# Non-Adaptive Cryptanalytic Time-Space Lower Bounds via a Shearer-like Inequality for Permutations

Itai Dinur* Nathan Keller† Avichai Marmor‡

**Abstract**

The power of adaptivity in algorithms has been intensively studied in diverse areas of theoretical computer science. In this paper, we obtain a number of sharp lower bound results which show that adaptivity provides a significant extra power in cryptanalytic time-space tradeoffs with (possibly unlimited) preprocessing time.

Most notably, we consider the discrete logarithm (DLOG) problem in a generic group of $N$ elements. The classical 'baby-step giant-step' algorithm for the problem has time complexity $T = O(\sqrt{N})$, uses $O(\sqrt{N})$ bits of space (up to logarithmic factors in $N$) and achieves constant success probability.

We examine a generalized setting where an algorithm obtains an advice string of $S$ bits and is allowed to make $T$ arbitrary non-adaptive queries that depend on the advice string (but not on the challenge group element for which the DLOG needs to be computed).

We show that in this setting, the $T = O(\sqrt{N})$ online time complexity of the baby-step giant-step algorithm cannot be improved, unless the advice string is more than $\Omega(\sqrt{N})$ bits long. This lies in stark contrast with the classical adaptive Pollard's rho algorithm for DLOG, which can exploit preprocessing to obtain the tradeoff curve $ST^2 = O(N)$. We obtain similar sharp lower bounds for the problem of breaking the Even-Mansour cryptosystem in symmetric-key cryptography and for several other problems.

To obtain our results, we present a new model that allows analyzing non-adaptive preprocessing algorithms for a wide array of search and decision problems in a unified way.

Since previous proof techniques inherently cannot distinguish between adaptive and non-adaptive algorithms for the problems in our model, they cannot be used to obtain our results. Consequently, we rely on information-theoretic tools for handling distributions and functions over the space $S_N$ of permutations of $N$ elements. Specifically, we use a variant of Shearer's lemma for this setting, due to Barthe, Cordero-Erausquin, Ledoux, and Maurey (2011), and a variant of the concentration inequality of Gavinsky, Lovett, Saks and Srinivasan (2015) for read-$k$ families of functions, that we derive from it. This seems to be the first time a variant of Shearer's lemma for permutations is used in an algorithmic context, and it is expected to be useful in other lower bound arguments.

*Ben-Gurion University and Georgetown University. dinuri@bgu.ac.il. Supported by a gift to Georgetown University.

†Bar-Ilan University. nathan.keller27@gmail.com. Partially supported by the Israel Science Foundation (grants no. 2669/21 and 2456/25) and by the US-Israel Binational Science Foundation (grant no. 2024120).

‡Bar-Ilan University. avichai@elmar.co.il. Partially supported by the Israel Science Foundation (grants no. 2669/21 and 2456/25) and by the Bar Ilan University's Presidential Scholarship.

# 1 Introduction

## 1.1 Background

**Cryptanalytic time-space tradeoffs.** We consider cryptanalytic time-space tradeoffs in the *preprocessing* (i.e., non-uniform) model. An algorithm in this model is divided into two phases. In the first phase, the preprocessing algorithm obtains access to an input defined by the setting of the problem (e.g., $f : [N] \to [N]$ in the function inversion problem), and produces an 'advice string' of $S$ bits. In the second phase, the online algorithm receives a specific challenge it has to solve with respect to the given input (e.g., an output of the function to invert). The online algorithm is given the advice string, along with oracle access to the input. The complexity of the algorithm is evaluated by the bit-length $S$ of the advice and the time complexity $T$ of the online phase, typically measured in terms of the number of oracle queries. Such an algorithm is called an $(S, T)$-algorithm.

The preprocessing model is very natural from a practical point of view, as in many scenarios, the adversary is willing to solve many specific instances of the problem and the cost of a one-time preprocessing is amortized over the multiple solved instances.

The first and most famous cryptanalytic algorithm in the preprocessing model is Hellman's algorithm [Hel80] for inverting a random function. For constant success probability, Hellman's algorithm obtains a time-space tradeoff of about $TS^2 = \tilde{O}(N^2)$ (where $\tilde{O}$ hides logarithmic factors in $N$) on a domain of size $N$. The algorithm was extended for inverting any function by Fiat and Naor [FN99].

In recent years, there has been a significant effort in the cryptographic community to understand the power of non-uniform algorithms by devising new algorithms and proving time-space lower bounds against them. The emphasis has been on proving lower bounds for various problems (see, for example, [DGK17, CDG18, CDGS18, CK18, CK19, BMZ19, GGH$^+$20, CHM20, CGLQ20, ACDW20, GLLZ21, FGK22, GGPS23, ABG$^+$24, AGL24, GGK24, ABC$^+$25]).

**Non-adaptive cryptanalytic time-space tradeoffs.** An algorithm with oracle access to an input is called *adaptive* if its oracle queries may depend on the outcome of its previous queries. Otherwise it is called non-adaptive. In this paper, we will be interested in non-adaptive cryptanalytic algorithms receiving advice.

Almost all best-known cryptanalytic time-space tradeoff algorithms are adaptive. This includes (for example) Hellman's algorithm [Hel80] and the Fiat and Naor [FN99] algorithm for function inversion, and the algorithms by Mihalcik [Mih10], Bernstein and Lange [BL13] and Corrigan-Gibbs and Kogan [CK18] for discrete log (DLOG).

However, in most cases there is no proof that non-adaptive algorithms cannot perform as well as the best known adaptive ones. For example, for the function inversion problem, the best-known lower bound of $ST \geq \tilde{\Omega}(N)$ is far from the best-known tradeoff of $TS^2 = \tilde{O}(N^2)$. Yet, for non-adaptive algorithms, the gap is even larger, as the lower bound remains roughly $ST \geq \tilde{\Omega}(N)$ (see [GGK24]), but the best-known algorithm cannot do better than $\max(T, S) = \tilde{O}(N)$.

In 2019, Corrigan-Gibbs and Kogan [CK19] formally raised the question of the power of adaptivity in the context of the function inversion problem. The authors of [CK19] argued that the lack of progress in establishing such lower bounds can be explained by a classical barrier in complexity theory. Indeed, they showed that a significantly improved lower bound on strongly non-adaptive algorithms for function inversion would imply a circuit lower bound against Boolean circuits of linear size and logarithmic depth, thus resolving a 50-year old well-known open question of Valiant [Val77]. Later on, it was shown that such a lower bound would also imply significant data structure lower bounds [DKKS21, GGH$^+$20]. Following the work of Corrigan-Gibbs and Kogan [CK19], the power of adaptivity in the context of the function inversion problem was studied in a number of papers [CHM20, DKKS21, GGK24, GGPS23].

**Our work.** In this paper, we investigate the question of [CK19] in a more general context:

What is the power of adaptivity in non-uniform cryptanalytic algorithms?

We present sharp lower bound proofs which demonstrate the significant power of adaptivity in a variety of cryptographic problems, in the preprocessing model. More specifically, in the context of public-key cryptography, we present sharp non-adaptive time-space lower bounds for the generic DLOG problem and the square-DDH (decisional Diffie-Hellman) problem. We also prove a lower bound for DDH that is sharp in a part of the range. In the context of symmetric-key cryptography, we present sharp time-space lower bounds for key-recovery attacks against the classical Even-Mansour cipher.

From a technical viewpoint, we show in Section 1.4 and Appendix A that previous lower bound proof techniques for the problems we consider are inherently limited and cannot distinguish between adaptive and non-adaptive algorithms.

**Definition of adaptivity in our model.** In the problems we analyze, each query of the online adversary is first specified in a *pre-translated* form and then processed by a translation function before it is passed to the oracle. Our notion of non-adaptivity allows the pre-translated queries of the online algorithm to depend on the advice string, but not on the challenge itself.

Importantly, after translation, the resulting oracle queries *may depend on the challenge*. For example, in the DLOG setting (see details below), the challenge is an element $x = g^d$ for an unknown exponent $d$, and a pre-translated query is specified by a pair $(a, b)$ chosen by the adversary. This pair is then converted into the group element $x^a g^b = g^{ad+b}$, so although $(a, b)$ itself is chosen independently of the challenge, the actual oracle query still depends on the hidden value $d$.

The implicit dependence on the challenge allows our notion of non-adaptivity to capture classical algorithms such as baby-step giant-step for DLOG (see below). This is in contrast to the function inversion problem, where requiring the queries to be independent of the challenge trivializes the problem. Our lower bounds show that this class of non-adaptive algorithms cannot be improved even if we allow an arbitrary advice string (of length $S$ bits), arbitrary non-adaptive queries (that may depend on the advice), and unlimited preprocessing time.

As we argue towards the end of the Introduction, extending our proof to handle challenge-dependent pre-translated queries seems to require proving new types of concentration inequalities.

Before presenting our results, we elaborate on the rich history of the problems we analyze.

**The generic discrete logarithm problem.** Given a group $G$ of $N$ elements with a fixed generator $g$, the DLOG problem in $G$ asks for solving the equation $g^d = x$, for a given $x \in G$. It is widely believed that the problem is hard if the group is chosen carefully, and the security of various cryptosystems relies on the hardness of this problem and its variants. In 1997, Shoup [Sho97] presented the *generic group model* (GGM), intended for proving lower bounds against a specific class of algorithms that are applicable to any group of a particular order. A major motivation for studying this model is that in some cryptographically relevant groups (such as some elliptic curve groups), the best-known algorithms for DLOG are generic.

The model uses an auxiliary set $\mathcal{W}$ of bit strings with $|\mathcal{W}| \geq N$. In the model, the elements of $G$ are 'hidden' by mapping $g^j$ to $\sigma(j)$, where $\sigma : \mathbb{Z}_N \to \mathcal{W}$ is a randomly chosen bijective function. (We show in the sequel that there is no loss of generality in stating our results with $\mathcal{W} = [N] = \{1, 2, \ldots, N\}$, and so we state them this way.) An algorithm for solving the problem receives as inputs the encodings $\sigma(1), \sigma(d)$ of $g, x$, and its goal is to find $d$. The oracle queries allowed for the algorithm are of the form $\sigma(a \cdot d + b \bmod N)$.[1] The complexity of the algorithm

[1]The original model of Shoup is slightly different but it is easy to show that this does not affect the analysis.

is the total number of queries it makes, and its success probability is the expectation of the probability that it outputs $d$, over all uniform choices of $\sigma, d$, and the internal randomness of the algorithm. In this model, Shoup proved that if $N$ is prime, then the success probability of any algorithm for DLOG that makes $T$ queries is at most $T^2/N$.

The $\Omega(\sqrt{N})$ lower bound asserted by Shoup on the complexity of any algorithm for DLOG in the GGM is matched by the classical *baby-step giant-step* algorithm proposed by Shanks [Sha71] in 1971. The algorithm is based on writing the equation $g^d = x$ in the form $g^{im+j} = x$, where $m = \lceil \sqrt{N} \rceil$ and $0 \leq i, j < m$, which in turn can be written as $g^j = x(g^{-m})^i$. The algorithm computes $g^j$ for $1 \leq j \leq m$ and stores the values in a table. Then, it computes $y = g^{-m}$ and tries to find matches between $x, xy, xy^2, \ldots$ and values in the table. Once a match of the form $g^j = x(g^{-m})^i$ is found, we know that $d = im + j$ is the solution of the equation $g^d = x$, that is, the discrete logarithm of $x$. The algorithm can be executed in the generic group model, by replacing each computation of $g^j$ by the query $\sigma(j)$ and each computation of $xy^i$ by the query $\sigma(im + d \bmod N)$. In particular, this shows that baby-step giant-step is non-adaptive in our sense: the online algorithm fixes in advance, independently of the challenge, the coefficient pairs of the form $(0, j)$ and $(1, -im)$, that yield the required oracle queries. The algorithm uses memory $O(\sqrt{N})$ words of $O(\log N)$ bits, and its query complexity is $O(\sqrt{N})$.

In 1978, Pollard [Pol78] introduced the *Pollard Rho* algorithm for DLOG, which obtains the same query complexity of $O(\sqrt{N})$ while using only a very small amount $\tilde{O}(1)$ of memory. The algorithm is based on finding collisions of the form $x^a g^b = x^{a'} g^{b'}$ and using them to find a solution of the equation $g^d = x$ via the extended Euclid's algorithm. The collisions are found using Floyd's cycle finding algorithm [Knu69], which finds a collision in a random function $f : [N] \to [N]$ in time $O(\sqrt{N})$, using only a few memory cells. This algorithm can also be executed in the GGM, as computations of $x^a g^b$ correspond to queries of the form $\sigma(a \cdot d + b \bmod N)$, and the rest of the algorithm is based on comparisons.

The baby-step giant-step algorithm is clearly non-adaptive. In contrast, Pollard's Rho algorithm is highly dependent on adaptivity using Floyd's algorithm, where each query depends on the output of the previous one.

**Time-space tradeoffs for the generic DLOG problem with preprocessing.** The generic group model is naturally extended to algorithms with preprocessing. In this case, an algorithm $A$ consists of a 'preprocessing' algorithm $A_0$ and an 'online' algorithm $A_1$. The algorithm $A_0$ has direct access to the encoding function $\sigma$ (but not to the input $\sigma(d)$) and produces an advice string $z$. The algorithm $A_1$ is defined as in the standard generic group model described above, but also receives the advice string $z$ as input.

The baby-step giant-step algorithm can be naturally viewed as an algorithm with preprocessing, as its first stage of computing $\{g^j : 0 \leq j < m\}$ can be done before the value of $x$ is known. However, it does not seem to fully exploit the power of preprocessing, since the time complexity of the preprocessing phase is equal to its space complexity.

Pollard's Rho algorithm does not use preprocessing. Yet, as noted above, Mihalcik [Mih10], Bernstein and Lange [BL13] and Corrigan-Gibbs and Kogan [CK18] presented generic algorithms that solve DLOG with preprocessing with high probability whose time and space complexity satisfy $ST^2 \leq \tilde{O}(N)$. In particular, if space of $\tilde{O}(N^{1/3})$ is allowed, then the online time complexity can be reduced to $O(N^{1/3})$. These algorithms are known to be best possible [CK18].

**The Decisional Diffie Hellman (DDH) and the square-DDH (sqDDH) problems.** The DDH problem in the GGM asks to distinguish between the triples $(\sigma(x), \sigma(y), \sigma(xy))$ and $(\sigma(x), \sigma(y), \sigma(z))$, where $x, y, z \in G$ are random elements. This corresponds to the situation in the Diffie-Hellman key exchange scheme (and many other cryptosystems) where the adversary sees the elements $g^x, g^y$ sent between the parties and wants to decide whether a candidate element is the shared secret key $g^{xy}$ or a random element $g^z$. When introducing the GGM,

Shoup [Sho97] showed that for a prime $N$, any algorithm for the DDH problem in the GGM with $|G| = N$ that makes $T$ queries, has a success probability of at most $\frac{1}{2} + \frac{T^2}{N}$.

The sqDDH problem [MW99] is the special case of DDH in which $x = y$. Namely, it asks to distinguish between the pairs $(\sigma(x), \sigma(x^2))$ and $(\sigma(x), \sigma(z))$, where $x, z \in G$ are uniform elements. While there are groups in which sqDDH is significantly easier than DDH (see [JN03]), it can be shown that in the GGM, any algorithm for sqDDH that makes $T$ queries, has a success probability of at most $\frac{1}{2} + \frac{T^2}{N}$, just like for DDH (see [CDG18]).

As in the case of the DLOG problem, the complexity of algorithms for DDH and sqDDH can be significantly reduced if preprocessing is allowed. Indeed, for the DDH problem with preprocessing, the generic algorithms of Mihalcik [Mih10], Bernstein and Lange [BL13], and Corrigan-Gibbs and Kogan [CK18] achieve success probability of $\frac{1}{2}+\tilde{\Omega}(ST^2/N)$. For the sqDDH problem, Corrigan-Gibbs and Kogan [CK18] presented a generic algorithm with preprocessing that achieves an even higher success probability of $\frac{1}{2} + \tilde{\Omega}(\sqrt{ST^2/N})$. In the other direction, Corrigan-Gibbs and Kogan [CK18] showed that any algorithm with preprocessing for the DDH problem or for the sqDDH problem in the GGM has success probability of $\frac{1}{2} + \tilde{O}(\sqrt{ST^2/N})$. Interestingly, the correct tradeoff formula for the DDH problem remained open for several years, until recently Akshima, Besselman, Guo, Xie and Ye [ABG+24] proved that it is equal to $\frac{1}{2} + \tilde{\Theta}(ST^2/N)$ by presenting a tighter proof.

**The Even-Mansour (EM) cryptosystem.** The EM cryptosystem was proposed in 1991 by Even and Mansour [EM97] as the 'simplest possible' construction of a block cipher from a public permutation. It is defined as $EM(m) = k_2 \oplus \sigma(k_1 \oplus m)$, where $k_1, k_2 \in \{0,1\}^n$ are $n$-bit keys and $\sigma : \{0,1\}^n \to \{0,1\}^n$ is a publicly known permutation. The EM cryptosystem has become a central element in block cipher constructions and the security level of EM and of its iterative variants was studied extensively (see, e.g., [CLL+18, CLS15, DSST17]).

An adversary is allowed to make encryption/decryption queries to $EM$ and public queries to $\sigma$ or $\sigma^{-1}$. We consider key-recovery attacks which aim at recovering $(k_1, k_2)$. Even and Mansour showed that if $\sigma$ is chosen uniformly at random, then such an attack which makes $T_2$ encryption/decryption queries to $EM$ and $T_1$ queries to $\sigma$ or $\sigma^{-1}$, has a success probability of $O(T_1T_2/2^n)$. The work [DKS15] showed that the same holds for the seemingly weaker *Single-key EM* cryptosystem, defined as $SEM(m) = k_1 \oplus \sigma(k_1 \oplus m)$ (i.e., EM with $k_2 = k_1$).

Both adaptive and non-adaptive attacks matching these bounds are known [BW00, Dae91, DKS15]. Like in the case of DLOG, it was shown that the online complexity can be reduced significantly in the preprocessing model. Specifically, a key recovery attack with complexity $S = T_1 = T_2 = \tilde{O}(2^{n/3})$ was presented in [FJM14] and an upper bound of $\frac{1}{2}+\tilde{O}(\sqrt{S(T_1+T_2)T_2/2^n}+ T_1T_2/2^n)$ on the success probability of any distinguishing attack, which matches the attack in the constant success probability setting, was shown in [CDG18]. The previous works on EM are described in detail in Section 6.

## 1.2 Our results

**Main motivation.** The main motivation behind this work is seeking a better understanding of classical generic algorithms for DLOG and other problems. Although Pollard's algorithm for DLOG is over 45 years old, we are not aware of any formal statement that justifies its need for adaptivity. Indeed, proving such a statement seems to be related to the major open problem of proving strong time-space lower bounds against short-output oblivious branching programs,[2] for which the best-known result was obtained by Babai, Nisan and Szegedy [BNS92] more than 30 years ago. Yet, given the historical significance of Pollard's algorithm, we view the question

[2]Technically, the open problem is to prove slightly super-linear time-space tradeoff lower bounds for oblivious branching programs. In our case, since the best algorithm runs in time $O(\sqrt{N})$, the analogous problem is to prove a time-space tradeoff of the form $TS \geq N^{1/2+\epsilon}$ for some $\epsilon > 0$.

of formally justifying its adaptivity in an alternative (but standard) computational model as fundamental. We make progress on this question by showing that adaptivity does provide a significant advantage for DLOG algorithms in the generic group model, in the *preprocessing* (auxiliary input) setting. We also obtain similar results for several other major problems.

In particular, we present a new model, which we call *permutation challenge* (PC), which allows proving lower bounds on non-adaptive time-space tradeoffs for a wide variety of problems, including DLOG, DDH, sqDDH, and breaking the EM cryptosystem.[3] We describe the model as part of the proof overview in Section 1.4. We now describe the results we obtain for the specific problems.

**Sharp lower bound for non-adaptive algorithms for DLOG with preprocessing.** Our main result concerns non-adaptive generic algorithms with preprocessing for the DLOG problem. We show that unlike the adaptive setting, for non-adaptive algorithms preprocessing cannot be exploited better than in the baby-step giant-step algorithm – indeed, its $O(\sqrt{N})$ online time complexity cannot be reduced even if a preprocessing phase of unbounded complexity is allowed, as long as the size of the advice $z$ is $\tilde{O}(\sqrt{N})$.

**Theorem 1.1.** *Let $A = (A_0, A_1)$ be a non-adaptive $(S,T)$-algorithm for the DLOG problem in the generic group model, over a group $G$ with a prime number $N$ of elements. Denote by $\boldsymbol{MaxS}_{\mathrm{DLOG}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most*

$$2 \cdot \boldsymbol{MaxS}_{\mathrm{DLOG}}(T) + \frac{4\log_e(2)ST}{N} + \frac{T^2}{N} \leq \frac{3T^2}{N} + \frac{4\log_e(2)ST}{N}.$$

The bound on $\boldsymbol{MaxS}_{\mathrm{DLOG}}(T) \leq \frac{T^2}{N}$ is by Shoup's theorem [Sho97] for any non-preprocessing DLOG algorithm.

**Lower bounds for non-adaptive algorithms for DDH and sqDDH with preprocessing.** Our next results analyze decisional problems in the GGM.

