Permuted Puzzles and Cryptographic Hardness

Elette Boyle[1], Justin Holmgren[2], and Mor Weiss[1]

[1] Department of Computer Science, IDC Herzliya, Herzliya, Israel.
eboyle@alum.mit.edu,mor.weiss01@post.idc.ac.il
[2] Department of Computer Science, Princeton University, Princeton, New Jersey, USA. justin.holmgren@princeton.edu

Abstract. A *permuted puzzle* problem is defined by a pair of distributions $\mathcal{D}_0, \mathcal{D}_1$ over Σ^n. The problem is to distinguish samples from $\mathcal{D}_0, \mathcal{D}_1$, where the symbols of each sample are *permuted* by a single secret permutation π of $[n]$.

The conjectured hardness of specific instances of permuted puzzle problems was recently used to obtain the first candidate constructions of Doubly Efficient Private Information Retrieval (DE-PIR) (Boyle et al. & Canetti et al., TCC'17). Roughly, in these works the distributions $\mathcal{D}_0, \mathcal{D}_1$ over $\mathbb{F}^n$ are evaluations of either a moderately low-degree polynomial or a random function. This new conjecture seems to be quite powerful, and is the foundation for the first DE-PIR candidates, almost two decades after the question was first posed by Beimel et al. (CRYPTO'00). However, while permuted puzzles are a natural and general class of problems, their hardness is still poorly understood.

We initiate a formal investigation of the cryptographic hardness of permuted puzzle problems. Our contributions lie in three main directions:

- **Rigorous formalization.** We formalize a notion of permuted puzzle distinguishing problems, extending and generalizing the proposed permuted puzzle framework of Boyle et al. (TCC'17).
- **Identifying hard permuted puzzles.** We identify natural examples in which a one-time permutation *provably* creates cryptographic hardness, based on "standard" assumptions. In these examples, the original distributions $\mathcal{D}_0, \mathcal{D}_1$ are easily distinguishable, but the permuted puzzle distinguishing problem is computationally hard. We provide such constructions in the random oracle model, and in the plain model under the Decisional Diffie-Hellman (DDH) assumption. We additionally observe that the Learning Parity with Noise (LPN) assumption itself can be cast as a permuted puzzle.
- **Partial lower bound for the DE-PIR problem.** We make progress towards better understanding the permuted puzzles underlying the DE-PIR constructions, by showing that a toy version of the problem, introduced by Boyle et al. (TCC'17), withstands a rich class of attacks, namely those that distinguish solely via statistical queries.

1 Introduction

Computational hardness assumptions are the foundation of modern cryptography. The approach of building cryptographic systems whose security follows

from well-defined computational assumptions has enabled us to obtain fantastical primitives and functionality, pushing far beyond the limitations of information theoretic security. But, in turn, the resulting systems are only as secure as the computational assumptions lying beneath them. As cryptographic constructions increasingly evolve toward usable systems, gaining a deeper understanding of the true hardness of these problems—and the relationship between assumptions—is an important task.

To date, a relatively select cluster of structured problems have withstood the test of time (and intense scrutiny), to the point that assuming their hardness is now broadly accepted as "standard." These problems include flavors of factoring [RSA78,Rab79] and computing discrete logarithms [DH76], as well as certain computational tasks in high-dimensional lattices and learning theory [GKL88,BFKL93,Ajt96,BKW00,Ale03,Reg05]. A central goal in the foundational study of cryptography is constructing cryptographic schemes whose security provably follows from these (or weaker) assumptions.

In some cases, however, it may be beneficial—even necessary—to introduce and study *new* assumptions (indeed, every assumption that is "standard" today was at some point freshly conceived). There are several important cryptographic primitives (notable examples include indistinguishability obfuscation (IO) [BGI+01,GGH+13] and SNARKs [BCC+17]) that we do not currently know how to construct based on standard assumptions. Past experience has shown that acheiving new functionalities from novel assumptions, especially *falsifiable* assumptions [Nao03,GW11,GK16], can be a stepping stone towards attaining the same functionality from standard assumptions. This was the case for fully homomorphic encryption [RAD78,Gen09,BV11], as well as many recent primitives that were first built from IO and later (following a long line of works) based on more conservative assumptions (notably, non-interactive zero-knowledge protocols for NP based on LWE [KRR17,CCRR18,HL18,CCH+19,PS19], and the cryptographic hardness of finding a Nash equilibrium based on the security of the Fiat-Shamir heuristic [BPR15,HY17,CHK+19]). Finally, cryptographic primitives that can be based on diverse assumptions are less likely to "go extinct" in the event of a devastating new algorithmic discovery.

Of course, new assumptions should be introduced with care. We should strive to extract some intuitive reasoning justifying them, and some evidence for their hardness. A natural approach is to analyze the connection between the new assumption and known (standard) assumptions, with the ultimate goal of showing that the new assumption is, in fact, implied by a standard assumption. However, coming up with such a reduction usually requires deep understanding of the new assumption, which can only be obtained through a systematic study of it.

DE-PIR and permuted polynomials. A recent example is the new computational assumption underlying the construction of *Doubly Efficient Private Information Retrieval* (DE-PIR) [BIPW17,CHR17], related to pseudorandomness of permuted low-degree curves.

Private Information Retrieval (PIR) [CGKS95,KO97] schemes are protocols that enable a client to access entries of a database stored on a remote server (or

multiple servers), while hiding from the server(s) which items are retrieved. If no preprocessing of the database takes place, the security guarantee inherently requires the server-side computation to be linear in the size of the database for each incoming query [BIM00]. Database preprocessing was shown to yield computational savings in the multi-server setting [BIM00], but the goal of single-server PIR protocols with sublinear-time computation was a longstanding open question, with no negative results or (even heuristic) candidate solutions. Such a primitive is sometimes referred to as *Doubly Efficient (DE)* PIR.[3]

Recently, two independent works [BIPW17,CHR17] provided the first candidate constructions of single-server DE-PIR schemes, based on a new conjecture regarding the hardness of distinguishing *permuted* local-decoding queries (for a Reed-Muller code [Ree54,Mul54] with suitable parameters) from a uniformly random set of points. Specifically, although given the queries $\{z_1, \ldots, z_k\} \subseteq [N]$ of the local decoder it is possible to guess (with a non-trivial advantage) the index i which is being locally decoded, the conjectures of [BIPW17,CHR17] very roughly assert that adding a secret permutation can computationally hide i. More precisely, if an adversary instead sees (many) samples of sets of *permuted* queries $\{\pi(z_1), \ldots, \pi(z_k)\}$, where $\pi : [N] \to [N]$ is a secret fixed permutation (the same for all samples), then the adversary cannot distinguish these from independent uniformly random size-k subsets of $[N]$.

This new assumption (which we will refer to as PermRM, see Conjecture 1 in Section 6.2) allowed for exciting progress forward in the DE-PIR domain. But what do we really know about its soundness? Although [BIPW17,CHR17] provide some discussion and cryptanalysis of the assumption, our understanding of it is still far from satisfactory.

Permuted puzzles. The PermRM assumption can be cast as a special case in a broader family of hardness assumptions: as observed in [BIPW17], it can be thought of as an example of an instance where a secret random permutation seems to make an (easy) "distinguishing problem" hard, namely the permutation is the only sources of computational hardness. It should be intuitively clear that such permutations may indeed create hardness. For example, while one can easily distinguish a picture of a cat from that of a dog, this task becomes much more challenging when the pixels are permuted. There are also other instances in which random secret permutations were used to introduce hardness (see Section 1.2 below). Therefore, using permutations as a source of cryptographic hardness seems to be a promising direction for research, and raises the following natural question:

> *Under which circumstances can a secret random permutation be a source of cryptographic hardness?*

[3] Namely, computationally efficient for both client and server.

1.1 Our Results

We initiate a formal investigation of the cryptographic hardness of permuted puzzle problems. More concretely, our contributions can be summarized within the following three directions.

Rigorous formalization. We formalize a notion of *permuted puzzle distinguishing problems*, which extends and generalizes the proposed framework of [BIPW17]. Roughly, a permuted puzzle distinguishing problem is associated with a pair of distributions $\mathcal{D}_0, \mathcal{D}_1$ over strings in Σ^n, together with a random permutation π over $[n]$. The permuted puzzle consists of the distributions $\mathcal{D}_{0,\pi}, \mathcal{D}_{1,\pi}$ which are defined by sampling a string s according to $\mathcal{D}_0, \mathcal{D}_1$ (respectively), and permuting the entries of s according to π. A permuted puzzle is *computationally hard* if no efficient adversary can distinguish between a sample from $\mathcal{D}_{0,\pi}$ or $\mathcal{D}_{1,\pi}$, even given arbitrarily many samples of its choice from either of the distributions. We also briefly explore related hardness notions, showing that a weaker and simpler variant (which is similar to the one considered in [BIPW17]) is implied by our notion of hardness, and that in some useful cases the weaker hardness notion implies our hardness notion. Our motivation for studying the stronger (and perhaps less natural) hardness notion is that the weaker variant is insufficient for the DE-PIR application.

Identifying Hard Permuted Puzzles. We identify natural examples in which a one-time permutation *provably* introduces cryptographic hardness, based on standard assumptions. In these examples, the distributions $\mathcal{D}_0, \mathcal{D}_1$ are efficiently distinguishable, but the permuted puzzle distinguishing problem is computationally hard. We provide such constructions in the random oracle model, and in the plain model under the Decisional Diffie-Hellman (DDH) assumption [DH76]. We additionally observe that the Learning Parity with Noise (LPN) assumption [BKW00,Ale03] itself can be cast as a permuted puzzle. This is described in the following theorem (see Proposition 1, Proposition 3, and Proposition 2 for the formal statements).

Informal Theorem 1 (Hard Permuted Puzzles)**.** *There exists a computationally-hard permuted puzzle distinguishing problem:*

- *In the random oracle model.*
- *If the DDH assumption holds.*
- *If the LPN assumption holds.*

Statistical Query Lower Bound for DE-PIR Toy Problem. We make progress towards better understanding the PermRM assumption underlying the DE-PIR constructions of [BIPW17,CHR17]. Specifically, we show that a toy version of the problem, which was introduced in [BIPW17], provably withstands a rich class of learning algorithms known as *Statistical Query (SQ) algorithms*.

Roughly, the toy problem is to distinguish randomly permuted graphs of random univariate polynomials of relatively low degree from randomly permuted

graphs of random functions. More formally, for a function $f : X \to Y$, we define its 2-dimensional graph $\mathsf{Graph}(f) : X \times Y \to \{0,1\}$ where $\mathsf{Graph}(f)(x,y) = 1 \Leftrightarrow y = f(x)$. For a security parameter λ and a field $\mathbb{F}$, the distributions $\mathcal{D}_0, \mathcal{D}_1$ in the toy problem are over $\{0,1\}^n$ for $n = |\mathbb{F}|^2$, and output a sample $\mathsf{Graph}(\gamma)$ where $\gamma : \mathbb{F} \to \mathbb{F}$ is a uniformly random degree-λ polynomial in $\mathcal{D}_0$, and a uniformly random function in $\mathcal{D}_1$.

We analyze the security of the toy problem against SQ learning algorithms. Our motivation for focusing on learning algorithms in general is that permuted puzzles are a special example of a learning task. Indeed, the adversary's goal is to classify a challenge sample, given many labeled samples. Thus, it is natural to explore approaches from learning theory as potential solvers for (equivalently, attacks on) the permuted puzzle. Roughly speaking, most known learning algorithms can be categorized within two broad categories. The first category leverages linearity, by identifying correlations with subspaces and using algorithms based on Gaussian elimination to identify these. The second category, which is our focus in this work, is SQ algorithms. Informally, an SQ algorithm obtains no labeled samples. Instead, it can make *statistical queries* that are defined by a boolean-valued function f, and the algorithm then obtains the outcome of applying f to a random sample. A statistical query algorithm is an SQ algorithm that makes polynomially many such queries. We show that the toy problem is hard for SQ algorithms (see Theorem 8):

Informal Theorem 2. *The BIPW toy problem is hard for statistical query algorithms.*

We contrast this statistical-query lower bound with the bounded-query statistical indistinguishability lower bound of [CHR17]. That result showed that there is some fixed polynomial B such that no adversary can distinguish B DE-PIR queries from random, even if computationally unbounded. In contrast, our result proves a lower bound for adversaries (also computationally unbounded), that have no a-priori polynomial bound on the number of queries that they can make—in fact, they can make up to $2^{\epsilon\lambda}$ queries where λ is the security parameter and ϵ is a small positive constant. However, they are restricted in that they cannot see the result of any individual query in its entirety; instead, adversaries can only see the result of applying bounded (up to $\epsilon\lambda$-bit) output functions separately to each query.

1.2 Other Instances of Hardness from Random Permutations

There are other instances in which random secret permutations were used to obtain computational hardness. The *Permuted Kernel Problem (PKP)* is an example in the context of a search problem. Roughly, the input in PKP consists of a matrix $A \in \mathbb{Z}_p^{m \times n}$ and a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$, where p is a large prime. A solution is a permutation π on $[n]$ such that the vector $\boldsymbol{v}'$ obtained by applying π to the entries of $\boldsymbol{v}$ is in the kernel of A. PKP is known to be NP-complete in the worst-case [GJ02], and conjectured to be hard on average [Sha89], for sufficiently

large $(n-m)$ and p. It is the underlying assumption in Shamir's identification scheme [Sha89], and has lately seen renewed interest due to its applicability to post-quantum cryptography (e.g., [LP12,FKM+18,KMP19]). Despite being studied for 3 decades, the best known algorithms to date run in exponential time; see [KMP19] and the references therein.

1.3 Techniques

We now proceed to discuss our results and techniques in greater detail.

Defining Permuted Puzzles We generalize and extend the intuitive puzzle framework proposed in [BIPW17], by formally defining the notions of (permuted) puzzle distinguishing problems.

We formalize a *puzzle distinguishing problem* as a pair of distributions $\mathcal{D}_0, \mathcal{D}_1$ over Σ^n, for some alphabet Σ and some input length n. Very roughly, hardness of a puzzle distinguishing problem means one cannot distinguish a single sample from $\mathcal{D}_0$ or $\mathcal{D}_1$, even given oracle access to $\mathcal{D}_0$ *and* $\mathcal{D}_1$. We say that a puzzle problem is (s, ϵ)-*hard* if any size-s adversary distinguishes $\mathcal{D}_0$ from $\mathcal{D}_1$ with advantage at most ϵ. This concrete hardness notion naturally extends to computational hardness of an *ensemble* of puzzles, in which case we allow the distributions to be *keyed* (by both public and secret key information) and require that they be efficiently sampleable given the key.

With this notion of puzzle distinguishing problems, we turn to defining a *permuted puzzle* which, informally, is obtained by sampling a random permutation π once and for all as part of the secret key, and permutating all samples according to π. Hardness of a permuted puzzle is defined identically to hardness of (standard) puzzle distinguishing problems.

We also consider a simpler hardness definition, in which the adversary is given oracle access *only* to a randomly selected $\mathcal{D}_b$ (but not to $\mathcal{D}_{1-b}$), and attempts to guess b. We say that a puzzle distinguishing problem is *weak computationally hard* if every adversary of polynomial size obtains a negligible advantage in this modified distinguishing game. Weak computational hardness captures the security notion considered in [BIPW17], but is too weak for certain applications, as it allows for trivial permuted puzzles, e.g., $\mathcal{D}_0 = \left\{0^{n/2}1^{n/2}\right\}, \mathcal{D}_1 = \left\{1^{n/2}0^{n/2}\right\}$. More generally, and as discussed in Remark 3 (Section 3), weak computational hardness is generally weaker than the definition discussed above (which is more in line with the DE-PIR application). Concretely, we show that the definition discussed above implies the weaker definition, and that in certain cases (e.g., when $\mathcal{D}_1$ is the uniform distribution), the weaker definition implies the stronger one. This last observation will be particularly useful in proving security of our permuted puzzle constructions.

Hard Permuted Puzzle in the Random Oracle (RO) Model Our first permuted puzzle is in the random oracle model. Recall that a permuted puzzle is

defined as the permuted version of a puzzle distinguishing problem. For our RO-based permuted puzzle, the underlying puzzle distinguishing problem is defined as follows. There is no key, but both the sampling algorithm and the adversary have access to the random oracle H. The sampling algorithm samples a uniformly random input x_0 for H, and uniformly random seeds $s_1, \ldots, s_n$, where $n = \lambda$, and computes x_n sequentially as follows. For every $1 \leq i \leq n$, $x_i \stackrel{\mathsf{def}}{=} H(s_i, x_{i-1})$. The sample is then $(x_0, x'_n, s_1, \ldots, s_n)$ where $x'_n \stackrel{\mathsf{def}}{=} x_n$ in $\mathcal{D}_0$, and x'_n is uniformly random in $\mathcal{D}_1$. Notice that in this (unpermuted) puzzle distinguishing problem one can easily distinguish samples from $\mathcal{D}_0$ and $\mathcal{D}_1$, by sequentially applying the oracle to x_0 and the seeds, and checking whether the output is x'_n. This will hold with probability 1 for samples from $\mathcal{D}_0$, and only with negligible probability for samples from $\mathcal{D}_1$ (assuming H has sufficiently long outputs). The corresponding permuted puzzle is obtained by applying a fixed random permutation π^* to the seeds $(s_1, \ldots, s_n)$.[4]

Hardness of the Permuted Puzzle. We focus on a simpler case in which the adversary receives only the challenge sample (and does not request any additional samples from its challenger). This will allow us to present the main ideas of the analysis, and (as we show in Section 4), the argument easily extends to the general case.

At a very high level, we show that the hardness of the permuted puzzle stems from the fact that to successfully guess b, the adversary has to guess the underlying random permutation π^*, *even though it has oracle access to* H.

We first introduce some terminology. For a random oracle H, input x_0 and seeds $s'_1, \ldots, s'_n$, each permutation π over the seeds uniquely defines a corresponding "output" x^π_n through a length-$(n+1)$ "path" P_π defined as follows. Let $x^\pi_0 \stackrel{\mathsf{def}}{=} x_0$, and for every $1 \leq i \leq n$, let $s''_i \stackrel{\mathsf{def}}{=} s'_{\pi^{-1}(i)}$ and $x^\pi_i \stackrel{\mathsf{def}}{=} H\left(s''_i, x^\pi_{i-1}\right)$. Then the label of the i'th node on the path P_π is x^π_i. We say that a node v with label x on some path P_π is *reachable* if x was the oracle answer to one of the adversary's queries in the distinguishing game. We note that when $s'_i = s_{\pi^*(i)}$, i.e., the seeds are permuted with the permutation used in the permuted puzzle, then $x_i^{\pi^*} = x_i$ for every $1 \leq i \leq n$. We call P_{π^*} the *special path*.

We will show that with overwhelming probability, unless the adversary queries H on all the x_i's on the special path (i.e., on $x_0^{\pi^*}, x_1^{\pi^*}, \ldots, x_n^{\pi^*} = x_n$), then he obtains only a negligible advantage in guessing b. Hardness of the permuted puzzle then follows because there are $n!$ possible paths, and the adversary has a negligible chance of guessing the special path (because π^* is a secret random permutation).

[4] We note that syntactically, this is not a permuted puzzle since the permutation should be applied to the *entire* sample. However, this simplified view of the permuted puzzle captures the fact that in our construction, the permutation essentially operates only over the seeds. In the actual construction, this is achieved by tagging the different parts of the sample (with either "input", "output", or "seed") such that any permutation over the entire sample uniquely determines a permutation over the seeds; see Section 4.

We would first like to prove that all node labels, over all paths P_π, are unique. This, however, is clearly false, because the paths are not disjoint: for example, the label of node 0 in all of them is x_0. More generally, if $\pi \neq \pi'$ have the same length-k prefix for some $0 \leq k < \lambda$, then for every $0 \leq i \leq k$, the i'th nodes on $\mathsf{P}_\pi, \mathsf{P}_{\pi'}$ have the same label. In this case, we say that the i'th nodes *correspond to the same node*. Let Unique denote the event that across all paths there do not exist two nodes that (1) do *not* correspond to the same node, but (2) have the same label. Our first observation is that Unique happens with overwhelming probability. Indeed, this holds when H's output is sufficiently large (e.g., of the order of $3\lambda \cdot \log \lambda$), because there are only $\lambda \cdot \lambda!$ different nodes (so the number of pairs is roughly of the order of $2^{2\lambda \cdot \log \lambda}$).

Let $\mathcal{E}$ denote the event that the adversary queries H on the label of an unreachable node, and let $\mathsf{ReachQ} = \bar{\mathcal{E}}$ denote its complement. Our next observation is that conditioned on Unique, ReachQ happens with overwhelming probability. Indeed, conditioned on Unique, the label of an unreachable node is uniformly random, even given the entire adversarial view (including previous oracle answers). Thus, querying H on an unreachable node corresponds to guessing the random node label. When H's output length is sufficiently large (on the order of $3\lambda \cdot \log \lambda$ as discussed above) this happens only with negligible probability.

Consequently, it suffices to analyze the adversarial advantage in the distinguishing game conditioned on $\mathsf{Unique} \wedge \mathsf{ReachQ}$. Notice that in this case, the only *potential* difference between the adversarial views when $b = 0$ and when $b = 1$ is in the label of the endpoint v_{end} of the special path P_{π^*}, which is x'_n when $b = 0$, and independent of x'_n when $b = 1$. Indeed, conditioned on Unique, the label of v_{end} appears nowhere else (i.e., is not the label of any other node on any path). Therefore, conditioned on $\mathsf{ReachQ} \wedge \mathsf{Unique}$, the label of v_{end} appears as one of the oracle answers only if v_{end} is reachable, i.e., only if the adversary queried H on all the node labels on the special path.

Hard Permuted Puzzles in the Plain Model Our second permuted puzzle is based on the Decisional Diffi-Helman (DDH) assumption. The underlying puzzle distinguishing problem is defined over a multiplicative cyclic group G of prime order p with generator g. The public key consists of G, g and a uniformly random vector $\boldsymbol{u} \leftarrow \left(\mathbb{Z}_p^*\right)^n$. A sample from $\mathcal{D}_0, \mathcal{D}_1$ is of the form $(g^{x_1}, \ldots, g^{x_n})$, where in $\mathcal{D}_0$ $(x_1, \ldots, x_n)$ is chosen as a uniformly random vector that is orthogonal to $\boldsymbol{u}$, whereas in $\mathcal{D}_1$ $(x_1, \ldots, x_n)$ is uniformly random. As discussed below, in this (unpermuted) puzzle distinguishing problem one can easily distinguish samples from $\mathcal{D}_0$ and $\mathcal{D}_1$. The corresponding permuted puzzle is obtained by applying a fixed random permutation to the samples $(g^{x_1}, \ldots, g^{x_n})$.

Why are both DDH and a permutation needed? The computational hardness of the permuted puzzles stems from the *combination* of the DDH assumption and the permutation, as we now explain. To see why the DDH assumption is needed, notice that in $\mathcal{D}_0$, all sampled $(x_1, \ldots, x_n)$ belong to an $(n-1)$-dimensional subspace of $\mathbb{Z}_p^n$, whereas in $\mathcal{D}_1$ this happens only with negligible probability,

because each sample is uniformly and independently sampled. Consider a simpler version in which $\mathcal{D}_0, \mathcal{D}_1$ simply output the vector $(x_1, \ldots, x_n)$. In this case, one can obtain an overwhelming distinguishing advantage by (efficiently) checking whether all samples $(x_1, \ldots, x_n)$ lie within an $(n-1)$-dimensional subspace, and if so guess that the underlying distribution is $\mathcal{D}_0$. This "attack" can be executed even if the samples are permuted (as is the case in a permuted puzzle), because applying a permutation to the $(x_1, \ldots, x_n)$ is a linear operation, and therefore preserves the dimension of the subspace. Therefore, a permutation on its own is insufficient to get computational hardness, and we need to rely on the DDH assumption.

To see why the permutation is needed, notice that even if the DDH assumption holds in G, given $(g^{x_1}, \ldots, g^{x_n})$ one can efficiently test whether the underlying exponents $(x_1, \ldots, x_n)$ are orthogonal to a known vector $\boldsymbol{u}$, by only computing exponentiations and multiplications in G. Notice that for a sufficiently large p, the exponents of a sample from $\mathcal{D}_1$ will be orthogonal to $\boldsymbol{u}$ only with negligible probability, so this "attack" succeeds with overwhelming probability.

Hardness of the permuted puzzle. We now show that the *combination* of the DDH assumption, and permuted samples, gives computational hardness. Notice that it suffices to prove that the permuted puzzle is weak computationally hard, because $\mathcal{D}_1$ is random over G^n (see Section 1.3). In this case, the adversarial view $\mathsf{V}_b, b \in \{0,1\}$ consists of the public key $(G, g, \boldsymbol{u})$, and a polynomial number of permuted samples of the form $(g^{x_1}, \ldots, g^{x_n})$ which were all sampled according to $\mathcal{D}_b$ and permuted using the same random permutation π.

Our first observation is that V_b is computationally indistinguishable from the distribution $\mathcal{H}_b$ in which the public key is $(G, g, \pi'(\boldsymbol{u}))$ for $\pi' \stackrel{\mathsf{def}}{=} (\pi)^{-1}$, and the samples from $\mathcal{D}_b$ are *unpermuted*.

Our second observation is that the DDH assumption implies that $\mathcal{H}_b$ is computationally indistinguishable from the distribution $\mathcal{H}'_b$ in which the $(x_1, \ldots, x_n)$ additionally lie in a random 1-dimensional subspace $L_{b,\boldsymbol{v}}$. That is, $(x_1, \ldots, x_n)$ are chosen at random from $L_{b,\boldsymbol{v}}$, where in $\mathcal{H}'_0$ $\boldsymbol{v}$ is random subject to $\boldsymbol{v} \cdot \boldsymbol{u} = 0$, and in $\mathcal{H}'_1$ $\boldsymbol{v}$ is uniformly random. Specifically, we show that the problem of distinguishing between $\mathcal{H}_b, \mathcal{H}'_b$ can be efficiently reduced to the task of distinguishing between a polynomial number of length-$(n-1)$ vectors of the form $(g^{y_1}, \ldots, g^{y_{n-1}})$, where the $(y_1, \ldots, y_{n-1})$ are all sampled from a random 1-dimensional subspace of $\mathbb{Z}_p^{n-1}$ or all sampled from the full space $\mathbb{Z}_p^{n-1}$. If the DDH assumption holds in G then a polynomial-sized adversary cannot efficiently distinguish between these distributions [BHHO08]. Consequently, it suffices to show that $\mathcal{H}'_0, \mathcal{H}'_1$ are computationally close.

The final step is to show that $\mathcal{H}'_0, \mathcal{H}'_1$ are computationally (in fact, statistically) close. The only difference between the two distributions is in the choice of $\boldsymbol{v}$ (which is orthogonal to $\boldsymbol{u}$ in $\mathcal{H}'_0$, and random in $\mathcal{H}'_1$), where all other sampled values are either identical or deterministically determined by the choice of $\boldsymbol{v}$. Notice that in $\mathcal{H}'_1$, $(\pi(\boldsymbol{u}), \boldsymbol{v})$ is uniformly random in $\mathbb{Z}_p^n \times \mathbb{Z}_p^n$. Thus, to show that $\mathcal{H}'_0, \mathcal{H}'_1$ are statistically close and conclude the proof, it suffices to prove that $(\pi(\boldsymbol{u}), \boldsymbol{v})$ in $\mathcal{H}'_0$ is statistically close to uniform over $\mathbb{Z}_p^n \times \mathbb{Z}_p^n$. Very roughly, this

follows from the leftover hash lemma due to the following observations. First, $\pi(\boldsymbol{u})$ has high min entropy even conditioned on $\boldsymbol{u}$ (because π is random). Second, the family of inner product functions with respect to a fixed vector (i.e., $h_{\boldsymbol{v}}(\boldsymbol{v}') = \boldsymbol{v} \cdot \boldsymbol{v}'$) is a pair-wise independent hash function.

Permuted Puzzles and the Learning Parity with Noise (LPN) Assumption. The argument used in the DDH-based permuted puzzle can be generalized to other situations in which it is hard to distinguish between the uniform distribution and a hidden permuted kernel (but easy to distinguish when the kernel is *not* permuted). This more general view allows us to cast the LPN assumption as a permuted puzzle, see Section 5.1.

Statistical-Query Lower Bound We show that SQ algorithms that make polynomially many queries obtain only a negligible advantage in distinguishing the distributions $\mathcal{D}_0, \mathcal{D}_1$ in the toy problem presented in Section 1.1. Recall that a sample in the toy problem is a permuted $\mathsf{Graph}(\gamma)$ where γ is either a uniformly random degree-λ polynomial (in $\mathcal{D}_0$), or a uniformly random function (in $\mathcal{D}_1$), and that the SQ algorithm obtains the outputs of boolean-valued functions f of its choice on random samples. Very roughly, we will show that the outcome of f on (permutation of) a random sample $x \leftarrow \mathcal{D}_b$ is independent of the challenge bit b and the permutation π.

Notice that every permutation π over $\mathsf{Graph}(\gamma)$ defines a partition $\Phi \stackrel{\mathsf{def}}{=} \{\pi(\{i\} \times \mathbb{F})\}_{i \in \mathbb{F}}$ of $\mathbb{F} \times \mathbb{F}$, where each set in the partition corresponds to a single x value. We say that π *respects* the partition Φ. Notice also that each set contains a single non-0 entry (which is $\pi(i, \gamma(i))$, where i is the value of x that corresponds to the set). Thus, an SQ algorithm can compute this partition, so we cannot hope to hide it. Instead, we show indistinguishability even when the adversary is given the partition.

Our main observation is that for every partition Φ, and any boolean-valued function f, there exists $p_{f,\Phi} \in [0,1]$ such that for every $b \in \{0,1\}$, with overwhelming probability over the choice of random permutation π that respects the partition Φ, the expectation $\mathbb{E}_{x \leftarrow \mathcal{D}_b}[f(\pi(x))]$ is very close to $p_{f,\Phi}$, where $\pi(x)$ denote that the entries of x are permuted according to π. Crucially, $p_{f,\Phi}$ is independent of the challenge bit b, any particular sample x, and the permutation (other than the partition).

We prove this observation in two steps. First, we show that in expectation over the choice of the permutation, $\mathbb{E}_{x \leftarrow \mathcal{D}_0}[f(\pi(x))]$ and $\mathbb{E}_{x \leftarrow \mathcal{D}_1}[f(\pi(x))]$ have the same value. To see this, we write the expectations over $x \leftarrow \mathcal{D}_b$ as a weighted sum $\sum_x P_b(x) f(\pi(x))$, and apply linearity of the expectation over π. To show that this is independent of b, we observe that for any fixed x, the distribution of $\pi(x)$ is the same (i.e. does not depend on x).

Next, we show that for any distribution $\mathcal{D}$, the variance (over the choice of the permutation π) of $\mathbb{E}_{x \leftarrow \mathcal{D}_b}[f(\pi(x))]$ is small. The variance is by definition the difference between

$$\mathop{\mathbb{E}}_{\pi}\left[\mathop{\mathbb{E}}_{x \leftarrow \mathcal{D}_b}[f(\pi(x))]^2\right] \tag{1}$$

and

$$\mathop{\mathbb{E}}_{\pi}\left[\mathop{\mathbb{E}}_{x\leftarrow\mathcal{D}_b}\left[f\left(\pi\left(x\right)\right)\right]\right]^2. \tag{2}$$

We show that both Eq. (1) and Eq. (2) can be expressed as an expectation (over some distribution of g, g') of $\mathbb{E}_{\pi}\left[\left(f(\pi(\mathsf{Graph}(g))), f(\pi(\mathsf{Graph}(g')))\right)\right]$. We observe that this depends only on the Hamming distance between g and g'. Finally, we observe that the distribution of (g, g') is uniform in Eq. (2) and two independent samples from $\mathcal{D}_b$ in Eq. (1). To complete the bound on the variance, we show that when g, g' are sampled independently from $\mathcal{D}_b$ (specifically, the interesting case is when they are sampled from $\mathcal{D}_0$), then the distribution of the Hamming distance between g and g' is nearly the same as when g and g' are independent uniformly random functions.

To prove this, we prove a lemma (Lemma 4) stating that when t-wise independent random variables $(X_1, \ldots, X_n)$ satisfy $\Pr[X_i \neq \star_i] = p_i$ for some values of $\star_i$ and p_i such that $\sum_{i\in[n]} p_i \leq \frac{t}{4} \geq \omega(\log\lambda)$, then $(X_1, \ldots, X_n)$ are statistically $\mathrm{negl}(\lambda)$-close to mutually independent. We apply this with X_i being the indicator random variable for the event that $g(i) \neq g'(i)$. This lemma quantitatively strengthens a lemma of [CHR17].

Open Problems and Future Research Directions The broad goal of basing DE-PIR on standard assumptions was a motivating starting point for this work, in which we put forth the framework of permuted puzzles. In describing hard permuted puzzles, we take a "bottom-up" approach by describing such constructions based on standard cryptographic assumptions. Since these permuted puzzles are still not known to imply DE-PIR, we try to close the gap between the permuted puzzle on which DE-PIR security is based, and provably hard permuted puzzles, by taking a "top-down" approach, and analyzing the security of a toy version of the DE-PIR permuted puzzle, against a wide class of possible attacks.

Our work still leaves open a fascinating array of questions, we discuss some of them below. First, it would be very interesting to construct a hard permuted puzzle based only on the existence of one-way functions, as well as to provide "public-key" hard permuted puzzles, namely ones in which the key generation algorithm needs no secret key, based on standard assumptions. In the context of DE-PIR and its related permuted puzzle, it would be interesting to construct DE-PIR based on other (and more standard) assumptions, as well as to analyze the security of its underlying permuted puzzle (and its toy version) against a wider class of attacks.

2 Preliminaries

For a set X, we write $x \leftarrow X$ to denote that x is sampled uniformly at random from X. For a distribution $\mathcal{D}$, we use $\mathrm{Supp}\left(\mathcal{D}\right)$ to denote its support. The min entropy of $\mathcal{D}$ is $\mathsf{H}_{\infty}\left(\mathcal{D}\right) \stackrel{\mathsf{def}}{=} \min_{x\in\mathrm{Supp}(\mathcal{D})} \log\frac{1}{\Pr[x]}$. For a pair X, Y of random

variables, we denote their statistical distance by $d_{\mathsf{TV}}(X, Y)$. We use $\cdot$ to denote inner product, i.e., for a pair $\boldsymbol{x} = (x_1, \ldots, x_n), \boldsymbol{y} = (y_1, \ldots, y_n)$ of vectors, $\boldsymbol{x} \cdot \boldsymbol{y} \stackrel{\mathsf{def}}{=} \sum_{i=1}^{n} x_i y_i$. We use $[n]$ to denote the set $\{1, \ldots, n\}$, and S_n to denote the group of permutations of $[n]$.

Notation 3 (Permutation of a vector)**.** *For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, and a permutation $\pi \in S_n$, we denote:*

$$\pi(\boldsymbol{x}) \stackrel{\mathsf{def}}{=} \left(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(n)}\right).$$

3 Distinguishing Problems and Permuted Puzzles

In this section, we formally define (permuted) puzzle problems which are, roughly, a (special case) of ensembles of keyed "string-distinguishing" problems.

We begin in Section 3.1 by developing terminology for general string-distinguishing and puzzle problems. In Section 3.2 we present the formal distinguishing challenge and define hardness. Then, in Section 3.3, we discuss the case of *permuted* puzzles, and present an alternative indistinguishability notion that is equivalent in certain cases.

3.1 String-Distinguishing Problems

At the core, we consider string-distinguishing problems, defined by a pair of distributions over n-element strings. We begin by defining a finite instance.

Definition 1 (String-Distinguishing Problems). *A* string-distinguishing problem *is a tuple $\Pi = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$, where n is a positive integer, Σ is a non-empty finite set, and each $\mathcal{D}_b$ is a distribution on Σ^n. We call n the* string length*, and Σ the* string alphabet*.*

More generally, an oracle-dependent string-distinguishing problem *is a function $\Pi^{(\cdot)}$ that maps an oracle $O : \{0,1\}^* \to \{0,1\}$ to a string-distinguishing problem Π^O.*

For example, we will consider permuted puzzle string-distinguishing problems relative to a random oracle in Section 4. Note that oracle-dependent string-distinguishing problems are strictly more general than string-distinguishing problems, as the distributions can simply ignore the oracle.

*Remark 1 (**Oracle Outputs**).* In the above, we modeled the oracle as outputting a single bit for simplicity. However, any (deterministic) oracle with multi-bit output can be emulated given a corresponding single-bit-output oracle, at the cost of making more oracle queries.

We will be interested in distinguishing problems where the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ may depend on common sampled "key" information. Parts of this key may be publicly available, or hidden from a distinguishing adversary (discussed in Definition 4); these parts are denoted pk, sk, respectively.

Definition 2 (Keyed Families). *A keyed family* of (oracle-dependent) string-distinguishing problems *is a tuple* $(\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$*, where* $\mathcal{K}$ *is a distribution on a non-empty finite set of pairs* $(\mathsf{pk}, \mathsf{sk})$ *and each* Π_k *is an (oracle-dependent) string-distinguishing problem. We refer to the support of* $\mathcal{K}$ *as the* key space*, and also denote it by* $\mathcal{K}$*.*

Note that any string-distinguishing problem can trivially be viewed as a keyed family by letting $\mathcal{K}$ be a singleton set.

Example 1 (Keyed Family: Dimension-t Subspaces). For a finite field $\mathbb{F}$, and $n \in \mathbb{N}$, consider an example keyed family of string-distinguishing problems $(\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ as follows:

- $\mathcal{K}$ samples a random $t \leftarrow \{1, \ldots, n-1\}$, and a random subspace $L \subseteq \mathbb{F}^n$ of dimension t, sets $\mathsf{pk} = t$ and $\mathsf{sk} = L$, and outputs $(\mathsf{pk}, \mathsf{sk})$.
- For a key $k = (t, L)$, the corresponding string-distinguishing problem is $\Pi_k = (n, \mathbb{F}, \mathcal{D}_0, \mathcal{D}_1)$ where $\mathcal{D}_0$ outputs a uniformly random $\boldsymbol{v} \in L$, and $\mathcal{D}_1$ outputs a uniformly random $\boldsymbol{v} \in \mathbb{F}^n$.

Note that in this example, it will be computationally easy to distinguish between the distributions $\mathcal{D}_0, \mathcal{D}_1$ given sufficiently many samples.

We next define a puzzle problem which, informally, is an *efficiently sampleable* ensemble of keyed families of string-distinguishing problems.

Definition 3 (Puzzle problem). *A* puzzle problem *is an ensemble* $\{(\mathcal{K}_\lambda, \{\Pi_k^{(\cdot)}\}_{k\in\mathcal{K}_\lambda})\}_{\lambda\in\mathbb{Z}^+}$ *of keyed families of (oracle-dependent) string-distinguishing problems associated with probabilistic polynomial-time algorithms* KeyGen *and* Samp *such that:*

- *For any* $\lambda \in \mathbb{Z}^+$*,* $\mathsf{KeyGen}(1^\lambda)$ *outputs a sample from* $\mathcal{K}_\lambda$*.*
- *For any* $k \in \mathcal{K}_\lambda$*, any* $b \in \{0,1\}$*, and any oracle* $O : \{0,1\}^* \to \{0,1\}$*,* $\mathsf{Samp}^O(k, b)$ *outputs a sample from* $\mathcal{D}_b$*, where* $\Pi_k^O = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$*.*

*Remark 2 (**Abbreviated terminology**).* Somewhat abusing notation, we will also refer to a *single* keyed family of string-distinguishing problems as a puzzle problem.

3.2 Distinguishing Games and Hardness

We will focus on puzzle problems where it is computationally hard to distinguish between the pair of distributions. This notion of hardness is formalized through the following distinguishing game. Roughly, the distinguishing adversary is given a challenge sample x from a randomly selected $\mathcal{D}_b$, and query access to both distributions (denoted by choices β below), and must identify from which $\mathcal{D}_b$ the x was sampled.

Definition 4 (Distinguishing Game). *Let $\mathcal{P} = (\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ be a puzzle problem, and let $\mathcal{O}$ be a distribution of oracles. The* distinguishing game $\mathcal{G}^{\mathcal{O}}_{\mathsf{dist}}[\mathcal{P}]$ *is run between an "adversary" $\mathcal{A}$ and a fixed "challenger" $\mathcal{C}$, and is defined as follows:*

1. *$\mathcal{C}$ samples a key $k = (\mathsf{pk}, \mathsf{sk})$ from $\mathcal{K}$, and $O \leftarrow \mathcal{O}$, and denote $\Pi^O_k = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$. $\mathcal{C}$ sends pk to $\mathcal{A}$, who is also given oracle access to O throughout the game.*
2. *$\mathcal{C}$ samples a random bit $b \leftarrow \{0,1\}$, samples $x \leftarrow \mathcal{D}_b$, and sends x to $\mathcal{A}$.*
3. *The following is repeated an arbitrary number of times: $\mathcal{A}$ sends a bit β to $\mathcal{C}$, who samples $x' \leftarrow \mathcal{D}_\beta$ and sends x' to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs a "guess" bit $b' \in \{0,1\}$.*

$\mathcal{A}$ is said to win the game if $b' = b$. $\mathcal{A}$'s advantage *is $\mathsf{Adv}_{\mathcal{A}}(\mathcal{G}^{\mathcal{O}}_{\mathsf{dist}}[\mathcal{P}]) \stackrel{\mathsf{def}}{=} 2\cdot\left|\Pr[b' = b] - \frac{1}{2}\right|$.*

Informally, a permuted puzzle is computationally hard if any polynomial-time adversary wins the distinguishing game of Definition 4 with negligible advantage. We first formalize the notion of *concrete* hardness.

Definition 5 (Concrete Hardness). *A puzzle problem $\mathcal{P} = (\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ is said to be (s, ϵ)-*hard (with respect to oracle distribution $\mathcal{O}$) *if in the game $\mathcal{G}^{\mathcal{O}}_{\mathsf{dist}}[\mathcal{P}]$, all adversaries $\mathcal{A}$ of size at most s have advantage at most ϵ.*

*We say a puzzle problem $\left\{(\mathcal{K}_\lambda, \{\Pi^{(\cdot)}_k\}_{k\in\mathcal{K}_\lambda})\right\}_{\lambda\in\mathbb{Z}^+}$ is $\big(s(\cdot), \epsilon(\cdot)\big)$-*hard (with respect to an ensemble $\{\mathcal{O}_\lambda\}$ of oracle distributions) *if each $(\mathcal{K}_\lambda, \{\Pi^{(\cdot)}_k\}_{k\in\mathcal{K}_\lambda})$ is $\big(s(\lambda), \epsilon(\lambda)\big)$-hard with respect to $\mathcal{O}_\lambda$.*

Definition 6 (Asymptotic Hardness). *As usual, we say simply that $\mathcal{P}$ is* (computationally) hard *if for every $s(\lambda) \leq \lambda^{O(1)}$, there exists $\epsilon(\lambda) \leq \lambda^{-\omega(1)}$ such that for every $\lambda \in \mathbb{Z}^+$, $\mathcal{P}$ is $(s(\cdot), \epsilon(\cdot))$-hard.*

$\mathcal{P}$ is statistically hard *if for some $\epsilon(\lambda) \leq \lambda^{-\omega(1)}$, $\mathcal{P}$ is $(\infty, \epsilon(\cdot))$-hard.*

*Remark 3 (**Discussion on Definition**).*

A slightly simpler and more natural definition would be to give the adversary access to (polynomially-many samples from) *only* a randomly selected $\mathcal{D}_b$, where the adversary must identify b.

For keyed puzzles, these definitions are in general *not* equivalent. Consider, for example, a modified version of Example 1, where both $\mathcal{D}_0$ and $\mathcal{D}_1$ are defined by random dimension-t subspaces, L_0 and L_1. Then over the choice of the key (including L_0, L_1), the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ on their own are *identical*: that is, even an unbounded adversary with arbitrarily many queries would have 0 advantage in the simplified challenge. However, given t samples from *both* distributions, as in Definition 4, $\mathcal{D}_0$ and $\mathcal{D}_1$ are trivially separated, and a sample x can be correctly labeled with noticeable advantage. On the other hand, hardness with respect to our definition implies hardness with respect to the simplified notion, by a hybrid argument over the number of queries (see Lemma 1).

Since our motivation for studying puzzles come from applications where correlated samples from the corresponding distributions can be revealed (e.g., correlated PIR queries on different indices i), we thus maintain the more complex, stronger definition.

The definitional separation in the example above stems from the fact that given access to only one distribution $\mathcal{D}_b$, one cannot necessarily simulate consistent samples from $\mathcal{D}_0$ and $\mathcal{D}_1$. However, in certain instances, this issue does not arise; for example, if one of the two is simply the uniform distribution over strings. We formally address this connection in the following section: presenting the simplified indistinguishability notion in Definition 8, and proving equivalence for certain special cases in Lemma 2.

3.3 Permuted Puzzles and a Related Indistinguishability Notion

In this work we will focus on *permuted* puzzles. This is a special case of puzzle problems, as we now define. Here, the key includes an additional secret random *permutation* on the indices of the n-element strings, and strings output by the distributions $\mathcal{D}_0, \mathcal{D}_1$ will be permuted as dictated by π.

Definition 7 (Permuted Puzzle Problems). *For a puzzle problem* $\mathcal{P} = \{(\mathcal{K}_\lambda, \{\Pi_k^{(\cdot)}\}_{k\in\mathcal{K}_\lambda})\}_{\lambda\in\mathbb{Z}^+}$*, we define the associated* permuted puzzle problem $\mathsf{Perm}(\mathcal{P}) \stackrel{\mathsf{def}}{=} \{(\mathcal{K}'_\lambda, \{\Pi'^{(\cdot)}_{k'}\}_{k'\in\mathcal{K}'_\lambda})\}_{\lambda\in\mathbb{Z}^+}$*, where:*

- *A sample from* $\mathcal{K}'_\lambda$ *is* $\big(\mathsf{pk}, (\mathsf{sk}, \pi)\big)$*, where:*
  - $(\mathsf{pk}, \mathsf{sk})$ *is sampled from* $\mathcal{K}_\lambda$*, and*
  - *If* $\Pi_k = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$*, then* π *is sampled uniformly at random from the symmetric group* S_n*.*
- *For any key* $k' = (\mathsf{pk}, (\mathsf{sk}, \pi))$*, if* $\Pi_{(\mathsf{pk},\mathsf{sk})} = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$ *then* $\Pi'_{k'} = (n, \Sigma, \mathcal{D}'_0, \mathcal{D}'_1)$*, where a sample from* $\mathcal{D}'_b$ *is* $\pi(x)$ *for* $x \leftarrow \mathcal{D}_b$*.*

Recall (Notation 3) for vector $x \in \Sigma^n$ and $\pi \in S_n$, that $\pi(x)$ denotes the index-permuted vector.

As discussed in Remark 3, we now present a simplified notion of indistinguishability, and show that in certain special cases, this definition aligns with Definition 6. In such cases, it will be more convenient to work with the simplified version.

Definition 8 (Weak Hardness of Puzzle Problems). *Let* $\mathcal{P} = (\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ *and* $\mathcal{O}$ *be as in Definition 4. The* simplified distinguishing game $\mathcal{G}^{\mathcal{O}}_{\mathsf{dist},s}[\mathcal{P}]$ *is defined similarly to* $\mathcal{G}^{\mathcal{O}}_{\mathsf{dist}}[\mathcal{P}]$*, except that in Step 3,* $\mathcal{C}$ *samples* $x' \leftarrow \mathcal{D}_b$ *(instead of* $x' \leftarrow \mathcal{D}_\beta$*).*

A puzzle problem $\mathcal{P} = (\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ *is* weak (s, ϵ)-hard *if* $\mathsf{Adv}_{\mathcal{A}}(\mathcal{G}^{\mathcal{O}}_{\mathsf{dist},s}[\mathcal{P}]) \le \epsilon$ *for any size-*s *adversary* $\mathcal{A}$*. Weak computational hardness is defined similarly to Definition 6.*

Note that *weak* computational (statistical) hardness (with respect to Definition 8) is implied by hardness with respect to Definition 4:

Lemma 1 (Standard ⇒ Weak). *Let $\mathcal{P} = \{(\mathcal{K}_\lambda, \{\Pi_k^{(\cdot)}\}_{k\in\mathcal{K}_\lambda})\}_{\lambda\in\mathbb{Z}^+}$ be a puzzle problem. If $\mathcal{P}$ is computationally (statistically, respectively) hard in the standard sense (Definition 6) then it is weak computationally (statistically, respectively) hard (Definition 8).*

The more interesting direction is that weak hardness implies (standard) hardness in the case that one of the two distributions $\mathcal{D}_0$ or $\mathcal{D}_1$ is efficiently sampleable and *permutation-invariant*, in the following sense.

Definition 9 (Permutation-Invariant Distributions). *Let $n \in \mathbb{N}$, let Σ be a non-empty set, and let $\mathcal{D}$ be a distribution over Σ^n. For a permutation $\pi \in S_n$, let $\mathcal{D}_\pi$ be the distribution induced by sampling $x \leftarrow \mathcal{D}$ and outputting $\pi(x)$. We say that $\mathcal{D}$ is* permutation-invariant *if for a uniformly random $\pi \in S_n$, the joint distribution $\mathcal{D}_\pi \times \mathcal{D}_\pi$ is identical to $\mathcal{D} \times \mathcal{D}_\pi$.*

Remark 4. One example of a permutation-invariant distribution $\mathcal{D}$ particularly useful in this work is the uniform distribution over Σ^n.

Lemma 2 (In certain cases Weak ⇒ Standard). *Let $\mathcal{P} = \{(\mathcal{K}_\lambda, \{\Pi_k^{(\cdot)}\}_{k\in\mathcal{K}_\lambda})\}_{\lambda\in\mathbb{Z}^+}$ be a puzzle problem. If:*

- *The corresponding permuted puzzle $\mathsf{Perm}(\mathcal{P})$ is* weak *computationally hard (Definition 8).*
- *For every λ, every $k = (\mathsf{pk}, \mathsf{sk}) \in \mathrm{Supp}(\mathcal{K}_\lambda)$, and every $\Pi_k = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$:*
  - *$\mathcal{D}_1$ is permutation-invariant.*
  - *One can efficiently sample from $\mathcal{D}_1$ without sk.*

Then $\mathsf{Perm}(\mathcal{P})$ is computationally hard in the standard sense (Definition 6).

Finally, we show that the existence of hard permuted puzzles for which the original distributions $\mathcal{D}_0, \mathcal{D}_1$ are *statistically far* implies the existence of one-way functions. Note that this holds with respect to our standard (strong) definition of computational hardness, but *not* in general for the weaker notion (where, for example, even trivially distinguishable singleton distributions D_0 over $(0,1)$ and D_1 over $(1,0)$ become statistically identical when receiving samples *only* from permuted-D_0 or permuted-D_1).

Lemma 3. *If $\mathcal{P}$ is a puzzle problem that is not statistically hard, but $\mathsf{Perm}(\mathcal{P})$ is computationally hard, then there exists a one-way function.*

The proofs of Lemmas 1, 2 and 3 are deferred to the full version.

4 Hard Permuted Puzzles in the Random Oracle Model

We show that there exist computationally hard permuted puzzles in the random oracle model. We first formally define the notion of a random oracle.

Definition 10 (Random Oracle). *We use the term* random oracle *to refer to the uniform distribution on functions mapping $\{0,1\}^* \to \{0,1\}$.*

Construction 4 (Permuted puzzles in the ROM)**.** *Let H be a random oracle. For a security parameter λ, we interpret H as a function $H_\lambda : \{0,1\}^{m_\lambda+\lambda} \to \{0,1\}^{m_\lambda}$ for $m_\lambda = 2(\lambda+1)\log\lambda$ (also see Remark 1). We define a puzzle problem $\mathcal{P} = \left\{(\mathcal{K}_\lambda, \{\Pi_k\}_{k\in\mathcal{K}_\lambda})\right\}$ by the following* KeyGen *and* Samp *algorithms:*

- $\mathsf{KeyGen}\left(1^\lambda\right)$ *outputs 1^λ as the public key (the secret key is empty).*[5]
  We note that for any λ, the corresponding string distinguishing problem $\Pi_\lambda = \left(n, \Sigma, \mathcal{D}_0^{(\cdot)}, \mathcal{D}_1^{(\cdot)}\right)$ has $n = \lambda + 2$ and $\Sigma = \{0,1\}^{m_\lambda} \times \{\texttt{INPUT}, \texttt{OUTPUT}, \texttt{SEED}\}$.
- $\mathsf{Samp}(k, b)$ *where $k = 1^\lambda$ outputs a sample from $\mathcal{D}_{\lambda,b}^{H_\lambda}$ for $H_\lambda : \{0,1\}^{m_\lambda+\lambda} \to \{0,1\}^{m_\lambda}$ as defined above, where $\mathcal{D}_{\lambda,b}^{H_\lambda}$ is defined as follows.*
  - *A sample from $\mathcal{D}_{\lambda,0}^{H_\lambda}$ is of the form $(\sigma_1, \ldots, \sigma_{\lambda+2})$, where:*
    - *For $i \in [\lambda]$, $\sigma_i = (s_i, \texttt{SEED})$ for uniformly random and independent $s_1, \ldots, s_\lambda$ in $\{0,1\}^{m_\lambda}$.*
    - *$\sigma_{\lambda+1} = (x_0, \texttt{INPUT})$, where x_0 is uniformly random in $\{0,1\}^{m_\lambda}$.*
    - *$\sigma_{\lambda+2} = (x_\lambda, \texttt{OUTPUT})$, where for each $i \in [\lambda]$, $x_i = H_\lambda(s_i', x_{i-1})$, where s_i' is the length-λ prefix of s_i. (That is, the random oracle uses length-λ seeds, and the rest of the bits in the seed are ignored.)*
  - *$\mathcal{D}_{\lambda,1}^{H_\lambda}$ is defined identically to $\mathcal{D}_{\lambda,0}^{H_\lambda}$, except that x_λ is uniformly random in $\{0,1\}^{m_\lambda}$, independent of x_0, H_λ, and s_1, …, s_λ.*

Proposition 1. *The puzzle problem $\mathcal{P}$ of Construction 4 is computationally easy, and the corresponding permuted puzzle problem* $\mathsf{Perm}(\mathcal{P})$ *is statistically hard, with respect to a random oracle.*

We note that $\mathcal{P}$ is computationally easy in an extremely strong sense: a polynomial-sized adversary can obtain advantage $1 - \mathrm{negl}(\lambda)$ in the distinguishing game. The proof of is deferred to the full version.

5 Hard Permuted Puzzles in the Plain Model

In this section we discuss permuted puzzle problems based on hidden permuted kernels. At a high level, these puzzles have the following structure. First, the distributions $\mathcal{D}_0, \mathcal{D}_1$ are associated with a group G with generator g, and a uniformly random public "constraint vector" $\boldsymbol{c}$. Samples from $\mathcal{D}_0$ and $\mathcal{D}_1$ are vectors in G^m, of the form $g^{\boldsymbol{x}}$. Specifically, $\mathcal{D}_1$ samples a uniformly random vector in G^m, whereas $\mathcal{D}_0$ samples a vector $\boldsymbol{x}$ that is uniformly random subject to being orthogonal to $\boldsymbol{c}$. Intuitively, since $\mathcal{D}_1$ is uniformly random, weak computational hardness of the permuted puzzle problem implies computational hardness by Lemma 2.

*Remark 5 (**An alternative formulation of the problem**).* In the high-level blueprint of a permuted puzzle problem described above, the constraint vector $\boldsymbol{c}$

[5] We note that in this permuted puzzle construction the key generation stage is obsolete.

is given "in the clear" (namely, we assume it is public, and indistinguishability does not rely on the secrecy of $\boldsymbol{c}$), and the samples $\boldsymbol{x}$ are permuted according to a random permutation $\pi \in S_n$, namely, the adversary obtains $\pi(\boldsymbol{x})$ (recall that $\pi(\boldsymbol{x}) = \left(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(n)}\right)$ Let $\mathcal{C}$ denote the set of "good" vectors $\boldsymbol{c}$, i.e., vectors that satisfy the requirement, and let G^n denote the domain over which $\mathcal{D}_0, \mathcal{D}_1$ are defined. Let $\mathcal{D}'_b \stackrel{\text{def}}{=} \left(\boldsymbol{c}, (\pi(\boldsymbol{x}_i))_{i\in[q]}\right)_{\boldsymbol{c}\leftarrow\mathcal{C},\pi\leftarrow S_n,\boldsymbol{x}_i\leftarrow\mathcal{D}_b}$ denote the distribution over the adversary's view in the simplified distinguishing game of Definition 8, where b is the challenge bit, and q is the number of samples the adversary receives from the challenger. Denote $\mathcal{D}''_b \stackrel{\text{def}}{=} \left(\pi(\boldsymbol{c}), (\boldsymbol{x}_i)_{i\in[q]}\right)_{\boldsymbol{c}\leftarrow\mathcal{C},\pi\leftarrow S_n,\boldsymbol{x}_i\leftarrow\mathcal{D}_b}$. The permuted puzzle problems described in this section will have the property that $\mathcal{D}'_b \approx \mathcal{D}''_b$ for $b \in \{0,1\}$, which will be used in the security proofs.

5.1 Permuted Puzzles and the Learning Parity With Noise (LPN) Assumption

We now describe how to cast the Learning Parity with Noise (LPN) assumption as a permuted puzzle.

Notation. For $\boldsymbol{a} \in \mathbb{F}_2^n$, we use $|\boldsymbol{a}|$ to denote the Hamming weight of $\boldsymbol{a}$. For $i \in [n]$, we denote $\mathbf{v}_{n,i} = 1^i \cdot 0^{n-i}$ (i.e., a canonical length-n vector of Hamming weight i). For $n \in \mathbb{N}$, let $\mathcal{R}_n$ denote the distribution that outputs a uniformly random $\boldsymbol{x} \leftarrow \mathbb{F}_2^n$. For a fixed $\boldsymbol{s} \in \mathbb{F}_2^n$, and $\gamma \in (0,1)$, let $\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}$ denote the distribution over $\mathbb{F}_2^n$ that with probability γ outputs a uniformly random $\boldsymbol{x} \leftarrow \mathbb{F}_2^n$, and otherwise (with probability $1-\gamma$) outputs a uniformly random element of the set $\{\boldsymbol{x} \in \mathbb{F}_2^n \ : \ \boldsymbol{x} \cdot \boldsymbol{s} = 0\}$.

Definition 11 (Learning Parity with Noise (LPN)). *Let $\gamma \in (0,1)$. The γ-*Learning Parity with Noise (γ-LPN)* *assumption conjectures that for every polynomial-sized oracle circuit ensemble $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ there exists a negligible function $\epsilon(\lambda)$ such that for every λ,*

$$\mathsf{Adv}^{\mathsf{LPN}}_{\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \left| \Pr_{\boldsymbol{s}\leftarrow\mathbb{F}_2^\lambda}\left[\mathcal{A}^{\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{R}_\lambda}(1^\lambda) = 1\right] \right| \leq \epsilon(\lambda).$$

*Remark 6 (**Equivalence to standard LPN formulation**).* Recall that the standard γ-LPN assumption, for $0 < \gamma < \frac{1}{2}$, states that any polynomial-time adversary obtains only a negligible advantage in distinguishing between (polynomially many samples from) the following distributions:

- $(\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i)_{i=1}^m$, where for every i, $\boldsymbol{a}_i \leftarrow \mathbb{F}_2^n$ and e_i is sampled from a Bernoulli distribution with $\Pr[e_i = 1] = \gamma$; vs.
- $(\boldsymbol{a}_i, u_i)_{i=1}^m$, where each $(\boldsymbol{a}_i, u_i)$ is sampled uniformly at random from $\mathbb{F}_2^{n+1}$.

We now show that if the standard LPN assumption holds with parameters $(\lambda - 1, \gamma/2)$, then Definition 11 holds with parameters (λ, γ), where the distinguishing advantage increases by at most $2^{-\lambda}$.

- In Definition 11 if $\boldsymbol{s} = \mathbf{0}$ then $\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}$ and $\mathcal{R}_\lambda$ are identically distributed, whereas in the standard LPN formulation they might be distinguishable (with some advantage ≤ 1).
- Conditioned on $\boldsymbol{s} \neq \mathbf{0}$ in Definition 11, there exists at least one nonzero coordinate $i \in [\lambda]$ such that $s_i = 1$, in which case the i'th coordinate of a sample from $\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}$ is a noisy linear function of the other coordinates. That is, with probability $(1-\gamma)+\frac{\gamma}{2}$, it holds that $\boldsymbol{x}$ is random subject to $x_i = \sum_{j \neq i} x_j s_j$, and with probability $\frac{\gamma}{2}$, the vector $\boldsymbol{x}$ is random subject to $x_i = \sum_{j \neq i} x_j s_j + 1$ with offset noise. Moreover, since $\boldsymbol{s}$ is uniformly random over non-zero vectors, such coordinate is equally likely to occur for any index $i \in [\lambda]$ (in contrast, in the standard LPN formulation the last coordinate always necessary satisfies this "special" structure; i.e., equivalent to $\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}$ with secret $\boldsymbol{s} = (\boldsymbol{s}', 1)$).

Thus, conditioned on the (overwhelming probability) event that $\boldsymbol{s} \neq \mathbf{0}$, we can reduce the problem of distinguishing standard LPN with parameters $(\lambda - 1, \gamma/2)$, to distinguishing our version parameters (λ, γ), by selecting a random $i \leftarrow [\lambda]$ and transposing the i'th coordinate of all received LPN samples with the final coordinate.

We now describe how to cast LPN as a permuted puzzle.

Construction 5 (Permuted puzzle problem from LPN)**.** *For a noise parameter $\gamma \in (0, 1/2)$, we define a puzzle problem $\mathcal{P} = \left\{(\mathcal{K}_\lambda, \{\Pi_k\}_{k \in \mathcal{K}_\lambda})\right\}$ by the following* KeyGen *and* Samp *algorithms:*

- $\mathsf{KeyGen}\left(1^\lambda\right)$ *samples a weight w according to the binomial distribution over $[n]$. It outputs w as the secret key (there is no public key).*
  For a key k generated by $\mathsf{KeyGen}\left(1^\lambda\right)$, the corresponding string-distinguishing problem $\Pi_k = \left(n, \Sigma, \mathcal{D}_0, \mathcal{D}_1\right)$ has string length $n = \lambda$ and alphabet $\Sigma = \mathbb{F}_2$.
- $\mathsf{Samp}\left(\mathsf{w}, b\right)$ *outputs a sample from $\mathcal{D}_{\lambda,b}$, where $\mathcal{D}_{\lambda,0} = \mathcal{D}_{\mathit{LPN},\mathbf{v}_{\lambda,\mathsf{w}},\gamma}$, and $\mathcal{D}_{\lambda,1} = \mathcal{R}_\lambda$.*

Proposition 2. *For any constant $\gamma \in (0, 1/2)$, the γ-LPN assumption is equivalent to the computational hardness of the permuted puzzle problem* $\mathsf{Perm}\left(\mathcal{P}_\gamma\right)$ *of Construction 5.*

Proof. Regarding the equivalence of the γ-LPN assumption and the computational hardness of $\mathsf{Perm}\left(\mathcal{P}_\gamma\right)$, notice that the permuted distribution $\mathcal{D}'_{\lambda,0}$ of the permuted puzzle is exactly $\mathcal{D}_{\mathsf{LPN},\boldsymbol{s},\gamma}$, where $\boldsymbol{s} = \pi\left(\mathbf{v}_{\lambda,\mathsf{w}}\right)$ for a uniformly random $\pi \in S_\lambda$, and a weight $\mathsf{w} \in [\lambda]$ which was sampled according to the binomial distribution, so $\boldsymbol{s}$ is uniformly random in $\mathbb{F}_2^n$. Therefore, the distinguishing advantage in the distinguishing game of the permuted puzzle corresponds exactly to the γ-LPN assumption (because additionally $\mathcal{D}'_{\lambda,1} = \mathcal{R}_\lambda$). □

*Remark 7 (**(Unpermuted) puzzle problem is computationally easy**).* We note that the (unpermuted) puzzle problem of Construction 5 is computationally

easy. Indeed, in the unpermuted puzzle problem there are only λ possible "secret" vectors (i.e., $\mathbf{v}_{\lambda,1},\ldots,\mathbf{v}_{\lambda,\lambda}$). Given a polynomial number of samples from $\mathcal{D}_{\lambda,0}$ the adversary can determine, with overwhelming probability, which of these is the secret vector used in $\mathcal{D}_{\lambda,0}$, and can then determine (with constant advantage) whether the challenge sample is from $\mathcal{D}_{\lambda,0}$ or $\mathcal{D}_{\lambda,1}$.

5.2 Permuted Puzzles Based on DDH

In this section we describe a permuted puzzle problem based on the DDH assumption. We first recall the standard DDH assumption, and describe an equivalent formulation which we use.

Definition 12 (Group Samplers). *A* group sampler *is a probabilistic polynomial-time algorithm $\mathcal{G}$ that on input 1^λ outputs a pair (G,g), where G is a multiplicative cyclic group of order $p=\Theta(2^\lambda)$, and g is a generator of G. We assume that p is included in the group description G, and that there exists an efficient algorithm that given G and descriptions of group elements g_1,g_2 outputs a description of $g_1\cdot g_2$.*

Definition 13 (DDH assumption). *For any cyclic group G of order p with generator g, define the following distributions:*

- $\mathcal{D}_{DDH}(G,g)$ *is uniform over the set* $\left\{(g^x,g^y,g^{xy}):x,y\in\mathbb{Z}_p\right\}$.
- $\mathcal{R}_{DDH}(G,g)$ *is uniform over* G^3.

For a group sampler $\mathcal{G}$, the DDH assumption over $\mathcal{G}$ *conjectures that for any polynomial-sized circuit family $\mathcal{A}=\{\mathcal{A}_\lambda\}_\lambda$ there exists a negligible function $\epsilon(\lambda)$ such that for every λ:*

$$\mathsf{Adv}_{\mathcal{A}}^{DDH(\mathcal{G})}(\lambda)\stackrel{\text{def}}{=}\left|\Pr_{\substack{(G,g)\leftarrow\mathcal{G}(1^\lambda)\\ v\leftarrow\mathcal{D}_{DDH}(G,g)}}\left[\mathcal{A}_\lambda(v)=1\right]-\Pr_{\substack{(G,g)\leftarrow\mathcal{G}(1^\lambda)\\ v\leftarrow\mathcal{R}_{DDH}(G,g)}}\left[\mathcal{A}_\lambda(v)=1\right]\right|\leq\epsilon(\lambda).$$

We will use the matrix version of DDH, defined next. Informally, in matrix DDH the adversary is given many vectors of the form $(g^{x_1},\ldots,g^{x_n})$, and the conjecture is that no polynomial-time adversary can distinguish between the case that the $(x_1,\ldots x_n)$ are sampled uniformly from $\mathbb{Z}_p^n$, and the case that $(x_1,\ldots,x_n)$ are sampled from a random 1-dimensional subspace of $\mathbb{Z}_p^n$.

Definition 14 (Matrix DDH assumption). *For a cyclic group G of order p, and $n,q\in\mathbb{N}$, define*

$$\mathsf{Rk}_i\left(G^{q\times n}\right)=\left\{g^A=\left(g^{a_{ij}}\right)_{i\in[q],j\in[n]}\ :\ A\in\mathbb{Z}_p^{q\times n},\mathsf{rank}(A)=i\right\}.$$

Let $\mathcal{G}$ be as in Definition 13, and let $n=n(\lambda),q=q(\lambda)$ be polynomials such that $q(\lambda)\geq n(\lambda)$ for every λ. The matrix DDH assumption over $\mathcal{G}$ *conjectures*

that for any polynomial-sized circuit family $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ there exists a negligible function $\epsilon(\lambda)$ such that for every λ:

$$\mathsf{Adv}^{\mathsf{M\text{-}DDH}(\mathcal{G})}_{\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr_{\substack{(G,g)\leftarrow\mathcal{G}(1^\lambda)\\ v\leftarrow\mathsf{Rk}_n\left(G^{q\times n}\right)}}[\mathcal{A}_\lambda(v)=1] - \Pr_{\substack{(G,g)\leftarrow\mathcal{G}(1^\lambda)\\ v\leftarrow\mathsf{Rk}_1\left(G^{q\times n}\right)}}[\mathcal{A}_\lambda(v)=1] \right| \leq \epsilon(\lambda).$$

Boneh et al. proved [BHHO08, Lemma 1] that the DDH assumption over $\mathcal{G}$ implies the matrix DDH assumption over $\mathcal{G}$:

Imported Theorem 6 (DDH implies matrix-DDH [BHHO08])**.** *Let λ be a security parameter, let $\mathcal{G}$ be as in Definition 13, and let $n = n(\lambda), q = q(\lambda)$ be polynomials. Then for any polynomial-sized adversary circuit $\mathcal{A}_{\mathsf{M\text{-}DDH}}$ there exists an adversary $\mathcal{A}_{\mathsf{DDH}}$ of size $|\mathcal{A}_{\mathsf{M\text{-}DDH}}| + \mathrm{poly}(q,n)$ such that $\mathsf{Adv}^{\mathsf{M\text{-}DDH}(\mathcal{G})}_{\mathcal{A}_{\mathsf{M\text{-}DDH}}}(\lambda) \leq (n-1)\cdot\mathsf{Adv}^{\mathsf{DDH}(\mathcal{G})}_{\mathcal{A}_{\mathsf{DDH}}}(\lambda)$.*

We are now ready to define the permuted puzzle problem based on DDH.

Construction 7 (Permuted puzzle problem from DDH)**.** *Let $\mathcal{G}$ be as in Definition 13. We define a puzzle problem $\mathcal{P} = \left\{(\mathcal{K}_\lambda, \{\Pi_k\}_{k\in\mathcal{K}_\lambda})\right\}$ by the following* KeyGen *and* Samp *algorithms:*

- KeyGen *on input 1^λ samples $(G,g)\leftarrow\mathcal{G}(1^\lambda)$, where $\mathcal{G}$ is the group sampling algorithm of Definition 13. Let p denote the order of G. Then,* KeyGen *samples a uniformly random vector $\boldsymbol{u}\in\mathbb{Z}_p^n$ for $n=\lambda^2$ and outputs $(G,g,\boldsymbol{u})$ as a public key (there is no secret key).*
  We note that for any $k=(G,g,\boldsymbol{u})$, the corresponding string distinguishing problem $\Pi_k = (n,\Sigma,\mathcal{D}_0,\mathcal{D}_1)$ has alphabet $\Sigma = G$.
- $\mathsf{Samp}(k,b)$ *for $k=(n,\Sigma,\mathcal{D}_0,\mathcal{D}_1)$ outputs a sample from $\mathcal{D}_b$, where:*
  - *$\mathcal{D}_0$ is uniform over $\{g^{\boldsymbol{x}}\in G^n : \boldsymbol{x}\cdot\boldsymbol{u}=0\}$.*
  - *$\mathcal{D}_1$ is uniform over G^n.*

Proposition 3. *The puzzle problem $\mathcal{P}$ of Construction 7 is computationally easy. Moreover, if $\mathcal{G}$ is an ensemble of groups in which the matrix DDH assumption of Definition 14 holds, then the corresponding permuted puzzle problem $\mathsf{Perm}(\mathcal{P})$ is computationally hard.*

We note that $\mathcal{P}$ is computationally easy in an extremely strong sense: a polynomial-sized adversary can obtain advantage $1-\mathrm{negl}(\lambda)$ in the distinguishing game. The proof of Proposition 3 is deferred to the full version.

6 Statistical Query Lower Bound

In this section we discuss a specific permuted puzzle toy problem introduced by [BIPW17], and study its hardness against a large class of potential adversarial algorithms called *statistical-query algorithms*. We first define this class of algorithms in Section 6.1, then present the toy problem in Section 6.2 and prove it is secure against such algorithms.

6.1 Statistical Query Algorithms

Definition 15 (Statistical Query Algorithms). *Let $\mathcal{P} = (\mathcal{K}, \{\Pi_k\}_{k\in\mathcal{K}})$ be a puzzle problem. A* statistical q-query algorithm *for $\mathcal{G}_{\mathsf{dist},s}[\mathcal{P}]$ is a stateful adversary $\mathcal{A}$ using an "inner adversary" $\mathcal{A}_{\mathsf{SQ}}$ as follows.*

1. *Upon receiving the public key pk, $\mathcal{A}$ forwards it to $\mathcal{A}_{\mathsf{SQ}}$.*
   Recall that pk is part of the key k, and denote $\Pi_k = (n, \Sigma, \mathcal{D}_0, \mathcal{D}_1)$.
2. *The following is repeated q times:*
   (a) *$\mathcal{A}_{\mathsf{SQ}}$ outputs a boolean-valued function f.*[6]
   (b) *$\mathcal{A}$ requests a sample $x \leftarrow \mathcal{D}_b$ from the challenger (where $b \in \{0,1\}$ is the challenger's secret bit), computes $f(x)$ (this is a single bit), and forwards $f(x)$ to $\mathcal{A}_{\mathsf{SQ}}$.*
3. *When $\mathcal{A}_{\mathsf{SQ}}$ outputs a "guess" bit b', $\mathcal{A}$ forwards b' to the challenger.*

Remark 8. We consider only statistical query algorithms for the simplified distinguishing game $\mathcal{G}_{\mathsf{dist},s}$ of Definition 8 because our lower bounds (proven in Section 6.2) hold for puzzle problems in which weak computational hardness (i.e., hardness of $\mathcal{G}_{\mathsf{dist},s}$) is equivalent to computational hardness (i.e., hardness of the more standard distinguishing game $\mathcal{G}_{\mathsf{dist}}$ of Definition 4) by Lemma 2.

Statistical Query (SQ) algorithms constitute a broad class of distinguishing algorithms, that is incomparable in power to polynomial-time algorithms. For example, an SQ algorithm can distinguish between a PRG output and a uniformly random string with a single query. On the other hand, SQ algorithms cannot distinguish between a distribution that is uniform on $\{0,1\}^n$ and one that is uniform on a random high-dimensional subspace of $\{0,1\}^n$. These distributions can be distinguished (given many samples) in polynomial time by a simple rank computation.

Still, in the context of distinguishing problems, SQ algorithms seem to be a powerful class of adversarial algorithms. In fact, except for the aforementioned examples of algorithms which exploit algebraic structure, we are not aware of any natural distinguishing algorithms that cannot be simulated by statistical query algorithms. A challenging and important open problem, which we leave for future work, is to formalize a class of algorithms that use algebraic structure (or even only linear algebra), possibly together with statistical queries, and to prove lower bounds against this class.

6.2 The Toy Problem and Lower Bound

The works [CHR17,BIPW17] base the security of their DE-PIR schemes on the PermRM conjecture, for which they also discuss different variants (e.g., noisy versions). Boyle et al. [BIPW17] also put forth a toy version of the problem, for which we will prove a lower bound against SQ algorithms. We first recall the PermRM conjecture and its toy version.

[6] We do not assume any bound on the description size or complexity of f, which will not matter for our lower bounds.

Conjecture 1 (PermRM*, Conjecture 4.2 in [BIPW17]).* Let $m \in \mathbb{N}$ be a dimension parameter, let $\lambda \in \mathbb{N}$ be a security parameter, let $d = d_m(n)$ be the minimal integer such that $n \geq \binom{m+d}{d}$, and let $\mathbb{F}$ be a finite field satisfying $|\mathbb{F}| > d\lambda + 1$. Define a probabilistic algorithm $\mathsf{Samp}(b, \pi, v)$ that operates as follows:

- If $b = 0$:
  1. Select m random degree-λ polynomial $p_1, \ldots, p_m \leftarrow \mathbb{F}[X]$ such that for every $1 \leq i \leq \lambda$, $p_i(0) = v$. Notice that these polynomials determine a curve $\gamma(t)$ in $\mathbb{F}^m$, given by $\{(p_1(t), \ldots, p_m(t)) \ : \ t \in \mathbb{F}\}$.
  2. Sample $d\lambda+1$ distinct points on the curve $\gamma(t)$, determined by non-zero parameters $t_0, \ldots, t_{d\lambda} \leftarrow \mathbb{F}$.
  3. Output the points, in order, where each point is permuted according to $\pi : \mathbb{F}^m \to \mathbb{F}^m$, namely output
$$\left(\pi\left(p_1(t_i), \ldots, p_m(t_i)\right)\right)_{i=0}^{d\lambda} \in \left(\mathbb{F}^m\right)^{d\lambda+1}.$$
- If $b = 1$: sample $d\lambda+1$ random points in $\mathbb{F}^m$ $(w_0, \ldots, w_{d\lambda}) \leftarrow \left(\mathbb{F}^m\right)^{d\lambda+1}$, and output $(w_0, \ldots, w_{d\lambda})$.

The PermRM conjecture is that for every efficient non-uniform $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mu(\lambda) = \mathrm{negl}(\lambda)$ such that:

$$\Pr\left[\begin{array}{c} \left(1^n, 1^{|\mathbb{F}|}, \mathsf{aux}\right) \leftarrow \mathcal{A}_1\left(1^\lambda\right) \\ \pi \leftarrow S_{(\mathbb{F}^m)}; b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}_2^{\mathsf{Samp}(b,\pi,\cdot)}\left(1^n, \mathsf{aux}\right) \end{array} \ : \ b' = b \right] \leq 1/2 + \mu(\lambda)$$

Let $\mathbb{F} = \{\mathbb{F}_\lambda\}_{\lambda \in \mathbb{Z}^+}$ denote an ensemble of finite fields with $|\mathbb{F}_\lambda| = \Theta(\lambda^2)$. Let $q = q_\lambda$ denote $|\mathbb{F}_\lambda|$.

For a function $f : X \to Y$, we define $\mathsf{Graph}(f) : X \times Y \to \{0,1\}$ such that

$$\mathsf{Graph}(f)(x,y) = \begin{cases} 1 & \text{if } y = f(x) \\ 0 & \text{otherwise.} \end{cases}$$

Define the puzzle problem $\Pi_\lambda = (n, \{0,1\}, \mathcal{D}_0, \mathcal{D}_1)$, where $n = q^2$, and $\mathcal{D}_0$ and $\mathcal{D}_1$ are defined as follows.

- A sample from $\mathcal{D}_0$ is $\mathsf{Graph}(\gamma)$, where $\gamma : \mathbb{F} \to \mathbb{F}$ is a uniformly random degree-λ polynomial.
- A sample from $\mathcal{D}_1$ is $\mathsf{Graph}(U)$, where $U : \mathbb{F} \to \mathbb{F}$ is a uniformly random function.

Conjecture 2 ([BIPW17]). The permuted puzzle problem $\mathcal{P} \stackrel{\mathsf{def}}{=} \mathsf{Perm}(\{\Pi_\lambda\}_{\lambda \in \mathbb{Z}^+})$ is computationally hard.

Theorem 8. *The simplified distinguishing game $\mathcal{G}_{\mathsf{dist},s}[\mathcal{P}]$ is hard for statistical-query algorithms. That is, for all polynomially bounded $q(\cdot)$, the advantage of any statistical $q(\lambda)$-query adversary in $\mathcal{G}_{\mathsf{dist},s}[\mathcal{P}]$ is at most $e^{-\Omega(\lambda)}$.*

Proof. We will show that even if we give the statistical query adversary additional information about π, it cannot distinguish permuted samples from $\mathcal{D}_0$ from permuted samples from $\mathcal{D}_1$. Specifically, we will give the adversary (for free) the unordered partition $\Phi_1 \cup \cdots \cup \Phi_q$ of $\mathbb{F} \times \mathbb{F}$, where $\Phi_i = \pi(\{i\} \times \mathbb{F})$. (Intuitively, Φ_i is the image under π of all points in which the X coordinate equals i. In particular, $\pi(\mathsf{Graph}(f))$ takes value "1" at exactly one coordinate in Φ_i.) Note that it is indeed possible for a statistical query adversary to learn $\Phi \stackrel{\mathsf{def}}{=} \{\Phi_1, \ldots, \Phi_q\}$: if (x, y) and (x', y') belong to the same Φ_i, then for a random sample $z \leftarrow \mathcal{D}_b$, it is never the case that $\pi(z)_{(x,y)} = \pi(z)_{(x',y')} = 1$. However, if (x, y) and (x', y') do *not* belong to the same Φ_i, then $\pi(z)_{(x,y)} = \pi(z)_{(x',y')} = 1$ with probability at least $\frac{1}{q^2}$.

We say that a permutation π $\mathsf{respects}$ a partition $\Phi = \{\Phi_1, \ldots, \Phi_q\}$ if $\{\pi(\{i\} \times \mathbb{F})\}_i = \Phi$. For any partition Φ, we will write $\Pr_\Phi$ to denote the probability space in which a permutation π is sampled uniformly at random from the set of permutations that respect Φ. Similarly, we will write $\mathbb{E}_\Phi$ to denote expectations in $\Pr_\Phi$, and we write Var_Φ to denote variances in $\Pr_\Phi$.

We will show that there is some negligible function $\nu : \mathbb{Z}^+ \to \mathbb{R}$ such that for any function $f : \{0,1\}^n \to \{0,1\}$ and any partition Φ, there exists some $p_{f,\Phi} \in [0,1]$ such that for every $b \in \{0,1\}$, it holds that

$$\Pr_{\Phi}\left[\mathop{\mathbb{E}}_{x \leftarrow \mathcal{D}_b}\left[f(\pi(x))\right] - p_{f,\Phi}\right| \geq \nu(\lambda)\Bigg] \leq \nu(\lambda).$$

Crucially, $p_{f,\Phi}$ is independent of the challenge bit b, the specific sample x, and the secret permutation π (except for its dependence on Φ). Thus, the answer to a query f can be simulated by computing $p_{f,\Phi}$.

The following two observations are at the core of our proof. Recall that Δ denotes the Hamming distance. For a pair of functions $g, g' : X \to Y$, we denote $\Delta(g, g') = |\{x \in X \ : \ g(x) \neq g'(x)\}|$.

Claim 1. For any partition Φ, any function $g : \mathbb{F} \to \mathbb{F}$, and any fixed permutation π^* that respects Φ, the distribution of $\pi(\mathsf{Graph}(g))$ under $\Pr_\Phi$ is identical to the distribution of $\pi^*(\mathsf{Graph}(u))$ when $u : \mathbb{F} \to \mathbb{F}$ is a uniformly random function.

Proof. To sample a random permutation π conditioned on $\left\{\pi(\{i\} \times \mathbb{F})\right\}_i = \Phi \stackrel{\mathsf{def}}{=} \{\Phi_1, \ldots, \Phi_q\}$, one can sample a uniformly random permutation $\sigma : \mathbb{F} \to \mathbb{F}$ and q independent bijections $\pi_i : \mathbb{F} \to \Phi_{\sigma(i)}$, and then define $\pi(j, k) = \pi_j(k)$.

$\pi(\mathsf{Graph}(g))$ is defined by the set of points $\{\pi(j, g(j))\}_{j \in \mathbb{F}} = \{\pi_j(g(j))\}$. It is clear that sampling g uniformly at random corresponds to independently picking each $g(j)$ at random, which produces an identical distribution of $\pi(\mathsf{Graph}(g))$ as picking the bijections $\{\pi_j\}$ independently and uniformly at random. