i = x) \mid \left(\bigwedge_{i' < i} (z_{i'} \neq x) \right) \right] \leq \frac{1}{n - i + 1}.$$

And thus

$$\begin{aligned} \text{Adv}_{q,H}^{\text{PR1}}(\mathcal{A}) &= \sum_{i=1}^q \Pr \left[(z_i = x) \wedge \left(\bigwedge_{i' < i} (z_{i'} \neq x) \right) \right] \\ &= \sum_{i=1}^q \Pr \left[(z_i = x) \mid \left(\bigwedge_{i' < i} (z_{i'} \neq x) \right) \right] \cdot \Pr \left[\left(\bigwedge_{i' < i} (z_{i'} \neq x) \right) \right] \\ &\leq \sum_{i=1}^q \Pr \left[(z_i = x) \mid \left(\bigwedge_{i' < i} (z_{i'} \neq x) \right) \right] \cdot 1 \\ &\leq \sum_{i=1}^q \frac{1}{n - i + 1} \leq \sum_{i=1}^q \frac{1}{n - q} \leq \frac{2q}{n}, \end{aligned}$$

where the last inequality is from $\frac{q}{n} \leq \frac{1}{2}$ (so $n - q \geq \frac{n}{2}$). $\square$