**Theorem 1.2.** *Let $A = (A_0, A_1)$ be a non-adaptive $(S,T)$ algorithm for the DDH (resp. sqDDH) problem in the generic group model, over a group $G$ with a prime number $N$ of elements. Denote by $\boldsymbol{MaxS}_{\mathrm{DDH}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most*

$$\boldsymbol{MaxS}_{\mathrm{DDH}}(T) + \sqrt{\frac{2\log_e(2)ST}{N}} + \frac{T^2}{N} \leq \frac{1}{2} + \frac{2T^2}{N} + \sqrt{\frac{2\log_e(2)ST}{N}},$$

where the inequality is by Shoup's theorem [Sho97] which bounds $\boldsymbol{MaxS}_{\mathrm{DDH}}(T) \leq \frac{1}{2} + \frac{T^2}{N}$ for any non-preprocessing DDH algorithm (the bound for sqDDH follows by a similar proof).

For the sqDDH problem, the theorem is sharp, as a simple non-adaptive variant of the adaptive algorithm of [CK18] sketched in Appendix B attains its success probability bound.

For the DDH problem, we conjecture that the bound on the success probability is not sharp, and the 'right' bound is $\frac{1}{2} + \tilde{O}(\frac{T^2}{N} + \frac{ST}{N})$. Possibly, the techniques used in the recent result [ABG+24] which determined the maximal success rate of adaptive algorithms can be combined with our techniques to show this improved bound in the non-adaptive setting.

[3] It seems possible to extend the model to prove time-space lower bounds for the permutation inversion problem in case the adversary is non-adaptive and makes challenge-dependent queries of a certain type (e.g., for each challenge, the query set is selected independently uniformly at random). However, such lower bounds are unlikely to improve the state-of-the-art, hence we did not pursue this direction further.

**Sharp lower bound for non-adaptive key-recovery attacks on Even-Mansour with preprocessing.** Our last result shows that unbounded preprocessing does not allow speeding up non-adaptive key-recovery attacks on EM (unless $S \geq \tilde{\Omega}(\sqrt{N})$), in stark contrast with adaptive attacks. In the theorem, we denote by $T = T_1 + T_2$ the total number of queries the online phase of the algorithm makes.

**Theorem 1.3.** *Let $A = (A_0, A_1)$ be a key-recovery, non-adaptive $(S, T)$-adversary for the Even-Mansour cryptosystem, which can query only the public and encryption oracles and not the decryption oracle. Denote by $\boldsymbol{MaxS}_{\mathrm{EM}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most*

$$2 \cdot \boldsymbol{MaxS}_{\mathrm{EM}}(T) + \frac{4\log_e(2)S(T+1)}{N} + \frac{T^2}{N} \leq \frac{3T^2}{N} + \frac{4\log_e(2)S(T+1)}{N}.$$

*Moreover, the theorem also holds for the single-key variant where $k_1 = k_2$.*

The inequality is by [DKS15, EM97]. We remark that by symmetry of the EM construction, we may allow querying the decryption (and not the encryption) oracle. This restriction is also natural in practice, since access to encryption and access to decryption typically arise from different mechanisms, and an adversary may well obtain only one of them. The theorem is tight by the non-adaptive attacks of [BW00, Dae91, DKS15] that can be viewed as preprocessing attacks in the setting of the theorem.[4]

## 1.3 Our techniques

Our bounds are obtained using techniques from information theory, as summarized below.

**Entropy and Shearer's lemma.** For a finite set $\mathcal{X}$, the entropy of a random variable $X$ assuming values in $\mathcal{X}$ is $H(X) = \sum_{x\in\mathcal{X}} \Pr[X = x] \log_e(1/\Pr[X = x])$. The entropy, which measures the amount of uncertainty associated with the possible outcomes of $X$, is one of the central notions in information theory.

A basic property of entropy is *subadditivity*: For any random variables $X_1, X_2, \ldots, X_N$, the entropy of the Cartesian product $(X_1, X_2, \ldots, X_N)$ (which assumes values in $\mathcal{X} \times \ldots \times \mathcal{X}$) satisfies the relation $H(X_1, \ldots, X_N) \leq \sum_{i=1}^{N} H(X_i)$. Shearer's lemma [CGFS86] is a classical inequality that generalizes this property to relate the entropy of a set of random variables to the entropies of collections of their subsets.

**Theorem 1.4** (Shearer's lemma)**.** *Let $X_1, \ldots, X_N$ be random variables, and let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ be subsets of $[N] = \{1, 2, \ldots, N\}$, such that each $i \in [N]$ belongs to at least $k$ of them. Then*

$$H(X_1, \ldots, X_N) \leq \frac{1}{k} \sum_{i=1}^{m} H(X_{\mathcal{U}_i}),$$

*where for $\mathcal{U} = \{i_1, \ldots i_\ell\} \subset [N]$, $H(X_{\mathcal{U}}) := H(X_{i_1}, X_{i_2}, \ldots, X_{i_\ell})$.*

Shearer's lemma was used to obtain numerous results in combinatorics, theoretical computer science, probability theory, and other areas (see the survey [Gal14]).

[4] Similarly to baby-step giant-step for DLOG, these algorithms have a challenge-independent phase that can be performed during preprocessing, and whose outcome can be viewed as an advice string.

**KL-divergence.** In our applications, it will be convenient to use Shearer's lemma via its version for the *Kullback-Leibler (KL) divergence*, which measures the amount of dissimilarity between two distributions. For two distributions $P, Q$ such that the support of $P$ is contained in the support of $Q$, the KL divergence between $P$ and $Q$ is

$$\mathrm{D}_{\mathrm{KL}}(P\|Q) = \sum_{x \in \mathrm{Support}(P)} P(x) \log \frac{P(x)}{Q(x)}.$$

The version of Shearer's lemma for KL-divergence reads as follows:

**Theorem 1.5** (Shearer's lemma for KL-divergence [GLSS15])**.** *For a finite set $\mathcal{X}$, let $Q = (Q_1, Q_2, \ldots, Q_N)$ be the uniform distribution on $\mathcal{X}^N = \mathcal{X} \times \ldots \times \mathcal{X}$, and let $P = (P_1, \ldots, P_N)$ be another distribution on $\mathcal{X}^N$. Let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. Then*

$$k \cdot \mathrm{D}_{KL}(P\|Q) \geq \sum_{j \in [m]} \mathrm{D}_{KL}(P_{\mathcal{U}_j} \| Q_{\mathcal{U}_j}),$$

*where for $\mathcal{U} = \{i_1, \ldots i_\ell\} \subset [N]$, $P_{\mathcal{U}} = (P_{i_1}, \ldots, P_{i_\ell})$ is the marginal distribution of $P$ on $\mathcal{X}^\ell$, and analogously for $Q$.*

**The concentration inequality of [GLSS15] for families of read-$k$ functions.** One of the directions in which Shearer's lemma was applied is showing that functions on a product space which depend on 'almost-disjoint' sets of variables behave, in several aspects, similarly to independent functions. A formal manifestation of this phenomenon is the following.

**Definition 1.6.** *A family $\{f_1, f_2, \ldots, f_m\}$ of functions over a product space $\mathcal{X}^N$ is called a* read-$k$ family *if for each $1 \leq i \leq N$, at most $k$ of the functions depend on the $i$'th coordinate.*

An easy application of Shearer's lemma allows showing that if $f_1, \ldots, f_m : \{0,1\}^n \to \{0,1\}$ is a read-$k$ family with $\Pr[f_i(x) = 1] = p$ for all $i$, then $\Pr[f_1(x) = f_2(x) = \ldots = f_m(x) = 1] \leq p^{m/k}$. That is, the probability that all functions are equal to 1 simultaneously is not much larger than $p^m$ (which would be the probability if the functions were completely independent).

Gavinsky, Lovett, Saks and Srinivasan [GLSS15] used the KL-divergence variant of Shearer's lemma to prove a significantly stronger result in this direction – a variant of Chernoff's bound for read-once families of Boolean functions. Recall that for independent random variables $X_1, X_2, \ldots, X_m \in \{0,1\}$ such that $\Pr[X_i = 1] = p$, Chernoff's inequality asserts $\Pr[\sum_{i=1}^m X_i \geq (p+\epsilon)m] \leq \exp(-2\epsilon^2 m)$. The authors of [GLSS15] proved the following.

**Theorem 1.7** (The concentration bound of [GLSS15] for read-$k$ families of Boolean functions)**.** *Let $f_1, \ldots, f_m : \{0,1\}^N \to \{0,1\}$ be a read-$k$ family of functions. If $\Pr[f_i(x) = 1] = p$ for all $i$, then $\Pr[\sum_{i=1}^m f_i(x) \geq (p+\epsilon)m] \leq \exp(-2\epsilon^2 m/k)$.*

In the 10 years since its introduction, this concentration inequality has become very useful and was applied to obtain various results in TCS and combinatorics (see, e.g., [HHTT21, HS24, Kah22, KSWY25]).

**Variants of Shearer's lemma and the concentration inequality of [GLSS15] for distributions defined over permutations.** The main technical tools we use in this paper are variants of the above results for distributions defined over the space $S_N$ of permutations. It will be convenient for us to state the results for the space of bijections from $[N]$ to a set with $N$ elements, which is obviously equivalent.

The following variant of Shearer's lemma for this non-product setting is a special case of a result that was proved (in equivalent forms) by Barthe, Cordero-Erausquin, Ledoux, and Maurey [BCELM11, Proposition 21] and by Caputo and Salez [CS24, Theorem 4].

**Theorem 1.8** (Variant of Shearer's inequality for random bijections)**.** *Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. Then*

$$2k \cdot \mathrm{D}_{KL}(P_X \| Q_X) \geq \sum_{j\in[m]} \mathrm{D}_{KL}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}),$$

*where $P_{X_\mathcal{U}}$ is the distribution of $X_\mathcal{U} := (X_i \mid i \in \mathcal{U})$ with respect to $P$ (and analogously for $Q$).*

The derivation of Theorem 1.8 from [BCELM11, Proposition 21] and [CS24, Theorem 4] is presented in Appendix C.1. Since the proofs of these two results are somewhat involved, we also present an elementary proof of the theorem, albeit with a worse constant of 9, in Appendix C.2. We note that the constant 2 in the theorem cannot be improved to 1, as there are examples in which the ratio between the two sides of the inequality is $\frac{N}{N-1}$. It is not known whether the constant 2 is optimal in general.

We derive from Theorem 1.8 the following variant of the concentration inequality of Gavinsky, Lovett, Saks and Srinivasan for $k$-read families [GLSS15], for the setting of bijections.

**Theorem 1.9** (Concentration for read-$k$ families on bijections)**.** *Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\{f_j\}_{j\in[m]}$ be a read-$k$ family of functions, with $f_j : \mathcal{X}^N \mapsto [0,1]$ for all $j$. Denote $p_j = \mathbb{E}_{P_X}[f_j(X)]$ and let $p = \frac{1}{m} \cdot \sum_{j\in[m]} p_j$ be the average of the expectations. Similarly, denote $q_j = \mathbb{E}_{Q_X}[f_j(X)]$ and $q = \frac{1}{m} \cdot \sum_{j\in[m]} q_j$. Then*

$$2k \cdot \mathrm{D}_{KL}(P_X \| Q_X) \geq m \cdot \mathrm{D}_{KL}(p\|q),$$

*where $\mathrm{D}_{KL}(p\|q) = p\log(\frac{p}{q}) + (1-p)\log(\frac{1-p}{1-q})$ is the KL-divergence between two Bernoulli distributions with parameters $p$ and $q$.*

The proof of Theorem 1.9 is presented in Section 2. We remark that this theorem easily yields a concentration result in the form of Theorem 1.7 for read-$k$ families on bijections, but we use the above variant as it is more useful to us.

### 1.4 Proof overview

We begin by defining a model, referred to as *permutation challenge* (PC), for the problems analyzed in this paper. This model is inspired by the pre-sampling technique [BMZ19, CDG18, CDGS18, DGK17, Unr07] which allows bounding the success probability of preprocessing adversaries by analyzing non-preprocessing adversaries. Analogously, our model allows bounding the success probability of non-adaptive preprocessing adversaries (for a certain class of problems) by analyzing non-preprocessing adversaries.

While the pre-sampling technique is specialized to adaptive adversaries, our model allows optimizations for non-adaptive adversaries, and uses a completely different set of analytic techniques. The model abstracts away the internal details of the problem that are less relevant, and allows focusing on its most important aspects that allow proving time-space tradeoffs for non-adaptive adversaries. The fact that a single model can capture the wide array of search and decision problems we consider and allows analyzing them collectively is highly non-trivial.

**The permutation challenge model.** At a high-level, in a game in the PC model, there is an "inner permutation" $\sigma : [N] \mapsto [N]$ and a secret $d$ selected from some space. An adversary $A_1$ is given direct oracle access to $\sigma$ by issuing inner queries.

In addition, the secret $d$ is used to implement an "outer function" which wraps the inner permutation. The adversary is allowed to query the outer function, where each query is translated using $d$ into a query to $\sigma$ via a *translation function*, and the output of $\sigma$ is post-processed (again using $d$) and given back to the adversary as the oracle answer to the outer query.

The goal of $A_1$ is to "unwrap" the inner permutation (for example, by recovering the secret or a part of it) after interacting with the inner permutation via inner queries and with the outer function via outer queries.

In the preprocessing setting, $A_1$ is given an additional advice string $z = A_0(\sigma)$. The game is essentially the same, and we allow the queries of $A_1$ to depend on $z$. We say that an algorithm is non-adaptive if its inner and outer queries (before translation) depend only on $z$ (but not on evaluations of $\sigma$ on secret-dependent values).

**Instantiations.** The DLOG problem can be easily reduced to a setting where the encoding $\sigma$ is a permutation on $[N]$, and the secret is the secret discrete-log $d$ which the adversary has to compute. A query of the adversary is a pair of group elements $(a, b)$, which is mapped to the group element $a \cdot d + b \bmod N$ via a linear function applied to the discrete log. If $a \equiv 0 \pmod N$, then the query directly accesses the inner permutation, and otherwise, it is an outer query. Here, there is no post-processing of the answer of $\sigma$.

Our model further supports the DDH and sqDDH problems (although their definitions are more technical).

For the Even-Mansour construction, the inner permutation $\sigma$ is the public permutation and the adversary is allowed to query it directly. The secret is the key $(k_1, k_2)$, and the outer function corresponds to the encryption/decryption oracle. The translation function maps an outer query by XORing it with one of the keys, while the post-processing function XORs the other key to the output of $\sigma$.

**The proof.** The most important component of the model is the translation function that maps each outer query to an input of $\sigma$ using the secret $d$.

A translation function is called *uniform* if, for every fixed outer query, the translated input to $\sigma$ is distributed roughly uniformly when the secret $d$ is chosen uniformly. We show that if the translation function is uniform, then a non-adaptive preprocessing adversary obtains only a limited advantage (as a function of $S$, $T$, and a uniformity parameter) over a non-preprocessing adversary.

To explain the basic argument, consider first an online non-adaptive adversary $A_1$ that makes no inner queries, and fix a preprocessing string $z$, which fixes all queries of the adversary. We compare two distributions on the inner permutation $\sigma$: the distribution $Q$, where $\sigma$ is uniform, corresponding to the non-preprocessing setting, and the distribution $P$, where $\sigma$ is conditioned on the event $A_0(\sigma) = z$, corresponding to the preprocessing setting. For each secret value $j$, let $f_j(\sigma)$ indicate whether the adversary succeeds when the secret is $j$. The uniformity of the translation function implies that the family $\{f_j\}$ is read-$k$ for some small $k$ (which depends on the number of outer queries of $A_1$ and the uniformity parameter). Therefore, Theorem 1.9 bounds the gap between the average success probability under $P$ and under $Q$ in terms of $\mathrm{D}_{\mathrm{KL}}(P\|Q)$. Intuitively, the theorem converts the read-$k$ property of $\{f_j\}$ into a bound showing that the average success probability under $P$ cannot exceed that under $Q$ by much unless $\mathrm{D}_{\mathrm{KL}}(P\|Q)$ is large. Since the only difference between $P$ and $Q$ is that in $P$ we condition on the short advice string $z$, this establishes Lemma 4.4, which restricts the adversary to making only outer queries. A more detailed overview of this argument appears in Section 4.2.1.

Lemma 4.4 already implies bounds for general adversaries by including the answers to the inner queries in the preprocessing string (Theorem 4.2). However, as demonstrated in Remark 4.3, this approach is highly suboptimal for decision problems such as DDH and sqDDH. Therefore, we analyze the inner queries directly and more carefully in order to obtain improved

bounds (Theorem 4.1). When handling adversaries that make inner queries, a direct application of Shearer-like inequalities fails, since some permutation indices may always be queried. Nevertheless, we show that Theorem 1.9 can still be applied to the part of the permutation that has not been directly queried. The proof proceeds via a hybrid argument, based on an intermediate game between the preprocessing and non-preprocessing games, which isolates the effect of the inner queries. The reader is referred to Section 4.2 for a complete comparison between these proof strategies and the corresponding bounds.

**Comparison with other techniques.** At a high level, there are currently three main generic techniques for proving cryptanalytic time-space tradeoffs: (1) compression arguments [CK18, FGK22, GT00, GGH$^+$20, Wee05], (2) the pre-sampling technique [BMZ19, CDG18, CDGS18, DGK17, Unr07], and (3) concentration inequalities [ABG$^+$24, ACDW20, AGL24, CHM20, CGLQ20, GLLZ21, IK10].

Most of these generic techniques were devised for adaptive algorithms, and it is not clear how to optimize them for non-adaptive algorithms for the problems we consider. In Appendix A, we discuss the limitations of previous techniques in detail and argue that they cannot meaningfully distinguish between adaptive and non-adaptive algorithms in our setting: even when applied to non-adaptive adversaries, they can improve over the tight bounds known for adaptive adversaries by at most a polylogarithmic factor in $N$.

In addition to generic techniques, several papers developed specialized methods for proving time-space lower bounds for specific problems (some of which extended the generic techniques and combined additional methods). Examples of such papers include [CHM20, GGK24, GGPS23], which deal with non-adaptive algorithms for function inversion, as well as [CL23] which analyzes the 3SUM-Indexing problem. These specialized techniques seem inapplicable in our setting.

**Open problems.** The independence of the queries from the challenge appears inherent to our proof strategy. Known Shearer-like bounds control information of a random function $f$ only on sets of input–output pairs that are *fixed* and independent of $f$, namely, of the form $\{(a, f(a)) : a \in A\}$ for fixed $A$. Allowing queries to depend on the challenge $c = f(d)$ leads to composed inputs such as $f(d + f(d))$, which no longer have this fixed-set structure. The paper [RS18] derived a Shearer-type inequality that allows for some dependence, but it is too weak for our setting. Overall, extending our approach beyond this restriction would require new Shearer-type inequalities or alternative techniques that can handle such challenge-dependent queries.

More generally, a natural goal to pursue, in view of our results, is to obtain time-space lower bounds for adaptive attacks with preprocessing, as a function of *the number of adaptivity rounds* for the problems considered in this paper. For the DLOG problem, we conjecture that the success probability of an $(S, T)$-algorithm with $r$ rounds of adaptivity is at most $\tilde{O}(T^2/N + rST/N)$. This matches our result for non-adaptive algorithms (i.e., 1 round of adaptivity), as well as the bounds of [CK18] for adaptive algorithms (i.e., $T$ rounds of adaptivity). Furthermore, it would be sharp, as for any $1 \le r \le T$, it is matched by a variant of the adaptive algorithm of [BL13, CK18, Mih10], in which instead of constructing one chain of length $T$ one constructs multiple chains of length $r$.

Another open problem, mentioned above, is to close the gap between upper and lower bounds for non-adaptive $(S, T)$-algorithms for the DDH problem. As written, we conjecture that the bound of Theorem 1.2 is not tight, and the optimal bound is $\frac{1}{2} + \tilde{O}(\frac{T^2}{N} + \frac{ST}{N})$.

**Organization of the paper.** In Section 2 we present in detail the definitions and results related to entropy and KL-divergence that we will use in our proofs, including the variants of Shearer's inequality and of the inequality of Gavinsky, Lovett, Saks and Srinivasan for functions

over permutations. In Section 3 we present the permutation challenge (PC) game model and show how DLOG, DDH, sqDDH, and EM key-recovery fit into it. In Section 4 we prove our main theorem for the PC model. In Section 5 we present the bounds for the DLOG, DDH, and sqDDH problems, and in Section 6 we present the bound for attacks on the EM cryptosystem.

# 2 Information Theory

In this section we present definitions and results from information theory that will be used throughout the paper. For more background on information theory, see [CT06].

## 2.1 Definitions and notations

Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be finite sets. Let $P_{X,Y,Z}$ and $Q_{X,Y,Z}$ be probability distributions over $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Denote the projections of $P_{X,Y,Z}$ on $\mathcal{X} \times \mathcal{Y}, \mathcal{X}$, and $\mathcal{Z}$, by $P_{X,Y}, P_X$, and $P_Z$, respectively, and use similar notations for $Q_{X,Y,Z}$. A random variable $X$ assuming values in $\mathcal{X}$ is said to be drawn from the distribution $P_X$ if $\Pr[X = x] = P_X(x)$ for all $x \in \mathcal{X}$.

(a). The *entropy* of a random variable $X$ drawn from $P_X$ is

$$\mathrm{H}(X) = \mathop{\mathbb{E}}_{P_X(x)}[\log(1/P_X(x))] = \sum_{x \in \mathcal{X}} P_X(x) \log(1/P_X(x)).$$

Here and throughout the paper, all logarithms are in base $e$, unless explicitly stated otherwise. In case of ambiguity about the distribution, we may also write $\mathrm{H}(P_X)$.

(b). The *conditional entropy* of $X$ given $Y$ (drawn from $P_Y$) is

$$\mathrm{H}(X \mid Y) = \mathop{\mathbb{E}}_{P_Y(y)}[\mathrm{H}(X \mid Y = y)] = \sum_{y \in \mathcal{Y}} P_Y(y)\, \mathrm{H}(X \mid Y = y) = \mathrm{H}(X, Y) - \mathrm{H}(Y).$$

(c). The *Kullback-Leibler divergence* (KL-divergence) between two distributions $P_X, Q_X$ is

$$\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) = \mathop{\mathbb{E}}_{P_X(x)}[\log(P_X(x)/Q_X(x))] = \mathop{\mathbb{E}}_{P_X(x)}[\log(1/Q_X(x))] - \mathrm{H}(P_X),$$

where we assume that the support of $P_X$ is contained in the support of $Q_X$ (otherwise, the KL-divergence is infinite).

(d). The KL-divergence between $P_X, Q_X$ conditioned on $P_Z$ is

$$\mathrm{D}_{\mathrm{KL}}(P_{X|Z} \| Q_{X|Z}) = \mathop{\mathbb{E}}_{P_Z(z)}[\mathrm{D}_{\mathrm{KL}}(P_{X|Z=z} \| Q_{X|Z=z})].$$

(e). The KL-divergence between two Bernoulli distributions $P_X, Q_X$ with parameters $p, q$, respectively (i.e., $\mathcal{X} = \{0, 1\}$, $P_X(1) = p, P_X(0) = 1 - p$, and similarly for $Q_X$) is denoted by

$$\mathrm{D}_{\mathrm{KL}}(p \| q) = \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X).$$

(f). The *mutual information* between $X$ and $Y$ (drawn from $P_X, P_Y$) is

$$\mathrm{I}(X; Y) = \mathrm{D}_{\mathrm{KL}}(P_{X,Y} \| P_X P_Y).$$

## 2.2 Basic properties

We shall use the following basic properties of entropy, KL-divergence and mutual information.

(1) Conditioning does not increase entropy, namely

$$\mathrm{H}(X) \geq \mathrm{H}(X \mid Y),$$

with equality if and only if $X, Y$ are independent, i.e., $P_{X,Y} = P_X \times P_Y$.

(2) The *chain rule for entropy* asserts that

$$\mathrm{H}(X, Y) = \mathrm{H}(X) + \mathrm{H}(Y \mid X).$$

Thus, $\mathrm{H}(X, Y) = \mathrm{H}(X) + \mathrm{H}(Y)$ if and only if $X, Y$ are independent.

(3) The entropy of $X$ is upper-bounded by the logarithm of the size of its support, namely

$$\mathrm{H}(X) \leq \log |\operatorname{Supp}(\mathcal{X})|.$$

(4) KL-divergence is non-negative, namely

$$\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq 0.$$

(5) KL-divergence is convex, namely for $0 \leq \lambda \leq 1$,

$$\mathrm{D}_{\mathrm{KL}}(\lambda P_X + (1-\lambda)P'_X \| \lambda Q_X + (1-\lambda)Q'_X) \leq \lambda \,\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) + (1-\lambda)\,\mathrm{D}_{\mathrm{KL}}(P'_X \| Q'_X).$$

(6) If $Q_X$ is uniform over its support (which includes the support of $P_X$), then

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) &= \mathop{\mathbb{E}}_{P_X(x)}[\log(1/Q_X(x))] - \mathrm{H}(P_X) = \mathop{\mathbb{E}}_{Q_X(x)}[\log(1/Q_X(x))] - \mathrm{H}(P_X) \\ &= \mathrm{H}(Q_X) - \mathrm{H}(P_X).\end{aligned}$$

(7) The *chain rule for KL-divergence* asserts that

$$\mathrm{D}_{\mathrm{KL}}(P_{X,Y} \| Q_{X,Y}) = \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) + \mathrm{D}_{\mathrm{KL}}(P_{Y|X} \| Q_{Y|X}).$$

(8) The *data processing inequality* asserts that for a function $f : \mathcal{X} \mapsto \mathcal{Y}$,

$$\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq \mathrm{D}_{\mathrm{KL}}(P_{f(X)} \| Q_{f(X)}).$$

(9) A special case of *Pinsker's inequality* states that

$$2(p-q)^2 \leq \mathrm{D}_{\mathrm{KL}}(p \| q).$$

(10) The mutual information satisfies

$$\mathrm{I}(X;Y) = \mathrm{D}_{\mathrm{KL}}(P_{X,Y} \| P_X P_Y) = \mathrm{D}_{\mathrm{KL}}(P_{X|Y} \| P_X) = \mathrm{H}(X) - \mathrm{H}(X \mid Y) \leq \mathrm{H}(X).$$

## 2.3 Lemmas

We shall use the following lemmas.

**Proposition 2.1.** *Let $p, \epsilon > 0$ such that $p + \epsilon \leq 1$. Then,*

$$\mathrm{D}_{KL}(p + \epsilon \| p) \geq \frac{\epsilon^2}{2(p + \epsilon)}.$$

*Proof.* The proof follows a standard analytic approach for proving inequalities on KL-divergence.

By Property (9), $\mathrm{D}_{\mathrm{KL}}(p+\epsilon\|p) \geq 2\epsilon^2$, implying the statement whenever $p+\epsilon \geq \frac{1}{4}$. Therefore, we may assume $p + \epsilon \leq \frac{1}{4}$. Let us define a function $f : [0, \epsilon] \to \mathbb{R}$ by

$$f(x) = \mathrm{D}_{\mathrm{KL}}(p + x\|p) - \frac{1}{2(p + \epsilon)}x^2.$$

Since $f(0) = 0$ and the statement is equivalent to $f(\epsilon) \geq 0$, it suffices to show that $f$ is monotonically increasing. We have

$$f'(x) = \log\left(\frac{p + x}{p}\right) - \log\left(\frac{1 - (p + x)}{1 - p}\right) - \frac{1}{p + \epsilon}x.$$

Again, we may notice that $f'(0) = 0$. Therefore, in order to show that $f'(x) \geq 0$ for all $x \in [0, \epsilon]$ and finish the proof, it suffices to show that $f''(x) \geq 0$ for all $x \in [0, \epsilon]$. Since $p+\epsilon \leq \frac{1}{4}$, we obtain that for all $x \in [0, \epsilon]$ we have

$$f''(x) = \frac{1}{(1 - (p + x))(p + x)} - \frac{1}{p + \epsilon} \geq \frac{1}{(1 - (p + \epsilon))(p + \epsilon)} - \frac{1}{p + \epsilon} \geq 0,$$

completing the proof. ■

**Corollary 2.2.** *Let $0 < p, q \leq 1$. Then,*

$$p \leq 2(q + \mathrm{D}_{KL}(p\|q)).$$

*Proof.* If $p - q < q$, then $p < 2q \leq 2(q + \mathrm{D}_{\mathrm{KL}}(p\|q))$, as the KL-divergence is non-negative. Otherwise, $p - q \geq q$. Applying Proposition 2.1, we obtain

$$\mathrm{D}_{\mathrm{KL}}(p\|q) \geq \frac{(p - q)^2}{2q} \geq \frac{(p - q)}{2}.$$

Therefore, $p \leq q + 2\,\mathrm{D}_{\mathrm{KL}}(p\|q)) \leq 2(q + \mathrm{D}_{\mathrm{KL}}(p\|q))$.

■

The following proposition can be viewed as a form of the data-processing inequality.

**Proposition 2.3** ([GLSS15], Claim 2.6)**.** *Let $Q_X$ be the uniform distribution over a set $\mathcal{X}$, let $P_X$ be a distribution over $\mathcal{X}$, and let $f : \mathcal{X} \mapsto [0, 1]$. Denote $q = \mathbb{E}_Q[f(X)]$ and $p = \mathbb{E}_P[f(X)]$. Then*

$$\mathrm{D}_{KL}(P_X\|Q_X) \geq \mathrm{D}_{KL}(p\|q).$$

## 2.4 A variant of the inequality of [GLSS15] for functions over bijections

In this subsection we derive Theorem 1.9, namely, the variant of the concentration bound of Gavinsky, Lovett, Saks and Srinivasan [GLSS15] for read-$k$ families of functions over bijections, from the variant of Shearer's lemma for the same setting (Theorem 1.8 above). Let us restate these two theorems:

**Theorem 1.8.** Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. Then

$$2k \cdot \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq \sum_{j \in [m]} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}),$$

where $P_{X_{\mathcal{U}}}$ is the distribution of the vector $X_{\mathcal{U}} := (X_i \mid i \in \mathcal{U})$ with respect to $P$ (and analogously for $Q$).

**Theorem 1.9**. Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\{f_j\}_{j \in [m]}$ be a read-$k$ family of functions, with $f_j : \mathcal{X}^N \mapsto [0,1]$ for all $j$. Denote $p_j = \mathbb{E}_{P_X}[f_j(X)]$ and let $p = \frac{1}{m} \cdot \sum_{j \in [m]} p_j$ be the average of the expectations. Similarly, denote $q_j = \mathbb{E}_{Q_X}[f_j(X)]$ and $q = \frac{1}{m} \cdot \sum_{j \in [m]} q_j$. Then

$$2k \cdot \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq m \cdot \mathrm{D}_{\mathrm{KL}}(p \| q),$$

where $\mathrm{D}_{\mathrm{KL}}(p\|q) = p \log(\frac{p}{q}) + (1-p)\log(\frac{1-p}{1-q})$ is the KL-divergence between two Bernoulli distributions with parameters $p$ and $q$.

*Proof (of Theorem 1.9).* Let $\{\mathcal{U}_j\}_{j \in [m]}$ be a family of index sets such that for each $j \in [m]$, $f_j$ depends only on the coordinates in $\mathcal{U}_j$. Since for each $i \in [N]$, $|\{j \in [m] \mid i \in \mathcal{U}_j\}| \leq k$, Theorem 1.8 implies

$$2k \cdot \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq \sum_{j \in [m]} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}).$$

For every $j \in [m]$, $Q_{X_{\mathcal{U}_j}}$ is uniform over its support (which includes the support of $P_{X_{\mathcal{U}_j}}$). Thus, we apply Proposition 2.3 with $f_j$ and deduce

$$\mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}) \geq \mathrm{D}_{\mathrm{KL}}(p_j \| q_j).$$

Hence,

$$2k \cdot \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq \sum_{j \in [m]} \mathrm{D}_{\mathrm{KL}}(p_j \| q_j) = m \sum_{j \in [m]} (1/m)\, \mathrm{D}_{\mathrm{KL}}(p_j \| q_j) \geq m \cdot \mathrm{D}_{\mathrm{KL}}(p \| q),$$

where the final inequality holds by convexity of the KL-divergence (Property (5)). ■

**Remark 2.4.** *Note that while the definition of a read-k family requires the functions $\{f_j\}$ to be defined over the entire domain $\mathcal{X}^N$, only their values on bijections are relevant in Theorem 1.9. Consequently, in the applications of this theorem we define each function only over bijections, while implicitly fixing it to 0 on inputs that are not bijections.*

# 3 The Permutation Challenge Model

## 3.1 Non-preprocessing setup

A permutation challenge (PC) game

$$\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$$

is defined as follows.

For an integer $N > 0$, let $\sigma : [N] \mapsto [N]$ be a permutation and let $\mathcal{D}$ be a space of secrets.

Let $\boldsymbol{tr} : \mathcal{D} \times \mathcal{M} \to [N]$ be a translation function that obtains as inputs a secret $d \in \mathcal{D}$ (e.g., the discrete log in the DLOG problem) and an outer query $m \in \mathcal{M}$ (from the outer query space $\mathcal{M}$) and returns a translated query to $\sigma$, denoted $\boldsymbol{tr}(d, m) \in [N]$. Let $\boldsymbol{post} : \mathcal{D} \times [N] \to [N]$ be a post-processing function that receives the secret and an inner permutation output and returns a value in $[N]$ (the output space of the outer function).

**Definition 3.1.** *A translation function* $\boldsymbol{tr}$ *is called* $u$*-uniform if for every* $m \in \mathcal{M}$ *and* $j \in [N]$,

$$|\{d \in \mathcal{D} \mid \boldsymbol{tr}(d, m) = j\}| \leq \frac{|\mathcal{D}|}{u}.$$

In several of our applications, the translation function $\boldsymbol{tr}$ will be $N$-uniform, which is the largest value possible.

Let $A_1$ be an adversary for $\boldsymbol{PC}$. We assume for simplicity that $A_1$ is deterministic. As noted in Remark 3.2 below, this assumption is without loss of generality.

$A_1$ has oracle access to two related oracles. The inner oracle allows $A_1$ to issue an *inner query* $i \in [N]$ to $\sigma$ and obtain $\sigma(i)$. In some applications, $A_1$ can also issue inner queries $i \in [N]$ to the inverse inner permutation and obtain $\sigma^{-1}(i)$.

The outer oracle allows $A_1$ to issue an *outer query* $m \in \mathcal{M}$ and obtain $\boldsymbol{post}(d, \sigma(\boldsymbol{tr}(d, m)))$.

After the interaction with the oracles which we denote in short by $\mathcal{O}(\sigma, d)$, $A_1^{\mathcal{O}(\sigma,d)}$ outputs a value $v$ from some domain $\mathcal{V}$.

Let $\boldsymbol{suc} := \boldsymbol{suc}_{A_1^{\mathcal{O}(\sigma,d)}}(d)$ be a 0/1 success predicate that obtains $d$ and has the same oracle accesses as $A_1^{\mathcal{O}(\sigma,d)}$. This predicate outputs 1 if $A_1$ succeeds and 0 otherwise.

Formally, $\boldsymbol{suc}$ has two phases. In the first phase, it simulates $A_1^{\mathcal{O}(\sigma,d)}$ and obtains its output. In the second phase, it outputs a value by applying a 0/1 predicate that receives as input the secret $d$, the output of $A$, and all the query answers obtained by $A$.[5] We may also allow $\boldsymbol{suc}$ to make additional queries to verify the success of $A$, which will be accounted for in the total time complexity (but we do not use this possibility in our applications).

We remark that when we write $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$, the success predicate is viewed as an interface that is instantiated given an adversary $A_1$.

Let $P_{\Sigma,D} = P_\Sigma P_D$ be a distribution such that $P_\Sigma$ is uniform over permutations and $P_D$ is uniform over the secret space $\mathcal{D}$.

The success probability of $A_1$ is defined as

$$\mathop{\mathbb{E}}_{P_{\Sigma,D}}[\boldsymbol{suc}_{A_1^{\mathcal{O}(\Sigma,D)}}(D)].$$

We measure the complexity of $A_1$ in terms of its number of queries to the inner permutation, $T_1$, and its number of queries to the outer function, $T_2$, and define $T = T_1 + T_2$.

**Non-adaptivity.** We say that $A_1$ is non-adaptive if its queries are fixed in advance and do not depend on $\sigma$.

### 3.2 Preprocessing setup

Let $A = (A_0, A_1)$ be a pair of deterministic preprocessing and online algorithms for $\boldsymbol{PC}$.

**Remark 3.2.** *In our context, we may assume that* $A$ *is deterministic without loss of generality, as our lower bound proofs hold for every randomness string shared by* $A_0$ *and* $A_1$*.*

The preprocessing algorithm $A_0$ gets direct access to $\sigma$ as input and outputs an advice string $z = A_0(\sigma)$. The online algorithm $A_1$ takes as an additional input the advice string $z$. We denote the algorithm $A_1$, when it is executed with a preprocessing string $z$, by $(A_1)_z$.

We call $(A_0, A_1)$ an $(S, T)$-algorithm if the length of $z = A_0(\sigma)$ is bounded by $S$ bits (for all $\sigma$) and $A_1$ makes at most $T$ oracle queries to both the inner permutation and the outer function.

We extend the non-preprocessing success predicate $\boldsymbol{suc}$ to the preprocessing model as follows: it additionally obtains an advice string $z$. As previously, $\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma,d)}}(d)$ has two phases.

[5] In our applications, the predicate only needs the output of $A$, but we allow it to obtain the query answers obtained by $A$ for generality.

In the first phase, it simulates $(A_1)_z^{\mathcal{O}(\sigma,d)}$ and obtains its output. In the second phase, it outputs a value by applying the same non-preprocessing 0/1 predicate that receives as input the secret $d$, the output of $A_1$ and all the query answers obtained by $A_1$.[6]

We extend the distribution $P_{\Sigma,D}$ as $P_{\Sigma,Z,D} = P_{\Sigma,Z} \times P_D$, where $P_{\Sigma,Z} = P_\Sigma P_{Z|\Sigma}$ such that $P_{Z|\Sigma=\sigma}(z) = 1$ if $z = A_0(\sigma)$.

The success probability $(A_0, A_1)$ in solving the problem is defined as

$$p := p(A_0, A_1) := \mathop{\mathbb{E}}_{P_{\Sigma,Z,D}}[\boldsymbol{suc}_{(A_1)_Z^{\mathcal{O}(\Sigma,D)}}(D)].$$

**Remark 3.3.** *Crucially, the predicate applied by* $\boldsymbol{suc}$ *after simulating* $(A_1)_z^{\mathcal{O}(\sigma,d)}$ *does not receive* $z$ *as input. Namely,* $\boldsymbol{suc}$ *must be able to verify whether the output of* $(A_1)_z$ *is correct on* $(\sigma, d)$ independently *of* $z$*. Technically, our analysis will sometimes fix* $z$ *that is not equal to* $A_0(\sigma)$*, but we require that the success of* $(A_1)_z$ *will still be defined with respect to* $(\sigma, d)$*.*

*Another way to state this remark is that it is sufficient to define the predicate* $\boldsymbol{suc}$ *for non-preprocessing algorithms. The extension to the preprocessing model is well-defined and it simply simulates* $(A_1)_z$ *as a black-box and checks its success with respect to* $(\sigma, d)$ *as in the non-preprocessing model.*

**Non-adaptivity.** We say that $A_1$ is non-adaptive if given any $z$, its queries are fixed and do not (further) depend on $\sigma$. Note that if $A_1$ is non-adaptive then $\boldsymbol{suc} := \boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma,d)}}(d)$ is a non-adaptive predicate.

We denote by $\mathcal{S} \subseteq [N]$ the set of inner queries of $A_1$ (which may include inverse queries) and by $\mathcal{U} \subseteq \mathcal{M}$ the set of its outer queries. To simplify notation, we assume that $\mathcal{S}$ refers to queries to $\sigma$ and not $\sigma^{-1}$ using the conversion that if $\sigma^{-1}(i) = j$, then $\sigma(j) = i$ is a query to $\sigma$. Both $\mathcal{S}$ and $\mathcal{U}$ may depend on $z$, and we sometimes emphasize this by writing (for example) $\mathcal{S}(z)$.

### 3.3 Instantiation in the generic group model

We first define the general setting of GGM which is common to all problems in the model.

Let $A_1$ be a generic group algorithm in $\mathbb{Z}_N$ such that $N$ is prime. We assume that $A_1$ knows the image of $\sigma : \mathbb{Z}_N \to \mathcal{W}$, as this knowledge can only increase its success probability. Thus, we restrict $\sigma$ to a subset of $\mathcal{W}$ of size $N$. Furthermore, by renaming the symbols of the image of $\sigma$, we assume without loss of generality that $\sigma : \mathbb{Z}_N \to \mathbb{Z}_N$ is a uniform bijection from $\mathbb{Z}_N$ to itself. We represent the elements of $\mathbb{Z}_N$ using $[N]$, and we can thus write $\sigma : [N] \to [N]$. In particular, note that $N \bmod N = 0$.

We remark that in GGM, we do not allow $A_1$ to query $\sigma^{-1}$.

We now define the DLOG, DDH, and sqDDH problems as permutation challenge games.

**Discrete-log.** Let $A_1$ be a discrete-log algorithm for $\mathbb{Z}_N$. We define a permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{DL}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ for the discrete-log problem as follows.

Let $\mathcal{D} = [N]$ (identified with $\mathbb{Z}_N$) be the space of secrets of discrete logarithms. Let $\boldsymbol{post}(d, j) = j$ be the trivial post-processing function (as $A_1$ always sees direct outputs of $\sigma$).

Let $\mathcal{M} = [N-1] \times [N]$ be the space of outer queries and define

$$\boldsymbol{tr}(d, (a, b)) = a \cdot d + b \mod N$$

as the translation function that maps a query $(a, b)$ (where $a \neq N$) to a group element according to the linear function $a \cdot d + b \mod N$. Note that in the representation, an inner query issued to $\sigma$ corresponds to the pair $(N, b)$ (but we view it as directly accessing $\sigma(b)$).

[6] In our applications, the predicate only compares the output of $(A_1)_z$ to some function of the secret.

The adversary receives as input the values $\sigma(1)$ and $\sigma(d)$. However, for simplicity we assume that these are given as outputs of the queries $(N, 1)$ and $(1, N)$, and the adversary receives no input (we make similar assumptions about DDH and sqDDH, defined below).

Finally, $\boldsymbol{suc}_{A_1^{\mathcal{O}(\sigma,d)}}(d)$ simply simulates $A_1$ and returns 1 if it outputs the discrete-log secret $d$.

**DDH.** Let $A_1$ be a DDH generic algorithm for $\mathbb{Z}_N$. We define a permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{DDH}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ for the DDH problem as follows.

Let $\mathcal{D} = [N]^3 \times \{0, 1\}$ be the secret space, consisting of 3 group elements $d_1, d_2, d_3$ and a bit $k$ that $A_1$ needs to output. Let $\boldsymbol{post}(d, j) = j$ be the trivial post-processing function.

Let $\mathcal{M} = [N]^4 \setminus (\{(N, N, N)\} \times [N])$ be the space of outer queries, which consists of 4 group elements that specify a multi-linear function, denoted by $(a_1, a_2, a_3, b)$, where we do not allow $a_1 = a_2 = a_3 = N$ (as this corresponds to an inner query).

Define the translation function $\boldsymbol{tr}((d_1, d_2, d_3, k), (a_1, a_2, a_3, b))$ as follows:

$$\boldsymbol{tr}((d_1, d_2, d_3, k), (a_1, a_2, a_3, b)) = \begin{cases} a_1 \cdot d_1 + a_2 \cdot d_2 + a_3 \cdot d_3 + b \mod N & \text{if } k = 0, \\ a_1 \cdot d_1 + a_2 \cdot d_2 + a_3 \cdot (d_1 d_2) + b \mod N & \text{if } k = 1. \end{cases}$$

Note that if $k = 0$, the algorithm effectively receives $\sigma(d_1), \sigma(d_2), \sigma(d_3)$, while if $k = 1$, the algorithm effectively receives $\sigma(d_1), \sigma(d_2), \sigma(d_1 d_2 \bmod N)$, as in DDH.

As in the discrete-log game, an inner query can be viewed as directly accessing $\sigma$.

Finally, $\boldsymbol{suc}_{A_1^{\mathcal{O}(\sigma,d_1,d_2,d_3,k)}}(d_1, d_2, d_3, k)$ simply simulates $A_1$ and returns 1 if it outputs the bit $k$.

**sqDDH.** Let $A_1$ be a sqDDH generic algorithm for $\mathbb{Z}_N$. We define a permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{sqDDH}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ for the sqDDH problem as follows.

Let $\mathcal{D} = [N]^2 \times \{0, 1\}$ be the space of secrets, consisting of 2 group elements $d_1, d_2$ and a bit $k$ that $A_1$ needs to output. Let $\boldsymbol{post}(d, j) = j$ be the trivial post-processing function.

Let $\mathcal{M} = [N]^3 \setminus (\{(N, N\} \times [N])$ be the space of outer queries, which consist of 3 group elements that specify a multi-linear function, denoted by $(a_1, a_2, b)$, where we do not allow $a_1 = a_2 = N$ (as this corresponds to an inner query).

Define the translation function $\boldsymbol{tr}((d_1, d_2, k), (a_1, a_2, b))$ as follows:

$$\boldsymbol{tr}((d_1, d_2, k), (a_1, a_2, b)) = \begin{cases} a_1 \cdot d_1 + a_2 \cdot d_2 + b \mod N & \text{if } k = 0, \\ a_1 \cdot d_1 + a_2 \cdot (d_1)^2 + b \mod N & \text{if } k = 1. \end{cases}$$

Note that if $k = 0$, the algorithm effectively receives $\sigma(d_1), \sigma(d_2)$, while if $k = 1$, the algorithm effectively receives $\sigma(d_1), \sigma((d_1)^2 \bmod N)$, as in sqDDH.

Once again, an inner query is viewed as directly accessing $\sigma$.

Finally, $\boldsymbol{suc}_{A_1^{\mathcal{O}(\sigma,d_1,d_2,k)}}(d_1, d_2, k)$ simply simulates $A_1$ and returns 1 if it outputs the bit $k$.

### 3.3.1 Uniformity of the translation function in GGM.

In all the $\boldsymbol{PC}$ games in GGM defined above, the $\boldsymbol{tr}$ function is a multi-variate polynomial of degree 1 or 2, in up to 3 group elements, chosen uniformly at random from $[N]$. The coefficients of this polynomial are determined by the query. For such queries, the Schwartz–Zippel lemma [Sch80] immediately gives the following general result.

**Lemma 3.4** ($u$-uniformity of $\boldsymbol{tr}$ in the GGM)**.** *Let $\boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ be a permutation challenge game in the GGM. Assume that $\boldsymbol{tr}$ is a multi-variate polynomial (with coefficients determined by the query) of degree $m > 0$ in $v$ variables chosen uniformly at random from $[N]$ with $N$ prime. Then, $\boldsymbol{tr}$ is $\frac{N}{m}$-uniform.*

### 3.4 Instantiation for the Even-Mansour cryptosystem

We represent plaintexts, ciphertexts and keys using $[N]$, and we can thus write $\sigma : [N] \to [N]$. Using this encoding, the bit representation of any $a \in [N]$ (used when XORing values in the domain) is the standard bit representation of $a - 1$.

We define a permutation challenge game for the key recovery. Let $\mathcal{D} = [N]^2$ be the space of secrets, consisting of pairs $(k_1, k_2)$ (in case of the single-key scheme, $k_1 = k_2$ and $\mathcal{D} = [N]$). Let $\mathcal{M} = [N]$ be the space of outer queries, which are chosen encryption messages. For a secret $(k_1, k_2) \in [N]^2$ and a message (outer query) $m \in [N]$, define $\boldsymbol{tr}((k_1, k_2), m) = m \oplus k_1$, and let $\boldsymbol{post}((k_1, k_2), j) = j \oplus k_2$.

Note that an inner query that is issued to $\sigma$ corresponds to a call to the public encryption oracle. Here, we allow the adversary to query $\sigma^{-1}$ as well.

Let $A_1$ be a key-recovery algorithm for EM. For the permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{EM-KR}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$, the success predicate $\boldsymbol{suc}_{A_1^{\mathcal{O}(\sigma,(k_1,k_2))}}(k_1, k_2)$ simply simulates $A_1$ and returns 1 if it outputs the key $(k_1, k_2)$.

## 4 Main General Result

In this section we prove our main result in the permutation challenge game model.

**Theorem 4.1.** *Let $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ be a permutation challenge game, such that $\boldsymbol{tr}$ is $u$-uniform, and $\boldsymbol{suc}$ compares the output of $(A_1)_z$ to some function of the secret. Let $(A_0, A_1)$ be an $(S, T)$ non-adaptive algorithm with preprocessing for $\boldsymbol{PC}$. Denote by $\boldsymbol{MaxS}(T)$ the optimal success probability (with respect to $\boldsymbol{suc}$) of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $(A_0, A_1)$ is at most*

$$\min\left(2 \cdot \boldsymbol{MaxS}(T) + \frac{4\log(2)ST}{u} + \frac{T^2}{u}, \boldsymbol{MaxS}(T) + \sqrt{\frac{\log(2)ST}{u}} + \frac{T^2}{2u}\right).$$

We note that a slightly better bound can be achieved for problems with a trivial post-processing function $\boldsymbol{post}(d, j) = j$. For further details, see Appendix D.

### 4.1 Discussion

Before proving the theorem, several important remarks are due.

**No need to analyze the preprocessing setting.** The power of the theorem comes from the fact that there is no need to analyze the preprocessing setting. Indeed, it suffices to instantiate the permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$, to prove that $\boldsymbol{tr}$ is a $u$-uniform translation function (ideally, for $u \geq \Omega(N)$), and to bound $\boldsymbol{MaxS}(T)$ (i.e., the success probability in the non-preprocessing setting). Plugging these values into Theorem 4.1 immediately bounds the success probability of an adversary in the preprocessing setting.

**Generality of *suc*.** We prove the theorem only for a limited class of success predicates that compare the output of $(A_1)_z$ to some function of the secret. Yet, almost all the steps of the proof apply to arbitrary non-adaptive success predicates (under the restrictions defined in Section 3), where the only exception that uses this restriction is Lemma 4.12. This lemma deals with a specific game in the non-preprocessing setting, and it is not possible to prove it in general for all predicates, as its statement is false for some artificial predicates.

However, for all "natural" predicates we are aware of, it is easy to extend the theorem tightly by extending the proof of Lemma 4.12 accordingly. This may require extending $\boldsymbol{suc}$ to make

additional non-adaptive queries, which are accounted for in the parameter $T$. For example, it is possible to support selective forgery attacks, where the goal of the adversary is to predict the value of a predefined outer query (that the adversary is not allowed to make). This is done by extending $\boldsymbol{suc}$ to make this additional (non-adaptive) outer query, thus adjusting the total query complexity to $T+1$. Now, $\boldsymbol{suc}$ verifies success by comparing the outcome of the query to the output of $(A_1)_z$. The proof is then adjusted by proving a corresponding variant of Lemma 4.12.

**$\boldsymbol{tr}$ vs. $\boldsymbol{post}$, and treatment of inner queries to $\sigma^{-1}$.** The $u$-uniformity of $\boldsymbol{tr}$ is crucially used in the proof. On the other hand, there are no additional requirements on the function $\boldsymbol{post}$, and indeed it does not play any direct role in the proof. Its relevance will be in applications of this theorem (in particular, it is important for bounding $\boldsymbol{MaxS}(T)$). Similarly, the proof holds regardless of whether $A_1$ can issue inner queries to $\sigma^{-1}$ in addition to $\sigma$, yet this fact may be important for bounding $\boldsymbol{MaxS}(T)$.

## 4.2 Warm-up: a weaker bound

**Inner and outer queries.** The proof of Theorem 4.1 is built from two complementary ideas, which we present separately to keep the exposition modular. The first idea applies directly only to adversaries that make *no inner queries* (Lemma 4.4); as shown in Theorem 4.2, it can nevertheless be applied to obtain a bound for general adversaries, albeit a weaker one. The second idea establishes a subtle reduction from general adversaries to adversaries without inner queries that preserves the parameters more tightly. Combining this reduction with the first idea yields the stronger bound of Theorem 4.1, proved in Section 4.3.

Intuitively, a permutation variant of Shearer's lemma (Theorem 1.8) provides a natural tool for quantifying the (limited) advantage gained from outer queries. The two arguments therefore share the same Shearer-based core; they differ only in the way they handle the inner queries—the weaker bound neutralizes them in a straightforward manner, while the stronger bound controls them more delicately through the reduction.

**Theorem 4.2** (Weaker version of Theorem 4.1)**.** *In the setting of Theorem 4.1, the success probability of $(A_0, A_1)$ is at most*

$$\min\left(2\cdot \boldsymbol{MaxS}(T) + \frac{4\log(2)T(S+T\log(N))}{u}, \boldsymbol{MaxS}(T) + \sqrt{\frac{\log(2)T(S+T\log(N))}{u}}\right).$$

**Remark 4.3.** *Comparison between Theorems 4.1 and 4.2: Theorems 4.1 and 4.2 may look similar, and they indeed obtain comparable bounds for several problems. For example, for search problems (such as in Theorem 1.1 and Theorem 1.3), Theorem 4.2 yields the bound $O(\frac{(S+T\log N)\cdot T}{N} + \frac{T^2}{N})$, instead of the $O(\frac{S\cdot T}{N} + \frac{T^2}{N})$ bound of Theorem 4.1, which constitutes only a logarithmic penalty. On the other hand, for applications of Theorem 4.2 to decision problems it yields much weaker results. Specifically, for Theorem 1.2, we would obtain*

$$\frac{1}{2} + O\left(\sqrt{\frac{(S+T\log N)\cdot T}{N}} + \frac{T^2}{N}\right), \qquad \textit{instead of} \qquad \frac{1}{2} + O\left(\sqrt{\frac{S\cdot T}{N}} + \frac{T^2}{N}\right),$$

*which is significantly worse for $S \ll T \ll \sqrt{N}$. For example, for $S = O(1)$ and $T = O(N^{1/3})$, the bound of Theorem 4.2 on the adversary's advantage over a coin toss is larger by a multiplicative factor of $N^{1/6}$. This motivates our more involved proof that handles the inner queries in a novel way.*

Theorem 4.2 follows easily from the following variant, which forces the adversary to make only outer queries:

**Lemma 4.4** (Restricted variant of Theorem 4.1)**.** *In the setting of Theorem 4.1, suppose that $A_1$ makes no inner queries. In addition, let* $\boldsymbol{suc}$ *be an arbitrary success predicate, as defined in Section 3. Then, the success probability of $(A_0, A_1)$ is at most*

$$\min\left(2 \cdot \boldsymbol{MaxS}(T) + \frac{4\log(2)ST}{u}, \boldsymbol{MaxS}(T) + \sqrt{\frac{\log(2)ST}{u}}\right).$$

*Proof (of Theorem 4.2 using Lemma 4.4).* Since inner queries are independent of the secret and the challenge, they can be incorporated into the preprocessing phase. Specifically, we may assume that $A_0$ makes all inner queries of $A_1$ and adds their outputs to the advice string. Then $A_1$ can simply read these outputs from the advice string instead of issuing the queries itself. This modification increases the length of the advice string to at most $S + T\log(N)$, transforms $A_1$ into a no-inner-queries adversary, and does not affect the success probability. The desired bound then follows directly from Lemma 4.4.

■

### 4.2.1 Proof overview of Lemma 4.4.

- Recall that in the distribution $P_{Z,\Sigma}$, $\Sigma$ is a uniform permutation and $Z$ is determined by $\Sigma$ as $Z = A_0(\Sigma)$. The proof essentially shows that a preprocessing adversary whose oracle is distributed as $P_{\Sigma|Z}$ cannot do much better that a non-preprocessing adversary whose oracle is distributed as $P_\Sigma$. Formally, this is done by defining a distribution $Q$, in which $\Sigma$ and $Z$ are independent, and the success probability of a non-preprocessing algorithm is bounded by $\boldsymbol{MaxS}(T)$ (Claim 4.5).

- We define $\kappa_z$ to measure the amount of information that $A_1$ obtains if the preprocessing algorithm $A_0$ outputs the value $z$, compared to a non-preprocessing adversary under $Q$ that obtains no information. We show that since $z$ is of length $S$ bits, on average over $z$, $\kappa_z \leq \log(2)S$ (Claim 4.6), confirming the intuitive insight that a string of $S$ bits can provide at most $S$ bits of information on average.

- Claim 4.7 is the heart of the proof. We fix $Z = z$, and show that the success probabilities of adversaries with and without $z$ are very close (as a function of $\kappa_z$ and $T/u$). This is done by observing that the translated outer queries (that are input to $\Sigma$) are $u$-uniform for a uniformly sampled secret $d$. It follows that the probability that $A_1$ queries any fixed $i \in [N]$ of $\Sigma$ is at most $\frac{T}{u}$, which allows us to apply variants of Shearer's lemma.

- The theorem follows by averaging over Claim 4.7 and applying Claim 4.6 to bound this average.

### 4.2.2 Proof of Lemma 4.4.

We define a distribution $Q_{\Sigma,Z,D}$, in which $\Sigma$ and $Z$ are distributed as in $P$, but are now sampled independently. Specifically, $Q_{\Sigma,Z} = P_\Sigma P_Z$. The conditioned distribution of $D$ remains unchanged, i.e., $Q_{D|\Sigma,Z} = P_{D|\Sigma,Z}$. Intuitively, the distribution $Q$ represents running the algorithms $(A_0, A_1)$, where each algorithm queries a different bijection, effectively making $A_1$ a non-preprocessing algorithm. Denoting the success probability of $(A_0, A_1)$ with respect to $Q$ by $q(A) := \mathbb{E}_{Q_{\Sigma,Z,D}}[\boldsymbol{suc}_{(A_1)_Z^{\mathcal{O}(\Sigma,D)}}(D)]$, we obtain the following claim as a direct corollary:

**Claim 4.5.**

$$q(A) \leq \boldsymbol{MaxS}(T).$$

Let

$$\kappa_z = \mathrm{D}_{\mathrm{KL}}(P_{\Sigma|Z=z} \| Q_{\Sigma|Z=z})$$

measure the amount of information $A_1$ obtains if the preprocessing algorithm $A_0$ outputs the value $z$. We have:

**Claim 4.6.**

$$\mathop{\mathbb{E}}_{P_Z(z)}[\kappa_z] \leq \log(2)S. \tag{1}$$

*Proof.* Recall that $Q_{Z,\Sigma} = P_Z P_\Sigma$. Therefore, by Property (10) above, we obtain:

$$\mathop{\mathbb{E}}_{P_Z(z)}[\kappa_z] =: \mathrm{D}_{\mathrm{KL}}(P_{\Sigma|Z} \| Q_{\Sigma|Z}) = \mathrm{D}_{\mathrm{KL}}(P_{\Sigma|Z} \| P_\Sigma) = \mathrm{D}_{\mathrm{KL}}(P_{\Sigma,Z} \| P_\Sigma P_Z) \leq \mathrm{H}(P_Z).$$

From Property (3), we get $\mathrm{H}(P_Z) \leq \log |\operatorname{Supp}(\mathcal{Z})| \leq \log(2)S$, which completes the proof. ■

We will consider the success probability of $(A_0, A_1)$ when some of the random variables are fixed and denote the fixed values in subscript. In particular, let $p_z := \mathbb{E}_{P_{\Sigma,D|Z=z}}[\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\Sigma,D)}}(D)]$ and $q_z := \mathbb{E}_{Q_{\Sigma,D|Z=z}}[\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\Sigma,D)}}(D)]$ denote the success probabilities of $(A_0, A_1)$ conditioned on $Z = z$ with respect to $P$ and $Q$, respectively. Clearly, $p = \mathbb{E}_{P_Z(z)}[p_z]$.

The heart of the proof of Lemma 4.4 is the following claim.

**Claim 4.7.** *For any $z$ in the support of $P_Z$,*

$$p_z \leq \min\left(2q_z + \frac{4\kappa_z T}{u}, q_z + \sqrt{\frac{\kappa_z T}{u}}\right).$$

The proof of Lemma 4.4 below simply averages both sides of this claim over $P_Z(z)$.

*Proof (of Claim 4.7).* We fix $z$, which fixes the queries of $A_1$.

Recall that $\mathcal{U} = \mathcal{U}(z)$ denotes the set of outer queries made by $A_1$, which includes all queries in this case. Define $|\mathcal{D}|$ sets $\{\mathcal{U}_d\}_{d\in\mathcal{D}}$, where each $\mathcal{U}_d = \{\boldsymbol{tr}(d,m) \mid m \in \mathcal{U}\}$ represents the set of indices of $\sigma$ queried by $A_1$ after translation, given that the secret is $d$ and the preprocessing string is $z$. Fix any $i \in [N]$.

Since $\boldsymbol{tr}$ is a $u$-uniform translation function, we have

$$|\{d \mid i \in \mathcal{U}_d\}| = |\cup_{m\in\mathcal{U}} \{d \mid \boldsymbol{tr}(d,m) = i\}| \leq \sum_{m\in\mathcal{U}} |\{d \mid \boldsymbol{tr}(d,m) = i\}| \leq \frac{|\mathcal{U}| \cdot |\mathcal{D}|}{u} = \frac{T \cdot |\mathcal{D}|}{u}. \tag{2}$$

For every $d \in \mathcal{D}$, define the indicator function $f_d : [N]^N \to \{0,1\}$ as $f_d(\sigma) = \boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma,d)}}(d)$, which outputs 1 if (the deterministic algorithm) $A_1$ succeeds on input $\sigma$ and secret $d$ when hard-wired with $z$.

Recall that $p_{z,d} := \mathbb{E}_{P_{\Sigma|Z=z,D=d}}[\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\Sigma,d)}}(d)]$, $q_{z,d} := \mathbb{E}_{Q_{\Sigma|Z=z,D=d}}[\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\Sigma,d)}}(d)]$ denote the success probabilities of $(A_0, A_1)$ conditioned on $Z = z$ and $D = d$, with respect to $P$ and $Q$, respectively. Note that under $Q$, generally $z \neq A_0(\sigma)$, but the success predicate $\boldsymbol{suc}$ does not depend on $z$ (see Remark 3.3) and remains accurate.

Thus,

$$\mathop{\mathbb{E}}_{P_{\Sigma|Z=z}}[f_d(\Sigma)] = p_{z,d}, \quad \mathop{\mathbb{E}}_{Q_{\Sigma|Z=z}}[f_d(\Sigma)] = q_{z,d},$$

and

$$\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} p_{z,d} = p_z, \quad \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} q_{z,d} = q_z.$$

Recall that for every $d \in \mathcal{D}$, $f_d(\sigma) = \boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma,d)}}(d)$ depends only on the secret and the part of $\sigma$ queried by $(A_1)_z^{\mathcal{O}(\sigma,d)}$. Hence, it follows from (2) that $\{f_d\}_{d\in\mathcal{D}}$ form a read-$(T \cdot |\mathcal{D}|/u)$ family on the $N$ variables of $\Sigma$.

Observe that $Q_\Sigma$ is the uniform distribution function over bijections on $[N]$, and $P_\Sigma$ is another distribution function over such bijections.

We apply Theorem 1.9 and deduce

$$(2 \cdot T \cdot |\mathcal{D}|/u) \cdot \kappa_z = (2 \cdot T \cdot |\mathcal{D}|/u) \cdot \mathrm{D}_{\mathrm{KL}}(P_{\Sigma|Z=z} \| Q_{\Sigma|Z=z}) \geq |\mathcal{D}| \cdot \mathrm{D}_{\mathrm{KL}}(p_z \| q_z).$$

Therefore, $\mathrm{D}_{\mathrm{KL}}(p_z \| q_z) \leq \frac{2 \cdot \kappa_z T}{u}$.

Applying Corollary 2.2 and Property (9), respectively, we conclude

$$p_z \leq 2q_z + \frac{4\kappa_z T}{u}, \text{ and } p_z \leq q_z + \sqrt{\frac{\kappa_z T}{u}},$$

as claimed. ■

Finally, we prove Lemma 4.4.

*Proof (of Lemma 4.4).* We average both sides of the inequality in Claim 4.7 over $P_Z(z)$.

On the left-hand-side we obtain $\mathbb{E}_{P_Z(z)}[p_z] = p$, which is the success probability of $(A_0, A_1)$ with respect to $P$.

On the right-hand-side, we consider $q_z$ and $\kappa_z$ separately. First, by Claim 4.5, we obtain

$$\mathop{\mathbb{E}}_{P_Z(z)}[q_z] = \mathop{\mathbb{E}}_{Q_Z(z)}[q_z] = q \leq \boldsymbol{MaxS}(T).$$

Second, recall that by Claim 4.6, we have $\mathbb{E}_{P_Z(z)}[\kappa_z] \leq \log(2)S$. Therefore, for the first term we obtain

$$\mathop{\mathbb{E}}_{P_Z(z)}\left[\frac{4\kappa_z T}{u}\right] \leq \frac{\log(2)4ST}{u}.$$

For the second term, we use the concavity of the square root function to conclude

$$\mathop{\mathbb{E}}_{P_Z(z)}\left[\sqrt{\frac{\kappa_z T}{u}}\right] \leq \sqrt{\frac{\mathbb{E}_{P_Z(z)}[\kappa_z]T}{u}} \leq \sqrt{\frac{\log(2)ST}{u}},$$

completing the proof. ■

### 4.3 Proof of Theorem 4.1

We now assume that $A_1$ makes $T_1 \geq 0$ inner queries and prove Theorem 4.1. The structure of the proof is similar to that of Lemma 4.4, albeit somewhat more involved.

We prove Theorem 4.1 by a hybrid argument. We refer to the hybrid as a middle game:

**Definition 4.8.** *A middle game* $\boldsymbol{MID}$ *has the same setting as* $\boldsymbol{PC}$*, and it differs only in the allowed actions for the adversary. An adversary* $\widehat{A}$ *is a pair of algorithms* $(\widehat{A}_0, \widehat{A}_1)$ *that work as follows:* $\widehat{A}_0$ *receives only* $N$ *as input, and outputs* constraints *which include a pair of sequences* $I = (I_1, \ldots, I_{T_1})$ *and* $O = (O_1, \ldots, O_{T_1})$*, each containing non-repeating elements from* $[N]$*. Then a permutation* $\sigma : [N] \to [N]$ *is sampled uniformly from the set of permutations satisfying* $\sigma(I_j) = O_j$ *for all* $j$*.* $\widehat{A}_1$ *is defined similarly to the algorithm* $A_1$ *from* $\boldsymbol{PC}$*, but with respect to that permutation* $\sigma$*, and* it is allowed to make only outer queries and no inner queries*. The outer queries of* $\widehat{A}_1$ *may depend on* $I$ *and* $O$*. The rest of the game is defined similarly. The running time* $T_2$ *of* $\widehat{A}_1$ *is defined as the number of queries it makes, and the total running time of* $\widehat{A}$ *is defined as* $T := T_1 + T_2$*.*

### 4.3.1 Proof overview.

$\boldsymbol{MID}$ can be thought of as a non-preprocessing game, where the adversary is allowed to choose a limited number of $T_1$ values of the permutation (while the other values are sampled uniformly). The proof is divided into two steps.

1. The first step shows (in Theorem 4.9 below) that the preprocessing model $\boldsymbol{PC}$ is not much stronger than $\boldsymbol{MID}$. First, allowing to fix $T_1$ values of the permutation gives the adversary in $\boldsymbol{MID}$ enough power to simulate the inner queries of the online algorithm in $\boldsymbol{PC}$, that depend on the preprocessing string. Once we have dealt with the inner queries, it remains to deal with the outer queries. We exploit the fact that in $\boldsymbol{MID}$ the non-fixed elements of the permutation are sampled uniformly from all the possible (remaining) values, hence we are left with a uniform permutation on a smaller space. Thus, the proof proceeds in similar way to the proof of Lemma 4.4 (where the online adversary only makes outer queries), considering the permutation on the smaller space. As Lemma 4.4, Theorem 4.9 is applicable to an arbitrary success predicate $\boldsymbol{suc}$, as defined in Section 3.

2. The second step shows (in Lemma 4.12 below) that fixing $T_1$ values of the permutation (independently of the secret) does not give the $\boldsymbol{MID}$ adversary much advantage over a non-preprocessing adversary. This lemma assumes that $\boldsymbol{suc}$ compares the output of $(A_1)_z$ to some function of the secret.

   The proof proceeds by transforming any adversary for the middle game into a standard non-preprocessing, non-adaptive adversary for the original permutation challenge, while losing only a small additive term in success probability. The simulator runs the $\boldsymbol{MID}$ adversary, ignores its constraints, and simply queries the real oracle on the same outer queries. A coupling argument shows that the two experiments agree except for two bad events that may cause divergence. These events correspond to collisions with the constrained input and output sets, and can be bounded by $O(T^2/u)$.

Combining the two above steps allows showing that in our setting, a preprocessing adversary is not much stronger than a non-preprocessing adversary, proving Theorem 4.1.

### 4.3.2 First step.

We first prove that the game $\boldsymbol{MID}$ is not much weaker than $\boldsymbol{PC}$.

**Theorem 4.9.** *Let $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ be a permutation challenge game, such that $\boldsymbol{tr}$ is $u$-uniform. Let $(A_0, A_1)$ be an $(S, T)$ non-adaptive algorithm with preprocessing for $\boldsymbol{PC}$. Denote by $\widehat{\boldsymbol{MaxS}}(T)$ the optimal success probability (with respect to $\boldsymbol{suc}$) of an adversary to $\boldsymbol{MID}$ with running time $T$. Then, the success probability of $(A_0, A_1)$ is at most*

$$\min\left(2 \cdot \widehat{\boldsymbol{MaxS}}(T) + \frac{4\log(2)ST}{u}, \widehat{\boldsymbol{MaxS}}(T) + \sqrt{\frac{\log(2)ST}{u}}\right).$$

In order to prove the theorem, some preparation is needed.

First, let us recall a few notations we shall use in the proof. The set of inner queries is denoted by $\mathcal{S}$, and we denote $\overline{\mathcal{S}} = [N] \setminus \mathcal{S}$. Recall that $\mathcal{S}$ depends on $z$. The distribution $P$ was defined in Section 3.2, as $P_{\Sigma,Z,D} = P_\Sigma P_{Z|\Sigma} P_D$, where $P_\Sigma$ is the uniform distribution over permutations, $P_D$ is the distribution over secrets $D$, and $P_{Z|\Sigma=\sigma}(z) = 1$ if $z = A_0(\sigma)$.

We define a distribution $Q'_{\Sigma,Z,D}$ which is identical to $P_{\Sigma,Z,D}$, with the exception that

$$Q'_{\Sigma|Z} = Q'_{\mathcal{S},\Sigma_\mathcal{S},\Sigma_{\overline{\mathcal{S}}}|Z} = Q'_{\mathcal{S},\Sigma_\mathcal{S}|Z} Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\mathcal{S},\Sigma_\mathcal{S}}$$

is defined by setting $Q'_{\mathcal{S},\Sigma_{\mathcal{S}}|Z} = P_{\mathcal{S},\Sigma_{\mathcal{S}}|Z}$, while $Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\mathcal{S},\Sigma_{\mathcal{S}}}$ is the uniform distribution over bijections mapping the elements of $\overline{\mathcal{S}}$ to those of $[N] \setminus \Sigma_{\mathcal{S}}$. On the other hand, $Q'_{\Sigma} = P_{\Sigma}$, $Q'_Z = P_Z$, $Q'_D = P_D$, and $Q'_{\Sigma,Z,D} = Q'_{\Sigma,Z} \times Q'_D$.

To simplify notation, denote the random variables that $A_1$ obtains from inner queries to $\sigma$ by $\Sigma^{in} := (\mathcal{S}, \Sigma_{\mathcal{S}})$.

Given $\Sigma^{in}$, the distribution function $Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\Sigma^{in}}$ does not depend on the value of $Z$. Therefore,

$$Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\Sigma^{in}} = Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}}.$$

Note that if $T_1 = 0$ (i.e., if $\mathcal{S} = \emptyset$), then $Q'$ is identical to $Q$ defined above. However, in general we have $Q'_{\Sigma|Z} = Q'_{\Sigma^{in}|Z} Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\Sigma^{in}} = P_{\Sigma^{in}|Z} Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}}$, while $Q_{\Sigma|Z} = Q_{\Sigma} = P_{\Sigma}$.

Denote the success probability of $(A_0, A_1)$ with respect to $Q'$ by

$$q' := \mathop{\mathbb{E}}_{Q'_{\Sigma,Z,D}} [\boldsymbol{suc}_{(A_1)_Z^{\mathcal{O}(\Sigma,D)}}(D)].$$

**Claim 4.10.** *We have*

$$q' \leq \widehat{\boldsymbol{MaxS}}(T).$$

*Proof.* Recall that $Q'_{\Sigma_{\overline{\mathcal{S}}}|Z,\Sigma^{in}} = Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}}$ is the uniform bijection mapping the elements of $\overline{\mathcal{S}}$ to those of $[N] \setminus \Sigma_{\mathcal{S}}$.

Since $(A_1)_z$ queries $\mathcal{S}$, the preprocessing string $z$ does not give $A_1$ any additional information about $\Sigma$. Thus, $A$ under $Q'$ cannot do better than in the following game: First $A_0$ receives a uniform permutation $\sigma'$, and chooses a set $\mathcal{S}$ of elements to fix to pre-chosen values. Then, these elements are fixed, and the other elements of $\sigma'$ are re-shuffled to obtain $\sigma$. The rest of the game occurs with respect to $\sigma$. One can see that this game is equivalent to $\boldsymbol{MID}$.

Formally, given a preprocessing algorithm $(A_0, A_1)$ we define a (randomized) adversary $(\widehat{A}_0, \widehat{A}_1)$ for $\boldsymbol{MID}$ that makes $T$ queries and has success probability identical to $(A_0, A_1)$ (i.e., $q'$). By definition, the success probability of $(\widehat{A}_0, \widehat{A}_1)$ is bounded by $\widehat{\boldsymbol{MaxS}}(T)$, hence this will complete the proof. It remains to define $(\widehat{A}_0, \widehat{A}_1)$.

$\widehat{A}_0$ receives $N$ as input, samples a permutation $\sigma'$, and calls $A_0$ to obtain $z$. Then, it calls $A_1$ with $z$ as input and obtains the set $\mathcal{S}(z)$ of inner queries. Finally, $\widehat{A}_0$ outputs $I = \mathcal{S}(z)$ and $O = \sigma'_{\mathcal{S}(z)}$. $\widehat{A}_1$ receives oracle access to a permutation $\sigma$ with $\sigma_{\mathcal{S}} = \sigma'_{\mathcal{S}}$ and independent values elsewhere. Notably, $\mathcal{S}, \sigma_{\mathcal{S}}$ and $z$ are known to $\widehat{A}_1$, since $\widehat{A}_0$ and $\widehat{A}_1$ share the randomness tape and it is the only randomness source of $\widehat{A}_0$. Therefore, the distribution of $\sigma$ conditioned on $z$ is exactly $Q'_{\Sigma|Z}$. Therefore, $\widehat{A}_1$ can perfectly simulate $A_1$ by making the same (non-adaptive) outer queries as $A_1$ and returning its output. Thus, the success probability of $\widehat{A}$ is $q'$, as required. ■

Fixing $z, \sigma^{in}$, we can think of $\sigma_{\overline{\mathcal{S}}}$ drawn from either $P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z}$ or $Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}$ as an extended bijection from $[N]$ to itself that maps the elements of $\mathcal{S}$ to those in $\sigma_{\mathcal{S}}$, and maps the elements of $\overline{\mathcal{S}}$ to those of $[N] \setminus \sigma_{\overline{\mathcal{S}}}$.

We can therefore think of $\sigma^{in}$ as being hardwired into $(A_1)_z$ such that it only makes queries to the random variables $\Sigma_{\mathcal{S}}$ (after translation). Of course, some of the (translated) queries of $(A_1)_z$ may still fall in $\mathcal{S}$, but $\sigma^{in} = (\mathcal{S}, \sigma_{\mathcal{S}})$ are formally no longer random variables.

Denote

$$p_{z,\sigma^{in}} := \mathop{\mathbb{E}}_{P_{\Sigma_{\overline{\mathcal{S}}},D|\Sigma^{in}=\sigma^{in},Z=z}} [\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma^{in},\Sigma_{\overline{\mathcal{S}}},D)}}(D)]$$

and

$$q'_{z,\sigma^{in}} := \mathop{\mathbb{E}}_{Q'_{\Sigma_{\overline{\mathcal{S}}},D|\Sigma^{in}=\sigma^{in},Z=z}} [\boldsymbol{suc}_{(A_1)_z^{\mathcal{O}(\sigma^{in},\Sigma_{\overline{\mathcal{S}}},D)}}(D)].$$

Also, denote

$$\kappa'_{z,\sigma^{in}} = \mathrm{D}_{\mathrm{KL}}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z} \| Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}).$$

The heart of the proof of Theorem 4.9 is the following additional claim.

**Claim 4.11.** *For any $z, \sigma^{in}$ in the support of $P_{Z,\Sigma^{in}}$,*

$$p_{z,\sigma^{in}} \leq \min\left(2q'_{z,\sigma^{in}} + \frac{4\kappa'_{z,\sigma^{in}}T_2}{u}, q'_{z,\sigma^{in}} + \sqrt{\frac{\kappa'_{z,\sigma^{in}}T_2}{u}}\right).$$

*Proof.* We fix $z, \sigma^{in}$ (hence fixing the queries of $A_1$).

As in the proof of Claim 4.7, define $|\mathcal{D}|$ sets $\{\mathcal{U}_d\}_{d\in\mathcal{D}}$, where $\mathcal{U}_d = \{\boldsymbol{tr}(d,m) \mid m \in \mathcal{U}\}$. Recall that $\Sigma^{in}$ and $D$ are independent random variables (under both $P$ and $Q'$), so for any value $\Sigma^{in} = \sigma^{in}$, $D$ is still uniform.

For every $d \in \mathcal{D}$, define the indicator function $f_c : [N]^{N-T_1} \mapsto \{0,1\}$ by setting $f_d(\sigma_{\overline{\mathcal{S}}}) = \boldsymbol{suc}_{(A_1)_z}^{\mathcal{O}(\sigma^{in},\sigma_{\overline{\mathcal{S}}},d)}(d)$, which outputs 1 if (the deterministic algorithm) $A_1$ succeeds with the secret $d$ on input $\sigma_{\overline{\mathcal{S}}}$ when hard-wired with $z, \sigma^{in}$.

Recall that

$$p_{z,\sigma^{in},d} := \mathop{\mathbb{E}}_{P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z,D=d}}[\boldsymbol{suc}_{(A_1)_z}^{\mathcal{O}(\sigma^{in},\Sigma_{\overline{\mathcal{S}}},d)}(d)]$$

and

$$q'_{z,\sigma^{in},d} := \mathop{\mathbb{E}}_{Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z,D=d}}[\boldsymbol{suc}_{(A_1)_z}^{\mathcal{O}(\sigma^{in},\Sigma_{\overline{\mathcal{S}}},d)}(d)]$$

denote the success probability of $(A_0, A_1)$ conditioned on $Z = z, \Sigma^{in} = \sigma^{in}, D = d$ with respect to $P$ and $Q'$, respectively.

Thus,

$$\mathop{\mathbb{E}}_{P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z}}[f_d(\Sigma_{\overline{\mathcal{S}}})] = p_{z,\sigma^{in},d} \qquad \text{and} \qquad \mathop{\mathbb{E}}_{Q_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z}}[f_d(\Sigma_{\overline{\mathcal{S}}})] = q'_{z,\sigma^{in},d}.$$

Also,

$$\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} p_{z,\sigma^{in},d} = p_{z,\sigma^{in}} \qquad \text{and} \qquad \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} q'_{z,\sigma^{in},d} = q'_{z,\sigma^{in}}.$$

Recall that when $\sigma^{in}$ is fixed, $A_1$ only makes queries in $\overline{\mathcal{S}}$ (after translation), and by similar calculation to (2), $\{f_d\}_{d\in\mathcal{D}}$ form a read-$(T_2 \cdot |\mathcal{D}|/u)$ family on the $N - T_1$ variables of $\Sigma_{\overline{\mathcal{S}}}$.

Note that $\sigma_{\overline{\mathcal{S}}}$ drawn from either $P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma_{\mathcal{S}}=\sigma_{\mathcal{S}},Z=z}$ or $Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z} = Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}$ is a bijection from $\overline{\mathcal{S}}$ to $[N] \setminus \sigma_{\mathcal{S}}$, and $Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}$ is uniformly chosen. We apply Theorem 1.9 and deduce

$$\begin{aligned}(2 \cdot T_2 \cdot |\mathcal{D}|/u)\kappa'_{z,\sigma^{in}} &= (2 \cdot T_2 \cdot |\mathcal{D}|/u)\,\mathrm{D}_{\mathrm{KL}}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z} \| Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}) \\ &\geq |\mathcal{D}|\,\mathrm{D}_{\mathrm{KL}}(p_{z,\sigma^{in}} \| q'_{z,\sigma^{in}}).\end{aligned}$$

Therefore,

$$\mathrm{D}_{\mathrm{KL}}(p_{z,\sigma^{in}} \| q'_{z,\sigma^{in}}) \leq \frac{2 \cdot \kappa'_{z,\sigma^{in}}T_2}{u}.$$

Applying Corollary 2.2 and Property (9), respectively, we conclude

$$p_{z,\sigma^{in}} \leq 2q'_{z,\sigma^{in}} + \frac{4\kappa'_{z,\sigma^{in}}T_2}{u} \qquad \text{and} \qquad p_{z,\sigma^{in}} \leq q'_{z,\sigma^{in}} + \sqrt{\frac{\kappa'_{z,\sigma^{in}}T_2}{u}},$$

as claimed. ∎

Now, we are ready to prove Theorem 4.9.

*Proof (of Theorem 4.9).* We average both sides of the inequality of Claim 4.11 over $P_{Z,\Sigma^{in}}(z, \sigma^{in}) = Q'_{Z,\Sigma^{in}}(z, \sigma^{in})$.

On the left-hand-side, we obtain $\mathbb{E}_{P_{Z,\Sigma^{in}}(z,\sigma^{in})}[p_{z,\sigma^{in}}] = p$.

On the right-hand-side, we consider $q'_{z,\sigma^{in}}$ and $\kappa'_{z,\sigma^{in}}$ separately.

First, since $P_{Z,\Sigma^{in}}(z,\sigma^{in}) = Q'_{Z,\Sigma^{in}}(z,\sigma^{in})$, we have

$$\mathop{\mathbb{E}}_{Q'_{Z,\Sigma^{in}}(z,\sigma^{in})}[q'_{z,\sigma^{in}}] = q' \leq \widehat{\boldsymbol{MaxS}}(T),$$

where the inequality is by Claim 4.10.

Second, we have

$$\mathop{\mathbb{E}}_{Q_{Z,\Sigma^{in}}(z,\sigma^{in})}[\kappa'_{z,\sigma^{in}}] = \mathop{\mathbb{E}}_{Q_{Z,\Sigma^{in}}(z,\sigma^{in})}[\mathrm{D}_{\mathrm{KL}}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z} \| Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}})].$$

Recall that for any $\sigma^{in}$ , $Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}$ is uniform over its support, hence, by Property (6)

$$\begin{aligned}
&\mathop{\mathbb{E}}_{Q_{Z,\Sigma^{in}}(z,\sigma^{in})}[\mathrm{D}_{\mathrm{KL}}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z} \| Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}})] \\
&= \mathop{\mathbb{E}}_{Q_{Z,\Sigma^{in}}(z,\sigma^{in})}[\mathrm{H}(Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in}}) - \mathrm{H}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}=\sigma^{in},Z=z})] \\
&= \mathrm{H}(Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in}}) - \mathrm{H}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma^{in},Z}) = \mathrm{H}(Q'_{\Sigma_{\overline{\mathcal{S}}}|\mathcal{S},\Sigma_{\mathcal{S}},Z}) - \mathrm{H}(P_{\Sigma_{\overline{\mathcal{S}}}|\mathcal{S},\Sigma_{\mathcal{S}},Z}) \\
&= \mathrm{H}(Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma_{\mathcal{S}},Z}) - \mathrm{H}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma_{\mathcal{S}},Z}).
\end{aligned}$$

By Property (2) and Property (1),

$$\begin{aligned}
\mathrm{H}(Q'_{\Sigma_{\overline{\mathcal{S}}}|\Sigma_{\mathcal{S}},Z}) &= \mathrm{H}(Q'_{\Sigma|Z}) - \mathrm{H}(Q'_{\Sigma_{\mathcal{S}}|Z}) = \mathrm{H}(Q'_{\Sigma|Z}) - \mathrm{H}(P_{\Sigma_{\mathcal{S}}|Z}) \leq \mathrm{H}(Q'_{\Sigma}) - \mathrm{H}(P_{\Sigma_{\mathcal{S}}|Z}) \\
&\leq \mathrm{H}(P_{\Sigma}) - \mathrm{H}(P_{\Sigma_{\mathcal{S}}|Z}),
\end{aligned}$$

and similarly,

$$\mathrm{H}(P_{\Sigma_{\overline{\mathcal{S}}}|\Sigma_{\mathcal{S}},Z}) = \mathrm{H}(P_{\Sigma|Z}) - \mathrm{H}(P_{\Sigma_{\mathcal{S}}|Z}).$$

Therefore, by Property (10), Property (3), and since $(A_0, A_1)$ is an $(S,T)$ algorithm,

$$\mathop{\mathbb{E}}_{Q_{Z,\Sigma^{in}}(z,\sigma^{in})}[\kappa'_{z,\sigma^{in}}] \leq \mathrm{H}(P_{\Sigma}) - \mathrm{H}(P_{\Sigma|Z}) = \mathrm{I}(\Sigma; Z) \leq \mathrm{H}(Z) \leq \log|\operatorname{Supp}(\mathcal{Z})| \leq \log(2)S.$$

Overall, we obtain

$$p \leq 2 \cdot \widehat{\boldsymbol{MaxS}}(T) + \frac{\log(2)4ST}{u}, \text{ and } p \leq \widehat{\boldsymbol{MaxS}}(T) + \sqrt{\frac{\log(2)ST}{u}},$$

where the second inequality uses the concavity of the square root function. This concludes the proof. ■

### 4.3.3 Second step - completing the proof of Theorem 4.1.

Theorem 4.1 follows immediately from Theorem 4.9 and the following lemma (which is the only lemma that restricts $\boldsymbol{suc}$):

**Lemma 4.12.** *Let $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ be a permutation challenge game, such that $\boldsymbol{tr}$ is $u$-uniform, and $\boldsymbol{suc}$ compares the output of $A_1$ to some function of the secret. Denote by $\boldsymbol{MaxS}(T)$ the optimal success probability (with respect to $\boldsymbol{suc}$) of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries.*

*Let $\boldsymbol{MID}$ be the corresponding game defined above (see Definition 4.8). Denote by $\widehat{\boldsymbol{MaxS}}(T)$ the optimal success probability (with respect to $\boldsymbol{suc}$) of an adversary to $\boldsymbol{MID}$ with running time $T$. Then*

$$\widehat{\boldsymbol{MaxS}}(T) \leq \boldsymbol{MaxS}(T) + \frac{T^2}{2u}.$$

*Proof.* Let $\widehat{A} = (\widehat{A}_0, \widehat{A}_1)$ be an adversary to $\boldsymbol{MID}$ with running time $T = T_1 + T_2$, and denote its success probability by $\widehat{q}$. We describe a non-preprocessing algorithm $A$ for $\boldsymbol{PC}$ with success probability $q \geq \widehat{q} - \frac{T^2}{2u}$ that makes $T$ queries.

$A$ simulates $\widehat{A}_0$ to obtain the constraints $I, O$, which also determine the outer queries of $\widehat{A}_1$ (recall that a $\boldsymbol{MID}$ game has no inner queries). It ignores the constraints of $\widehat{A}$, requests its outer queries from $\mathcal{O}$, sends the responses to $\widehat{A}_1$ and outputs the same value.

Let $g : \mathcal{D} \to \mathcal{D}'$ be the function of the secret computed by $\boldsymbol{suc}$ ($\mathcal{D}'$ is its output space). Denote by $\mathrm{P} : \mathcal{D}' \times \mathcal{D}' \to \{0, 1\}$ the success predicate of $\boldsymbol{suc}$.

Below, we define the algorithm $\boldsymbol{suc}_{A^{\mathcal{O}(\sigma,d)}}(d)$ in the above case, where $\widehat{A}$ is simulated by $A$ (we omit the explicit interaction with $A$ for simplicity).

1. Receive $I, O$ from $\widehat{A}_0$.
2. Pass $I, O$ to $\widehat{A}_1$ and receive the outer queries $\mathcal{U} = (m_1, \ldots, m_{T_2})$.
3. Sample $d \sim \mathcal{D}$.
4. For all $1 \leq i \leq T_2$:
   (a) Denote $u_i \leftarrow \boldsymbol{tr}(d, m_i)$.
   (b) If $u_i = u_j$ for some $j < i$, set $v_i \leftarrow v_j$. Otherwise, sample $v_i \sim [N] \setminus \{v_1, \ldots, v_{i-1}\}$ uniformly.
5. Pass $(\boldsymbol{post}(d, v_1), \ldots, \boldsymbol{post}(d, v_{T_2}))$ to $\widehat{A}_1$ and receive the answer $\tilde{d} \in D'$.
6. Return $\mathrm{P}(g(d), \tilde{d})$.

Below, we define the algorithm $\boldsymbol{suc}_{\widehat{A}_1^{\mathcal{O}(\sigma,d)}}(d)$, in case that $\widehat{A}$ runs in the $\boldsymbol{MID}$ game. It uses two additional flags, $W_1, W_2$, that do not affect the output, and are only defined for the sake of the analysis. The differences from the previous algorithm are underlined.

1. Receive $I, O$ from $\widehat{A}_0$.
2. Pass $I, O$ to $\widehat{A}_1$ and receive the outer queries $\mathcal{U} = (m_1, \ldots, m_{T_2})$.
3. Sample $d \sim \mathcal{D}$.
4. <u>Set $W_1 \leftarrow 0, W_2 \leftarrow 0$.</u>
5. For all $1 \leq i \leq T_2$:
   (a) Denote $u_i \leftarrow \boldsymbol{tr}(d, m_i)$.
   (b) <u>If $u_i = I_k \in I$, set $v_i \leftarrow O_k$ and $W_1 \leftarrow 1$. Continue to $m_{i+1}$.</u>
   (c) If $u_i = u_j$ for some $j < i$, set $v_i \leftarrow v_j$. Otherwise, sample $v_i \sim [N] \setminus \{v_1, \ldots, v_{i-1}\}$ uniformly.
   (d) <u>If $v_i \in O$, re-sample $v_i \sim [N] \setminus (O \cup \{v_1, \ldots, v_{i-1}\})$ uniformly, and set $W_2 \leftarrow 1$.</u>
6. Pass $(\boldsymbol{post}(d, v_1), \ldots, \boldsymbol{post}(d, v_{T_2}))$ to $\widehat{A}_1$ and receive the answer $\tilde{d} \in D'$.
7. Return $\mathrm{P}(g(d), \tilde{d})$.

Observe that the algorithms are equivalent, as long as $W_1 = W_2 = 0$ at the end of the execution of $\boldsymbol{suc}_{\widehat{A}_1^{\mathcal{O}(\sigma,d)}}(d)$). Thus,

$$\begin{aligned} q &= \Pr[\boldsymbol{suc}_{A^{\mathcal{O}(\Sigma,D)}}(D)] \\ &\geq \Pr[\boldsymbol{suc}_{\widehat{A}_1^{\mathcal{O}(\Sigma,D)}}(D)) \cap (W_1 = W_2 = 0)] \\ &\geq \Pr[\boldsymbol{suc}_{\widehat{A}_1^{\mathcal{O}(\Sigma,D)}}(D))] - \Pr[W_1 = 1] - \Pr[W_2 = 1] \\ &= \widehat{q} - \Pr[W_1 = 1] - \Pr[W_2 = 1]. \end{aligned}$$

By a union bound over the $T_2$ outer queries, we have $\Pr[W_1 = 1] \leq \frac{T_1 T_2}{u} \leq \frac{T^2}{4u}$, where the probability is taken over the choice of $d$. By another union bound over the $T_2$ outer queries, $\Pr[W_2 = 1] \leq \frac{T^2}{4u}$, where the probability is taken over the choice of $c_1, c_2, \ldots$. Overall, $q \geq \widehat{q} - \frac{T^2}{2u}$, as claimed. ■

# 5 Applications to Problems in the Generic Group Model

In this section we describe our applications to the DLOG, DDH, and sqDDH problems. All these applications (as well as the application to the Even-Mansour cryptosystem presented in Section 6) are obtained via Theorem 4.1. This theorem can indeed be used, since in all these applications, $\boldsymbol{suc}$ compares the output of $(A_1)_z$ to some function of the secret (as defined in Section 3.3). We stress again that none of the uses of this theorem directly analyzes the preprocessing setting. For simplicity, we count the input group elements of the generic group algorithm $A_1$ as part of its $T$ queries. Otherwise, we have to add to $T$ a small additive factor based on its input size.

## 5.1 The discrete-log problem

As was written in the introduction, the bound we prove for the DLOG problem is the following.

**Theorem 1.1.** Let $A = (A_0, A_1)$ be a non-adaptive $(S, T)$-algorithm for the DLOG problem in the generic group model, over a group $G$ with a prime number $N$ of elements. Denote by $\boldsymbol{MaxS}_{\mathrm{DLOG}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most

$$2 \cdot \boldsymbol{MaxS}_{\mathrm{DLOG}}(T) + \frac{4\log(2)ST}{N} + \frac{T^2}{N} \leq \frac{3T^2}{N} + \frac{4\log(2)ST}{N}.$$

*Proof (of Theorem 1.1).* We would like to use Theorem 4.1 via the corresponding permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{DL}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ defined in Section 3.3. For this purpose it remains to show that $\boldsymbol{tr}$ is $N$-uniform, where $\boldsymbol{tr}(d, (a, b)) = a \cdot d + b \mod N$. Since for every query the translation function defines a polynomial of degree 1 in $d$, Lemma 3.4 implies that $\boldsymbol{tr}$ is $N$-uniform.

Finally, as was mentioned in the introduction, the bound on online$(A_1)$ on the right-hand-side is by Shoup's theorem [Sho97] which bounds $\boldsymbol{MaxS}_{\mathrm{DLOG}}(T) \leq \frac{T^2}{N}$ for DLOG. ■

## 5.2 The DDH and sqDDH problems

As was written in the introduction, the main theorem we prove for DDH and sqDDH is the following.

**Theorem 1.2.** Let $A = (A_0, A_1)$ be a non-adaptive $(S, T)$ algorithm for the DDH (resp. sqDDH) problem in the GGM, over a group $G$ with a prime number $N$ of elements. Denote by

$\boldsymbol{MaxS}_{\mathrm{DDH}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most

$$\boldsymbol{MaxS}_{\mathrm{DDH}}(T) + \sqrt{\frac{2\log(2)ST}{N}} + \frac{T^2}{N} \leq \frac{1}{2} + \frac{2T^2}{N} + \sqrt{\frac{2\log(2)ST}{N}}.$$

*Proof (of Theorem 1.2).* We treat DDH and sqDDH separately.

**DDH.** We would like to use Theorem 4.1 via the corresponding permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{DDH}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ defined in Section 3.3. For this purpose, it remains to show that $\boldsymbol{tr}$ is $(N/2)$-uniform.

Recall that

$$\boldsymbol{tr}((d_1, d_2, d_3, k), (a_1, a_2, a_3, b)) = \begin{cases} a_1 \cdot d_1 + a_2 \cdot d_2 + a_3 \cdot d_3 + b \mod N & \text{if } k = 0, \\ a_1 \cdot d_1 + a_2 \cdot d_2 + a_3 \cdot (d_1 d_2) + b \mod N & \text{if } k = 1. \end{cases}$$

Since for every query, the translation function defines a polynomial of degree 1 or 2 in $c_1, c_2, c_3$, Lemma 3.4 implies that $\boldsymbol{tr}$ is $(N/2)$-uniform, as asserted.

**sqDDH.** We apply Theorem 4.1 via the corresponding permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{sqDDH}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ defined in Section 3.3. For this purpose, it remains to show that $\boldsymbol{tr}$ is $(N/2)$-uniform.

Recall that

$$\boldsymbol{tr}((d_1, d_2, k), (a_1, a_2, b)) = \begin{cases} a_1 \cdot d_1 + a_2 \cdot d_2 + b \mod N & \text{if } k = 0, \\ a_1 \cdot d_1 + a_2 \cdot (d_1)^2 + b \mod N & \text{if } k = 1. \end{cases}$$

Since for every query, the translation function defines a polynomial of degree 1 or 2 in $d_1, d_2$, Lemma 3.4 implies that $\boldsymbol{tr}$ is $(N/2)$-uniform, as asserted.

Finally, the inequality in the assertion holds by Shoup's theorem [Sho97] which bounds $\boldsymbol{MaxS}_{\mathrm{DDH}}(T) \leq \frac{1}{2} + \frac{T^2}{N}$ for DDH, and a similar argument which yields the same bound for sqDDH (see [CDG18]). ■

# 6 Application to the Even-Mansour Cryptosystem

In this section we present our application to the EM cryptosystem. We first briefly review previous results on EM, and then we present the proof of Theorem 1.3.

## 6.1 Previous results on the security of EM

Recall that the Even-Mansour cryptosystem [EM97] is defined as $EM(m) = k_2 \oplus \sigma(k_1 \oplus m)$, where $k_1, k_2 \in \{0,1\}^n$ are $n$-bit keys and $\sigma : \{0,1\}^n \to \{0,1\}^n$ is a publicly known permutation. Even and Mansour showed that if $\sigma$ is chosen uniformly at random, then any attack which makes $T_2$ encryption/decryption queries to $EM$ and $T_1$ queries to $\sigma$ or $\sigma^{-1}$, has a success probability of $O(T_1 T_2 / 2^n)$.

Right after the introduction of the EM cryptosystem, Daemen [Dae91] presented a matching attack: Pick an arbitrary $\alpha \in \{0,1\}^n$, query $\sigma$ to obtain $T_1/2$ pairs of values $(\sigma(x_i), \sigma(x_i \oplus \alpha))$, and store the pairs $(\sigma(x_i) \oplus \sigma(x_i \oplus \alpha), x_i)$ in a sorted table. Then, query $EM$ to obtain $T_2/2 = N/T_1$ pairs of values $(EM(m_i), EM(m_i \oplus \alpha))$, and check, for each $j$, whether $EM(m_j) \oplus EM(m_j \oplus \alpha)$ appears in the table. If a collision of the form $EM(m_j) \oplus EM(m_j \oplus \alpha) = \sigma(x_i) \oplus \sigma(x_i \oplus \alpha)$ is found, then it is likely that $k_1 = m_j \oplus x_i$ or $k_1 = m_j \oplus x_i \oplus \alpha$. At this stage, the second key $k_2$ can be retrieved easily. The attack algorithm is based on the fact that

XOR with a secret key preserves XOR differences, and on the birthday paradox. The memory complexity of Daemen's attack is $\tilde{O}(T_1)$.

In 2000, Biryukov and Wagner [BW00] presented an alternative attack with complexities of $T_1 = T_2 = O(2^{n/2})$, and in 2012, Dunkelman, Keller and Shamir [DKS15] showed that its memory complexity can be reduced to $\tilde{O}(1)$, using an adaptive collision-search procedure. The authors of [DKS15] also generalized the attack of [BW00] to the entire tradeoff curve $T_1T_2 = O(2^n)$ and asked whether its memory complexity can be reduced to $\tilde{O}(1)$ for the entire curve. This was partially addressed by Fouque, Joux and Mavromati [FJM14], who showed that the memory complexity can be reduced to $o(T_1)$ using an adaptive procedure, though a reduction to $\tilde{O}(1)$ has not been found yet.

While Daemen's attack is clearly non-adaptive, the Biryukov-Wagner attack and its enhancements heavily use adaptivity via Floyd's algorithm. Like in the case of DLOG, for about 35 years it hasn't been known whether adaptivity is indeed essential for reducing the memory complexity below the $\tilde{O}(T_1)$ complexity of Daemen's algorithm.

Daemen's algorithm can be naturally viewed as an algorithm with preprocessing, as the public knowledge of $\sigma$ allows performing all queries to it offline. The Biryukov-Wagner algorithm and its variants do not use preprocessing. Fouque, Joux and Mavromati [FJM14] showed that if preprocessing with a space of $S = 2^{n/3}$ is allowed, then an attack with online complexity of $T = \tilde{O}(2^{n/3})$ can be mounted. In the other direction, Coretti, Dodis and Guo [CDG18] showed that any distinguishing attack on EM with preprocessing has a success probability of $\frac{1}{2} + \tilde{O}(\sqrt{S(T_1+T_2)T_2/2^n} + T_1T_2/2^n)$, matching the attack of [FJM14] in the case of constant success probability.

## 6.2 Our bound for EM

In this section we prove Theorem 1.3 stated in the introduction. We assume that adversary can query only the public and encryption oracles and not the decryption oracle. A direct corollary due to the symmetry of EM is that the same results hold if the adversary can query only the public and decryption oracles.

**Theorem 1.3.** Let $A = (A_0, A_1)$ be a key-recovery, non-adaptive $(S, T)$-adversary for the Even-Mansour cryptosystem, which can query only the public and encryption oracles and not the decryption oracle. Denote by $\boldsymbol{MaxS}_{\mathrm{EM}}(T)$ the optimal success probability of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $A$ is at most

$$2 \cdot \boldsymbol{MaxS}_{\mathrm{EM}}(T) + \frac{4\log(2)S(T+1)}{N} + \frac{T^2}{N} \leq \frac{3T^2}{N} + \frac{4\log(2)S(T+1)}{N}.$$

Moreover, the theorem also holds for the single-key variant where $k_1 = k_2$.

*Proof (of Theorem 1.3).* We would like to use Theorem 4.1 via the corresponding permutation challenge game $\boldsymbol{PC} := \boldsymbol{PC}_{EM-KR}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ defined in Section 3.4. First, observe that $\boldsymbol{suc}$ compares the output of $(A_1)_z$ to the secret key, which is a function of the secret. It remains to show that $\boldsymbol{tr}$ is $N$-uniform, where $\boldsymbol{tr}((k_1, k_2), m) = m \oplus k_1$.

Fix $m \in [N]$ and let $j \in [N]$. Then,

$$\boldsymbol{tr}((k_1, k_2)), m) = j \Leftrightarrow m \oplus k_1 = j \Leftrightarrow k_1 = j \oplus m.$$

Therefore,

$$\mathop{\mathbb{E}}_{P_{K_1,K_2}}[\mathbb{1}(\boldsymbol{tr}((K_1, K_2), m) = j)] = \mathop{\mathbb{E}}_{P_{K_1,K_2}}[\mathbb{1}(K_1 = j \oplus m)] = \tfrac{1}{N},$$

as required. Note that this also holds for the single-key scheme with $k_1 = k_2$. ■

# A Limitations of Previous Techniques in Dealing with Non-Adaptive Algorithms

We consider the three main generic techniques for proving cryptanalytic time-space tradeoffs: compression arguments, the pre-sampling technique, and concentration inequalities. The pre-sampling technique and the concentration inequalities technique are closely related, as demonstrated in [GLLZ21]. Our techniques are related to both, as they are also based on concentration inequalities. Yet, the previously used concentration inequalities are somewhat similar to martingales, which are generally not strong enough to obtain "dimension-free" inequalities such as those obtained from Shearer's inequality (see [Von10, Section 4]). Furthermore, it seems that compression arguments are not sufficiently strong to obtain our results as well (though, we cannot give a short intuitive argument for this).

We now argue in more detail that previous techniques are inherently limited: even when applied to non-adaptive algorithms, they can improve over the tight bounds known for adaptive algorithms by at most a polylogarithmic factor in $N$. Thus, they cannot meaningfully distinguish between adaptive and non-adaptive algorithms in our setting. For simplicity, we focus on the DLOG problem. Since this argument is obvious for the pre-sampling technique, we focus on techniques based on concentration inequalities and compression arguments.

We remark that we cannot completely rule out the possibility that a simple extension of these techniques would give a significant improvement, but this seems highly unlikely.

## A.1 Concentration inequality-based proofs in the multi-instance model

### The MI model.

We summarize the main ideas of the technique that is based on concentration inequalities. It reduces proving security against adversaries with advice to analyzing multi-instance (MI) security against adversaries without advice. In the (basic) MI DLOG game, an adversary plays a sequence of DLOG instances. Initially, a uniform permutation is chosen, and it remains fixed throughout the MI DLOG game. In each instance, a new uniform DLOG secret is independently chosen and the adversary is allowed to issue $T$ new queries in order to solve it. The adversary wins the multi-instance game if all instances are solved correctly. The main theorem asserts that if the success probability (of any adversary) in the MI DLOG game with $S$ instances is at most $\delta^S$ for some $0 < \delta < 1$, then the success probability of any adversary with $S$ bits of advice is at most $O(\delta)$.

For simplicity, we focus on the setting of a constant $\delta$. Below we describe a non-adaptive adversary to the MI game with success probability of at least $2^{-\tilde{O}(N/T^2)}$ regardless of the number of instances. Asymptotically, this is also the success probability of the best general (adaptive) adversary, hence this technique cannot distinguish between adaptive and non-adaptive adversaries.

Considering time-space tradeoffs in the proprocessing model, our adversary shows that this technique cannot prove a bound that is better than $ST^2 \geq \tilde{\Omega}(N)$ (which is tight for adaptive algorithms [CK18]), while Theorem 1.1 proves a much better bound.

### The MI adversary.

Assume that $T^2 = o(N)$, as otherwise, a non-preprocessing algorithm already achieves a constant advantage. In addition, the MI adversary can guess each secret DLOG with probability $N^{-1}$, hence it can achieve a success probability of at least $2^{-\tilde{O}(N/T^2)}$ with $\tilde{O}(N/T^2)$ instances. We may thus assume that the number of instances $S$ is at least $\tilde{\Omega}(N/T^2)$.

In the MI game, we have a sequence of DLOG instances, with corresponding secrets $d_1, d_2, \ldots$. Consider an MI adversary $B$ that for each instance, executes the same sequence of queries

$(a_j, 0)_{j\in[T]}$. We select a generator $g \in \mathbb{Z}_N^*$ and choose $a_i = g^{-i} \bmod N$ for $1 \leq i \leq T/2$ and $a_i = g^{(i-T/2)\cdot T} \bmod N$ for $T/2 < i \leq T$ (somewhat mimicking the baby-step-giant-step algorithm). These queries are translated to oracle queries of the form $a_j d_i \bmod N$.

Observe that if a collision in the translated queries of the form $a_j d_i \bmod N = a_{j'} d_{i'} \bmod N$ occurs for $d_i \neq d_{i'}$, then $d_i = a_{j'} \cdot (a_j)^{-1} \cdot d_{i'} \bmod N$ (recall that $a_j \neq 0$ by assumption). Hence, $d_{i'}$ determines $d_i$ and vice versa. Such collisions can be detected by $B$, as they lead to collisions at the output of $\sigma$.

Thus, for each instance, if its (translated) queries do not collide with the (translated) queries of a previous instance, $B$ guesses its secret. Otherwise, using the collision, $B$ computes the secret of the instance deterministically using the previous guess.

**Analysis sketch.**

Let us show that for $S \geq \tilde{\Omega}(N/T^2)$, the success probability of $B$ is at least $2^{-\tilde{O}(N/T^2)}$. $B$ guesses $d_i$ correctly for all $1 \leq i \leq \tilde{O}(N/T^2)$ with probability $2^{-\tilde{O}(N/T^2)}$. In this case, $B$ succeeds if each of the remaining secrets can be determined from them. Hence, it remains to show that each of the remaining secrets can be determined with high probability.

Notably, after $\tilde{\Omega}(N/T^2)$ instances — corresponding to a total of $\tilde{\Omega}(N/T)$ queries — the following property holds with very high probability due to our choice of non-adaptive queries: every interval of the form $g^i, g^{i+1}, \ldots, g^j$, where $i, j \in [N-1]$ and $j-i \bmod N-1 = T/2$, contains at least one queried point. The proof is by a Chernoff bound for each interval (exploiting the fact that each interval is hit by a query of an instance with probability $\Omega(T^2/N)$), followed by a union bound over all such intervals. Once this property holds, the queries of every new instance are guaranteed to collide with the queries of a (successfully guessed) previous one, allowing the recovery of its secret.

## A.2 Compression arguments

Compression arguments for proving time-space lower bounds for the DLOG problem have the following structure: one starts with a DLOG adversary with advice $A = (A_0, A_1)$ that possesses a (too good to be true) time-space tradeoff. This adversary is then used to devise encoding and decoding procedures for the random permutation oracle. These procedures compress the oracle beyond what is possible information-theoretically, leading to a contradiction.

Specifically, on input $\sigma$, the encoding procedure first runs $A_0(\sigma)$ and writes the advice $z$ of length $S$ bits as part of the encoding string. Then (somewhat similarly to the MI game), it runs $A_1$ sequentially using $z$ on multiple DLOG instances, and the goal is to gain beyond $S$ bits in the encoding length, establishing a contradiction.[7]

In each instance, the encoding procedure encodes the query answers of $A_1$, so that the decoding procedure can repeat the same steps. Assuming that $A_1$ is deterministic, this is sufficient (otherwise, shared randomness is used). A saving in the encoding length is obtained when $A_1$ answers the instance correctly, as this answer gives information about $\sigma$ (of at most $\log N$ bits) that does not need to be encoded.

For simplicity, we assume that $A$ answers correctly with constant probability. We shall show that after answering $\tilde{\Omega}(N/T^2)$ instances, a saving cannot be obtained (with high probability). Thus, the total saving is bounded by $\tilde{O}(N/T^2)$ bits and the technique cannot be used to prove a time-space tradeoff that is better than $ST^2 \geq \tilde{\Omega}(N)$. Once again, this time-space tradeoff is tight for adaptive algorithms, and this technique cannot distinguish between adaptive and non-adaptive algorithms (up to logarithmic factors).

[7] These DLOG instances have non-uniformly distributed secrets, on which the success of $A_1$ is not guaranteed. However, this problem is addressed using the random self-reducibility property of the DLOG problem, which allows to execute $A_1$ on instances where the secret is uniformly distributed.

It remains to show that after answering $\tilde{\Omega}(N/T^2)$ instances, a saving cannot be obtained (with high probability). The argument here is similar to the one used for concentration inequalities. At the threshold of $\tilde{\Omega}(N/T^2)$ instances, about $\tilde{\Omega}(N/T)$ queries were already made. Therefore, it is very likely that at least one of the $T$ queries for the new instance will collide with a previous one. Such a collision query gives no new information about $\sigma$, hence encoding it in a standard way, using about $\log N$ bits, results in a loss that nullifies the encoding advantage from running $A_1$.

One may try to encode collisions in a special way to mitigate the loss by encoding the specific query indices that collide. However, at the threshold of $\tilde{\Omega}(N/T^2)$ instances (where $\tilde{\Omega}(N/T)$ queries have already been issued), encoding the query pair seems to require more than $\log T + (\log N - \log T) = \log N$ bits, once again nullifying the encoding advantage from running $A_1$.

# B Non-adaptive sqDDH Algorithm

We sketch the details of the non-adaptive variant of the adaptive sqDDH algorithm of [CK18]. Essentially, the algorithm replaces the adaptive random walk of [CK18] with a sequence that can be computed non-adaptively.

First, define a biased pseudorandom predicate $\mathrm{P} : [N]^2 \to \{0,1\}$ that evaluates to 1 with probability about $T^{-1}$. We call a pair of the form $(\sigma(x), \sigma(x^2))$ (where $x \in [N]$) special if $\mathrm{P}(\sigma(x), \sigma(x^2)) = 1$.

Second, define a balanced pseudorandom predicate $\mathrm{Q} : [N]^2 \to \{0,1\}$ that evaluates to 1 with probability $1/2$.

Third, define a pseudorandom function $g : [N]^2 \to [S]$ that evaluates to each $i \in [S]$ with probability about $S^{-1}$.

Given $\sigma$, for each $v \in [S]$, define the set

$$\mathcal{L}_v = \{(\sigma(x), \sigma(x^2)) \in [N]^2 \colon \mathrm{P}(\sigma(x), \sigma(x^2)) = 1 \wedge g(\sigma(x), \sigma(x^2)) = v\},$$

which contains roughly $\frac{N}{ST}$ pairs. Moreover, define the majority value of Q on $\mathcal{L}_v$ by

$$\mathrm{Maj}_v = \mathrm{Majority}\{\mathrm{Q}(\sigma(x), \sigma(x^2)) \colon (\sigma(x), \sigma(x^2)) \in \mathcal{L}_v\}.$$

A standard probabilistic argument shows that with high probability, $\mathrm{Maj}_v$ agrees with a pair chosen uniformly form $\mathcal{L}_v$ on the value of Q with probability about $\frac{1}{2} + \sqrt{\frac{ST}{N}}$. Namely, with high probability,

$$\Pr_{(\sigma(x),\sigma(x^2))\sim\mathcal{L}_v}[\mathrm{Q}(\sigma(x), \sigma(x^2)) = \mathrm{Maj}_v] \geq \frac{1}{2} + \Omega\Big(\sqrt{\frac{ST}{N}}\Big).$$

The preprocessing algorithm $A_0$ works by calculating and storing $\mathrm{Maj}_v$ for each $v \in [S]$ in the advice string.

The online algorithm $A_1$ defines a sequence of $T/2$ pairs in $[N]^2$ that can be computed using non-adaptive queries and preserve the sqDDH relation. For example, for some pseudorandom function $f : [N] \to [N]$, starting from the secret $(d_1, d_2)$, define the sequence $(\sigma(d_1), \sigma(d_2)), (\sigma(d_1 \cdot f(1)), \sigma(d_2 \cdot f(1)^2)), (\sigma(d_1 \cdot f(2)), \sigma(d_2 \cdot f(2)^2)) \ldots$.

With high probability, one of these $T/2$ pairs evaluates to 1 on P (otherwise, $A_1$ guesses the answer). Denote the first such pair by $(y_1, y_2)$. $A_1$ computes $v := g(y_1, y_2)$ and $b := \mathrm{Q}(y_1, y_2)$. It then returns 1 if $b = \mathrm{Maj}_v$, and 0 otherwise.

The main observation in the analysis is that on a YES instance, $b = \mathrm{Maj}_v$ with probability about $\frac{1}{2} + \sqrt{\frac{ST}{N}}$ (as noted above). On the other hand, on a NO instance, $b = \mathrm{Maj}_v$ with probability of about $\frac{1}{2}$, since Q is balanced and uncorrelated with pairs computed for NO instances.

# C A Variant of Shearer's Lemma for Bijections

In this appendix we consider Theorem 1.8 – namely, the variant of Shearer's lemma for bijections. First, we prove that the theorem follows from [BCELM11, Proposition 21] of Barthe, Cordero-Erausquin, Ledoux and Maurey and from [CS24, Theorem 4] of Caputo and Salez. Then we present an elementary proof of a slightly weaker version of the theorem in which the constant 2 is replaced by 9.

## C.1 Deduction of Theorem 1.8 from the results of [BCELM11, CS24]

In this subsection we show how Theorem 1.8 follows from [BCELM11, Proposition 21] and from [CS24, Theorem 4], which are stated in a very different form. Let us recall Theorem 1.8.

**Theorem 1.8.** Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. Then

$$2k \cdot \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) \geq \sum_{j\in[m]} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}),$$

where $P_{X_{\mathcal{U}}}$ is the distribution of the vector $X_{\mathcal{U}} := (X_i \mid i \in \mathcal{U})$ with respect to $P$ (and analogously for $Q$).

Since the equivalence between [BCELM11, Proposition 21] and [CS24, Theorem 4] is discussed in a remark after [CS24, Theorem 4], we show how Theorem 1.8 follows from [CS24, Theorem 4]. We first cite this result and then explain the notations involved.

**Theorem C.1** ([CS24, Theorem 4, Equation (34)])**.** *For any choice of a probability vector $(\theta_A, A \subseteq [N])$ on subsets of $[N]$ and every function $f : S_N \to \mathbb{R}_{\geq 0}$,*

$$\sum_{A\subseteq[N]} \theta_A \, \mathit{Ent}(E_A[f]) \leq (1-\kappa)\, \mathit{Ent}(f),$$

*where $\kappa = \min_{i\neq j} \sum_{A\supseteq\{i,j\}} \theta_A$.*

The reader is referred to [CS24] for a full description of the functional Ent and its properties. Here we only state two basic observations, that follow immediately from its definitions:

**Observation C.2.** *Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections, and let $f : S_N \to \mathbb{R}$ be defined as $f(\sigma) = \frac{P(\sigma)}{Q(\sigma)}$. Let $A \subseteq [N]$ be a set of indices. Then:*

- *$\mathit{Ent}(f) = \mathrm{D}_{KL}(P\|Q)$.*
- *$\mathit{Ent}(E_A[f]) = \mathrm{D}_{KL}(P_{X_{A^c}} \| Q_{X_{A^c}})$ (note that the distributions are projected to the complement set $A^c = [N] \setminus A$ and not to $A$).*

Using these observations, we show that Theorem 1.8 follows directly from Theorem C.1.

*Proof (of Theorem 1.8).* Let $P, Q, f$ be as in Observation C.2. Let $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. For every $A \subseteq [N]$, we denote $\theta_A = |\{j \mid \mathcal{U}_j = A^c\}|/m$. Applying Theorem C.1 we obtain

$$\sum_{A\subseteq[N]} \theta_A \, \mathrm{Ent}(\mathrm{E}_A[f]) \leq (1-\kappa)\, \mathrm{Ent}(f),$$

where $\kappa = \min_{i \neq i'} \sum_{A \supseteq \{i,i'\}} \theta_A$. By Observation C.2, an equivalent formula is

$$\sum_{A \subseteq [N]} \theta_A \, \mathrm{D}_{\mathrm{KL}}(P_{X_{A^c}} \| Q_{X_{A^c}}) \leq (1-\kappa) \, \mathrm{D}_{\mathrm{KL}}(P \| Q).$$

Substituting the formula for $\theta_A$, we obtain

$$\frac{1}{m} \sum_{j=1}^{m} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}) \leq (1-\kappa) \, \mathrm{D}_{\mathrm{KL}}(P \| Q). \tag{3}$$

It remains to bound $\kappa$. Notice that $\kappa = \min_{i \neq i'} \sum_{A \supseteq \{i,i'\}} \theta_A = \frac{1}{m} \min_{i \neq i'} |\{j \mid \{i,i'\} \subseteq \mathcal{U}_j^c\}|$. From the assumption on $\mathcal{U}_j$ we obtain that $|\{j \mid i \in \mathcal{U}_j^c\}| \geq m - k$ for all $i$, implying that $\kappa \geq \frac{m-2k}{m}$. The statement now follows from (3). ■

## C.2 An elementary proof of a slighly weaker version of the theorem

In this subsection we present an elementary proof of a slightly weaker version of Theorem 1.8, in which the constant 2 is replaced by 9. Specifically, we prove the following.

**Theorem C.3.** *Let $\mathcal{X}$ be a set of size $N$. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution over bijections from $[N]$ to $\mathcal{X}$, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution over such bijections. Let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ be subsets of $[N]$, such that each $i \in [N]$ belongs to at most $k$ of them. Then*

$$9k \cdot \mathrm{D}_{KL}(P_X \| Q_X) \geq \sum_{j \in [m]} \mathrm{D}_{KL}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}),$$

*where $P_{X_{\mathcal{U}}}$ is the distribution of the vector $X_{\mathcal{U}} := (X_i \mid i \in \mathcal{U})$ with respect to $P$ (and analogously for $Q$).*

We need several tools in order to prove the theorem. Let us begin with the following technical lemma:

**Lemma C.4.** *Let $N \in \mathbb{N}$ and let $\ell \leq \frac{N}{4}$. Let $p_1, \ldots, p_\ell \in (0,1)$ with $\sum_i p_i \leq 1$, and let us denote $\epsilon_i = p_i - \frac{1}{N}$. Denote also $p' = \frac{1 - \sum_i p_i}{N - \ell}$ and $\epsilon' = p' - \frac{1}{N}$. Then the following holds:*

$$Np' \log\left(Np'\right) - N\epsilon' \leq 4 \sum_i \left(p_i \log(Np_i) - \epsilon_i\right).$$

*Proof.* The proof utilizes the following inequalities (recall that logarithms are in base $e$):

(i) For all $x > 0$ it holds that $x \log x - x + 1 \geq 0$.

(ii) For all $0 < x \leq 2$ it holds that $x \log x - x + 1 \leq (x-1)^2 \leq |x-1|$.

(iii) For all $x < 5$ it holds that $x \log x - x + 1 \geq \frac{1}{4}(x-1)^2$.

(iv) For all $x \geq 5$ it holds that $x \log x - x + 1 \geq |x-1|$.

All these inequalities can be easily proved by basic analytical tools.

We prove the statement by case-analysis. Let us first assume that

$$\sum_{i : Np_i \geq 5} |\epsilon_i| \geq \frac{1}{2} \sum_{i : Np_i < 5} |\epsilon_i| .$$

We obtain

$$\sum_i |\epsilon_i| \leq 3 \sum_{i : Np_i \geq 5} |\epsilon_i| .$$

Since $\ell \leq \frac{N}{4}$ and following the definition of $\epsilon'$, we may also obtain

$$\left|N\epsilon'\right| \leq \frac{4}{3}\sum_i |\epsilon_i| \leq 4 \sum_{i:Np_i \geq 5} |\epsilon_i| \,. \tag{4}$$

Therefore, we may bound:

$$\begin{aligned}
Np' \log\left(Np'\right) - N\epsilon' &= (Np')\log(Np') - Np' + 1 \\
&\leq \left|N\epsilon'\right| && \text{ineq. (ii) with } x = Np' \\
&\leq 4 \sum_{i:Np_i \geq 5} |\epsilon_i| && \text{Equation (4)} \\
&\leq \frac{4}{N} \sum_{i:Np_i \geq 5} (Np_i \log(Np_i) - Np_i + 1) && \text{ineq. (iv) with } x = Np_i \\
&= 4 \sum_{i:Np_i \geq 5} (p_i \log(Np_i) - \epsilon_i) \\
&\leq 4 \sum_i (p_i \log(Np_i) - \epsilon_i) && \text{ineq. (i) with } x = Np_i,
\end{aligned}$$

as required.

Next, we assume

$$\sum_{i:Np_i \geq 5} |\epsilon_i| < \frac{1}{2} \sum_{i:Np_i < 5} |\epsilon_i| \,,$$

implying that

$$\left|N\epsilon'\right| \leq \frac{4}{3}\sum_i |\epsilon_i| \leq 2 \sum_{i:Np_i < 5} |\epsilon_i| \,. \tag{5}$$

Therefore we may bound:

$$\begin{aligned}
Np' \log\left(Np'\right) - N\epsilon' &= (Np')\log(Np') - Np' + 1 \\
&\leq \left(N\epsilon'\right)^2 && \text{ineq. (ii) with } x = Np' \\
&\leq \left(2 \sum_{i:Np_i < 5} |\epsilon_i|\right)^2 && \text{Equation (5)} \\
&\leq 4\ell \sum_{i:Np_i < 5} \epsilon_i^2 && \text{Cauchy–Schwarz inequality} \\
&\leq \frac{16\ell}{N} \sum_{i:Np_i < 5} (p_i \log(Np_i) - \epsilon_i) && \text{ineq. (iii) with } x = Np_i \\
&\leq 4 \sum_{i:Np_i < 5} (p_i \log(Np_i) - \epsilon_i) && 4\ell \leq N \\
&\leq 4 \sum_i (p_i \log(Np_i) - \epsilon_i) && \text{ineq. (i) with } x = Np_i,
\end{aligned}$$

as required. ■

Lemma C.4 allows proving a variant of Shearer's inequality for indicator vectors.

**Lemma C.5** (Variant of Shearer's inequality for indicator vectors)**.** *Let $N$ be a positive integer. Let $Q_X = Q_{X_1,\ldots,X_N}$ be the uniform distribution function over the set $\binom{[N]}{1} := \{v \in \{0,1\}^N \mid \sum_i v_i = 1\}$ of indicator vectors, and let $P_X = P_{X_1,\ldots,X_N}$ be another distribution function over the same set. Let $\mathcal{U}_1, \ldots, \mathcal{U}_m \subseteq [N]$ be such that for each $i \in [N]$, $|\{j \in [m] \mid i \in \mathcal{U}_j\}| \leq k$. Then*

$$9k \cdot \mathrm{D}_{KL}(P_X \| Q_X) \geq \sum_{j \in [m]} \mathrm{D}_{KL}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}).$$

*Proof.* Denote $S = \mathrm{D}_{\mathrm{KL}}(P_X \| Q_X)$. By applying Property (8) we obtain that for every $\mathcal{U} \subseteq [N]$ we have $\mathrm{D}_{\mathrm{KL}}(P_{X_\mathcal{U}} \| Q_{X_\mathcal{U}}) \leq S$. Since the number of sets $\mathcal{U}_j$ with $|\mathcal{U}_j| > \frac{N}{4}$ is at most $4k$, we may assume that $|\mathcal{U}_j| \leq \frac{N}{4}$ and prove that

$$\sum_{j \in [m]} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}) \leq 5kS.$$

Moreover, since $\mathrm{D}_{\mathrm{KL}}(P_{X_i} \| Q_{X_i}) \geq 0$ following Property (4), we may assume that $|\{j \in [m] \mid i \in \mathcal{U}_j\}| = k$ for all $i \in [N]$.

For every $i \in [N]$, we notice that $Q_{X_i}(1) = \frac{1}{N}$ (i.e., the probability of the event $X_i = 1$ when sampled with respect to $Q$ is $1/N$) and denote $p_i = P_{X_i}(1)$ and $\epsilon_i = p_i - \frac{1}{N}$, to obtain by definition $S = \sum_i p_i \log(Np_i)$. Additionally, for every set $\mathcal{U} \subseteq [N]$ we denote

$$p'(\mathcal{U}) = \frac{1 - \sum_{i \in \mathcal{U}} p_i}{N - |\mathcal{U}|} \quad \text{and} \quad \epsilon'(\mathcal{U}) = p'(\mathcal{U}) - \frac{1}{N} = -\frac{\sum_{i \in \mathcal{U}} \epsilon_i}{N - |\mathcal{U}|}.$$

Let $\mathcal{U} \subseteq [N]$ be an index set. By the definition of KL-divergence, we may obtain:

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(P_{X_\mathcal{U}} \| Q_{X_\mathcal{U}}) &= \sum_{v \in \mathrm{Supp}(X_\mathcal{U})} P_{X_\mathcal{U}}(v) \cdot \log\left(\frac{P_{X_\mathcal{U}}(v)}{Q_{X_\mathcal{U}}(v)}\right) \\
&= \sum_{i \in \mathcal{U}} p_i \log(Np_i) + \left(1 - \sum_{i \in \mathcal{U}} p_i\right) \log\left(\frac{1 - \sum_{i \in \mathcal{U}} p_i}{1 - \frac{|\mathcal{U}|}{N}}\right) \\
&= \sum_{i \in \mathcal{U}} p_i \log(Np_i) + (N - |\mathcal{U}|) p'(\mathcal{U}) \log\left(Np'(\mathcal{U})\right).
\end{aligned}$$

Notably, $\sum_{j \in [m]} \sum_{i \in \mathcal{U}_j} p_i \log(Np_i) = kS$, so it remains to bound

$$\sum_{j \in [m]} (N - |\mathcal{U}_j|) p'(\mathcal{U}_j) \log\left(Np'(\mathcal{U}_j)\right).$$

Since $|\{j \in [m] \mid i \in \mathcal{U}_j\}| = k$ is assumed to be constant, we obtain that

$$\sum_{j \in [m]} (N - |\mathcal{U}_j|) \epsilon'(\mathcal{U}_j) = -\sum_i k\epsilon_i = 0.$$

Therefore, we may bound:

$$\begin{aligned}
\sum_{j \in [m]} (N - |\mathcal{U}_j|) p'(\mathcal{U}_j) \log\left(Np'(\mathcal{U}_j)\right) &= \sum_{j \in [m]} (N - |\mathcal{U}_j|) \left(p'(\mathcal{U}_j) \log\left(Np'(\mathcal{U}_j)\right) - \epsilon'(\mathcal{U}_j)\right) \\
&\leq \sum_{j \in [m]} N \left(p'(\mathcal{U}_j) \log\left(Np'(\mathcal{U}_j)\right) - \epsilon'(\mathcal{U}_j)\right) \\
&\leq 4 \sum_{j \in [m]} \sum_{i \in \mathcal{U}_j} (p_i \log(Np_i) - \epsilon_i) \\
&= 4kS,
\end{aligned}$$

where the first inequality follows from the fact that $(p'(\mathcal{U}_j) \log(Np'(\mathcal{U}_j)) - \epsilon'(\mathcal{U}_j)) \geq 0$ (implied by ineq. (i) from the proof of Lemma C.4), and the second inequality follows from Lemma C.4. ■

In order to prove Theorem C.3, we need another lemma which shows that under an additional assumption, conditioning does not decrease KL-divergence.

**Lemma C.6.** *Let $P$ and $Q$ be two distributions on pairs $(X, Y)$. Suppose that $X$ and $Y$ are independent with respect to $Q$ (i.e., $Q_{X,Y} = Q_X Q_Y$). Then*

$$\mathrm{D}_{KL}(P_Y \| Q_Y) \leq \mathrm{D}_{KL}(P_{Y|X} \| Q_{Y|X}).$$

*Proof.* Since $Q_{X,Y} = Q_X Q_Y$, we obtain

$$Q_Y(y) = \sum_x P_X(x) Q_{Y|X=x}(y).$$

Moreover, by the law of total probability, we obtain

$$P_Y(y) = \sum_x P_X(x) P_{Y|X=x}(y).$$

Therefore, we obtain

$$\mathrm{D}_{\mathrm{KL}}(P_Y \| Q_Y) = \mathrm{D}_{\mathrm{KL}}\left(\left(\sum_x P_X(x) P_{Y|X=x}(y)\right) \| \left(\sum_x P_X(x) Q_{Y|X=x}(y)\right)\right).$$

Finally, we may apply Property (5) inductively to obtain

$$\mathrm{D}_{\mathrm{KL}}(P_Y \| Q_Y) \leq \sum_x P_X(x)\, \mathrm{D}_{\mathrm{KL}}(P_{Y|X=x} \| Q_{Y|X=x}) =: \mathrm{D}_{\mathrm{KL}}(P_{Y|X} \| Q_{Y|X}).$$

■

Now we are ready to prove Theorem C.3.

*Proof (Proof of Theorem C.3).* First we set some notations, given a bijection $X = X_1, \ldots, X_N : [N] \to \mathcal{X}$, indices $i, j \in [N]$ and a set $\mathcal{S} \subseteq [N]$:

$$\begin{aligned} X_{i \to j} &= \begin{cases} 1 & X_i = j, \\ 0 & \text{otherwise;} \end{cases} \\ X_{\to j} &= X^{-1}(j), \\ X_{\mathcal{S} \to j} &= \begin{cases} X^{-1}(j) & X^{-1}(j) \in \mathcal{S}, \\ * & \text{otherwise.} \end{cases} \end{aligned}$$

Note that $X = X_1, \ldots, X_N$ and the sequence $X_{\to 1}, \ldots, X_{\to N}$ are equivalent in the sense that they both encode the same information about the permutation. Similarly, $X_i$ is equivalent to $(X_{i \to 1}, \ldots, X_{i \to N})$, and $X_{\mathcal{S}}$ is equivalent to $(X_{\mathcal{S} \to 1}, \ldots, X_{\mathcal{S} \to N})$.

By Property (7), we obtain:

$$\mathrm{D}_{\mathrm{KL}}(P_X \| Q_X) = \mathrm{D}_{\mathrm{KL}}(P_{X_{\to 1}} \| Q_{X_{\to 1}}) + \mathrm{D}_{\mathrm{KL}}\left(P_{X_{\to 2}, \ldots, X_{\to N} | X_{\to 1}} \| Q_{X_{\to 2}, \ldots, X_{\to N} | X_{\to 1}}\right). \tag{6}$$

Similarly, given a set $\mathcal{U}$, we obtain:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}}} \| Q_{X_{\mathcal{U}}}) = &\mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U} \to 1}} \| Q_{X_{\mathcal{U} \to 1}}) + \\ &+ \mathrm{D}_{\mathrm{KL}}\left(P_{X_{\mathcal{U} \to 2}, \ldots, X_{\mathcal{U} \to N} | X_{\mathcal{U} \to 1}} \| Q_{X_{\mathcal{U} \to 2}, \ldots, X_{\mathcal{U} \to N} | X_{\mathcal{U} \to 1}}\right). \end{aligned} \tag{7}$$

By Lemma C.5, we have

$$\sum_{j \in [m]} \mathrm{D}_{\mathrm{KL}}(P_{X_{\mathcal{U}_j \to 1}} \| Q_{X_{\mathcal{U}_j \to 1}}) \leq 9k \cdot \mathrm{D}_{\mathrm{KL}}(P_{X_{\to 1}} \| Q_{X_{\to 1}}). \tag{8}$$

In addition, notice that the conditioned variable $X_{\mathcal{U} \to 2}, \ldots, X_{\mathcal{U} \to N} \mid X_{\mathcal{U} \to 1}$, distributed with respect to $Q$, is independent of $X_{\to 1}$. Indeed, if $X_{\mathcal{U} \to 1} = i \in \mathcal{U}$ then $X_{\to 1} = i$ deterministically.

Otherwise, $X_{\mathcal{U}\to 1} = *$, implying that $X_{\mathcal{U}}$ is a uniform sequence of $|\mathcal{U}|$ distinct elements other than 1 (since $Q$ is the uniform distribution on bijections). For any $i \notin \mathcal{U}$, conditioning on $X_{\to 1} = i$ has no effect on this distribution. Therefore, we may apply Lemma C.6 to obtain:

$$\mathrm{D_{KL}}\left(P_{X_{\mathcal{U}\to 2},\ldots,X_{\mathcal{U}\to N}|X_{\mathcal{U}\to 1}} \| Q_{X_{\mathcal{U}\to 2},\ldots,X_{\mathcal{U}\to N}|X_{\mathcal{U}\to 1}}\right) \leq \mathrm{D_{KL}}\left(P_{X_{\mathcal{U}\to 2},\ldots,X_{\mathcal{U}\to N}|X_{\to 1}} \| Q_{X_{\mathcal{U}\to 2},\ldots,X_{\mathcal{U}\to N}|X_{\to 1}}\right).$$

The distribution $Q_{X_{\mathcal{U}\to 2},\ldots,X_{\mathcal{U}\to N}|X_{\to 1}}$ is equivalent to a uniform distribution over bijections on $N-1$ elements. Therefore, we may apply an inductive argument to obtain

$$\begin{aligned}
&\sum_{j\in[m]} \mathrm{D_{KL}}\left(P_{X_{\mathcal{U}_j\to 2},\ldots,X_{\mathcal{U}_j\to N}|X_{\mathcal{U}_j\to 1}} \| Q_{X_{\mathcal{U}_j\to 2},\ldots,X_{\mathcal{U}_j\to N}|X_{\mathcal{U}_j\to 1}}\right) \\
&\qquad \leq \sum_{j\in[m]} \mathrm{D_{KL}}\left(P_{X_{\mathcal{U}_j\to 2},\ldots,X_{\mathcal{U}_j\to N}|X_{\to 1}} \| Q_{X_{\mathcal{U}_j\to 2},\ldots,X_{\mathcal{U}_j\to N}|X_{\to 1}}\right) \\
&\qquad \leq 9k \cdot \mathrm{D_{KL}}\left(P_{X_{\to 2},\ldots,X_{\to N}|X_{\to 1}} \| Q_{X_{\to 2},\ldots,X_{\to N}|X_{\to 1}}\right).
\end{aligned}$$

Combining with Equations (6), (7), and (8), we obtain

$$\sum_{j\in[m]} \mathrm{D_{KL}}(P_{X_{\mathcal{U}_j}} \| Q_{X_{\mathcal{U}_j}}) \leq 9k \cdot \mathrm{D_{KL}}(P_X \| Q_X),$$

completing the proof. ■

# D Improved Bound for Problems without Post-Processing

Recall Theorem 4.1. A slightly better bound can be achieved for problems with a trivial post-processing function $\boldsymbol{post}(d, j) = j$. This rule captures the DLOG and DDH problems, but does not capture EM key recovery.

**Theorem D.1.** *Let $\boldsymbol{PC} := \boldsymbol{PC}(N, \mathcal{D}, \mathcal{M}, \boldsymbol{tr}, \boldsymbol{post}, \boldsymbol{suc})$ be a permutation challenge game, such that $\boldsymbol{tr}$ is $u$-uniform and $\boldsymbol{post}(d, j) = j$ is a trivial post-processing function. Let $(A_0, A_1)$ be an $(S, T)$ non-adaptive algorithm with preprocessing for $\boldsymbol{PC}$. Denote by $\boldsymbol{MaxS}(T)$ the optimal success probability (with respect to $\boldsymbol{suc}$) of a non-preprocessing, non-adaptive algorithm that makes at most $T$ queries. Then, the success probability of $(A_0, A_1)$ is at most*

$$\min\left(2 \cdot \boldsymbol{MaxS}(T) + \frac{4\log(2)ST}{u}, \boldsymbol{MaxS}(T) + \sqrt{\frac{\log(2)ST}{u}}\right).$$

*Proof.* By Theorem 4.9, the success probability of $(A_0, A_1)$ is at most

$$\min\left(2 \cdot \widehat{\boldsymbol{MaxS}}(T) + \frac{4\log(2)ST}{u}, \widehat{\boldsymbol{MaxS}}(T) + \sqrt{\frac{\log(2)ST}{u}}\right),$$

where $\widehat{\boldsymbol{MaxS}}(T)$ is the optimal success probability of an adversary to $\boldsymbol{MID}$ with running time $T$. Thus, it remains to show that $\widehat{\boldsymbol{MaxS}}(T) \leq \boldsymbol{MaxS}(T)$ and complete the proof. Let $\widehat{A} = (\widehat{A}_0, \widehat{A}_1)$ be an adversary to $\boldsymbol{MID}$ with success probability $\widehat{\boldsymbol{MaxS}}(T)$. Let us describe a non-preprocess algorithm $A$ to the problem:

1. $A$ receives $N$ and a non-adaptive oracle access by inner and outer queries to the permutation $\sigma$ with respect to a secret $d$.
2. $A$ runs $\widehat{A}_0(N)$ to obtain the sequences $I = (I_1, \ldots, I_{T_1})$ and $O = (O_1, \ldots, O_{T_1})$.
3. $A$ runs $\widehat{A}_1$ and obtains its queries (recall that $\widehat{A}_1$ is allowed to make only outer queries), $m_1, \ldots, m_{T_2} \in \mathcal{M}$.

4. $A$ makes the inner queries $I_1, \ldots, I_{T_1}$ and obtains $O'_1, \ldots, O'_{T_1}$. Additionally, $A$ makes the outer queries $m_1, \ldots, m_{T_2}$. Since the post-processing function is assumed to be trivial, the outputs are of the form $r_i := \sigma(\boldsymbol{tr}(d, m_i))$.

5. Using some deterministic process that depends only on $\{O_1, \ldots, O_{T_1}, O'_1, \ldots, O'_{T_1}\}$, $A$ constructs a permutation $\pi$ with $\pi(O'_i) = O_i$ for all $i$.

6. $A$ simulates $\widehat{A}_1$'s oracle, and returns $\pi(r_i)$ as the response to the query $m_i$. $A$ returns $\widehat{A}_1$'s output.

Clearly, the permutation $\pi \circ \sigma$ is uniformly distributed over the permutations with $\forall i : I_i \mapsto O_i$. Moreover, the responses $\widehat{A}_1$ obtained are $\pi(r_i) = (\pi \circ \sigma)(\boldsymbol{tr}(d, m_i))$. Therefore, the distribution of the permutation $\pi \circ \sigma$ is exactly as it is supposed to be, and the responses to all queries are correct with respect to it. Therefore, the simulation is accurate and does not affect the success probability. Thus, the success probability of $A$ is exactly $\widehat{\boldsymbol{MaxS}}(T)$. ■

**Acknowledgments.**

We thank the anonymous reviewers for their useful comments. In particular, we thank a reviewer who pointed out an error in the proof of Theorem 4.9 in a previous version of this paper.

## References

[ABC+25] Akshima, Tyler Besselman, Kai-Min Chung, Siyao Guo, and Tzu-Yi Yang. Tight quantum time-space tradeoffs for permutation inversion, arxiv:2510.12112. 2025.

[ABG+24] Akshima, Tyler Besselman, Siyao Guo, Zhiye Xie, and Yuping Ye. Tight time-space tradeoffs for the decisional diffie-hellman problem. In *STOC 2024*, pages 1739–1749. ACM, 2024.

[ACDW20] Akshima, David Cash, Andrew Drucker, and Hoeteck Wee. Time-Space Tradeoffs and Short Collisions in Merkle-Damgård Hash Functions. In *CRYPTO 2020*, volume 12170 of *Lecture Notes in Computer Science*, pages 157–186. Springer, 2020.

[AGL24] Akshima, Siyao Guo, and Qipeng Liu. Time-space lower bounds for finding collisions in merkle-damgård hash functions. *J. Cryptol.*, 37(2):10, 2024.

[BCELM11] Franck Barthe, Dario Cordero-Erausquin, Michel Ledoux, and Bernard Maurey. Correlation and Brascamp–Lieb inequalities for Markov semigroups. *Internat. Math. Res. Notices*, 2011(10):2177–2216, 2011.

[BL13] Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2013.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The Distinction Between Fixed and Random Generators in Group-Based Assumptions. In *CRYPTO 2019*, volume 11693 of *Lecture Notes in Computer Science*, pages 801–830. Springer, 2019.

[BNS92] László Babai, Noam Nisan, and Mario Szegedy. Multiparty Protocols, Pseudorandom Generators for Logspace, and Time-Space Trade-Offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992.

[BW00] Alex Biryukov and David A. Wagner. Advanced slide attacks. In *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer, 2000.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, 2018.

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random Oracles and Non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10820 of *Lecture Notes in Computer Science*, pages 227–258. Springer, 2018.

[CGFS86] Fan R. K. Chung, Ronald L. Graham, Peter Frankl, and James B. Shearer. Some intersection theorems for ordered sets and graphs. *J. Comb. Theory A*, 43(1):23–37, 1986.

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In *FOCS 2020*, pages 673–684. IEEE, 2020.

[CHM20] Dror Chawin, Iftach Haitner, and Noam Mazor. Lower bounds on the time/memory tradeoff of function inversion. In *TCC 2020*, volume 12552 of *Lecture Notes in Computer Science*, pages 305–334. Springer, 2020.

[CK18] Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In *EUROCRYPT 2018*, volume 10821 of *Lecture Notes in Computer Science*, pages 415–447. Springer, 2018.

[CK19] Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In *TCC 2019*, volume 11891 of *Lecture Notes in Computer Science*, pages 393–421. Springer, 2019.

[CL23] Eldon Chung and Kasper Green Larsen. Stronger 3SUM-Indexing Lower Bounds. In Nikhil Bansal and Viswanath Nagarajan, editors, *SODA 2023*, pages 444–455. SIAM, 2023.

[CLL+18] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round even-mansour cipher. *J. Cryptol.*, 31(4):1064–1119, 2018.

[CLS15] Benoit Cogliati, Rodolphe Lampe, and Yannick Seurin. Tweaking Even-Mansour ciphers. In *CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 189–208. Springer, 2015.

[CS24] Pietro Caputo and Justin Salez. Entropy factorization via curvature. *CoRR*, abs/2407.13457, 2024.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2nd ed.)*. Wiley, 2006.

[Dae91] Joan Daemen. Limitations of the Even-Mansour construction. In *ASIACRYPT 1991*, volume 739 of *Lecture Notes in Computer Science*, pages 495–498. Springer, 1991.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited. In *EUROCRYPT 2017*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495, 2017.

[DKKS21] Pavel Dvorák, Michal Koucký, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures. In *ESA 2021*, volume 204 of *LIPIcs*, pages 39:1–39:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[DKS15] Orr Dunkelman, Nathan Keller, and Adi Shamir. Slidex attacks on the Even-Mansour encryption scheme. *J. Cryptol.*, 28(1):1–28, 2015.

[DSST17] Yuanxi Dai, Yannick Seurin, John P. Steinberger, and Aishwarya Thiruvengadam. Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. In *CRYPTO 2017*, volume 10403 of *Lecture Notes in Computer Science*, pages 524–555. Springer, 2017.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptol.*, 10(3):151–162, 1997.

[FGK22] Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Time-space tradeoffs for sponge hashing: Attacks and limitations for short collisions. In *CRYPTO 2022*, volume 13509 of *Lecture Notes in Computer Science*, pages 131–160. Springer, 2022.

[FJM14] Pierre-Alain Fouque, Antoine Joux, and Chrysanthi Mavromati. Multi-user collisions: Applications to discrete logarithm, even-mansour and PRINCE. In *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 420–438. Springer, 2014.

[FN99] Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 1999.

[Gal14] David Galvin. Three tutorial lectures on entropy and counting. *CoRR*, abs/1406.7872, 2014.

[GGH+20] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3sum with preprocessing. In *STOC 2020*, pages 294–307. ACM, 2020.

[GGK24] Karthik Gajulapalli, Alexander Golovnev, and Samuel King. On the power of adaptivity for function inversion. In *ITC 2024*, volume 304 of *LIPIcs*, pages 5:1–5:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[GGPS23] Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. Revisiting time-space tradeoffs for function inversion. In *CRYPTO 2023*, volume 14082 of *Lecture Notes in Computer Science*, pages 453–481. Springer, 2023.

[GLLZ21] Siyao Guo, Qian Li, Qipeng Liu, and Jiapeng Zhang. Unifying Presampling via Concentration Bounds. In *TCC 2021*, volume 13042 of *Lecture Notes in Computer Science*, pages 177–208. Springer, 2021.

[GLSS15] Dmitry Gavinsky, Shachar Lovett, Michael E. Saks, and Srikanth Srinivasan. A tail bound for read-$k$ families of functions. *Random Struct. Algorithms*, 47(1):99–108, 2015.

[GT00] Rosario Gennaro and Luca Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic Constructions. In *FOCS 2000*, pages 305–313. IEEE Computer Society, 2000.

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

[HHTT21] Pooya Hatami, William M. Hoza, Avishay Tal, and Roei Tell. Fooling constant-depth threshold circuits (extended abstract). In *FOCS 2021*, pages 104–115. IEEE, 2021.

[HS24] Shuichi Hirahara and Nobutaka Shimizu. Planted clique conjectures are equivalent. In *STOC 2024*, pages 358–366. ACM, 2024.

[IK10] Russell Impagliazzo and Valentine Kabanets. Constructive Proofs of Concentration Bounds. In *APPROX-RANDOM 2010*, volume 6302 of *Lecture Notes in Computer Science*, pages 617–631. Springer, 2010.

[JN03] Antoine Joux and Kim Nguyen. Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptol.*, 16(4):239–247, 2003.

[Kah22] Jeff Kahn. Hitting times for Shamir's problem. *Trans. Amer. Math. Soc.*, 375(1):627–668, 2022.

[Knu69] Donald E. Knuth. *The Art of Computer Programming, vol. II: Seminumerical Algorithms*. Addison-Wesley, 1969.

[KSWY25] Dmitriy Kunisky, Daniel A. Spielman, Alexander S. Wein, and Xifan Yu. Statistical inference of a ranked community in a directed graph. In *STOC 2025*, pages 2107–2117. ACM, 2025.

[Mih10] Joseph P. Mihalcik. *An analysis of algorithms for solving discrete logarithms in fixed groups, Master's thesis*. Naval Postgraduate School, Monterey, California, 2010.

[MW99] Ueli M. Maurer and Stefan Wolf. The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. *SIAM J. Comput.*, 28(5):1689–1721, 1999.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[RS18] Anup Rao and Makrand Sinha. Simplified Separation of Information and Communication. *Theory Comput.*, 14(1):1–29, 2018.

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.

[Sha71] Daniel Shanks. Class number, a theory of factorization and genera. In *Proc. Symp. Pure Math. 20*, pages 415–440. Providence, R.I.: American Mathematical Society, 1971.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Unr07] Dominique Unruh. Random Oracles and Auxiliary Input. In *CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 205–223. Springer, 2007.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS 1977*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977.

[Von10] Jan Vondrák. A note on concentration of submodular functions. *CoRR*, abs/1005.2791, 2010.

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *STOC 2005*, pages 523–532. ACM, 2005.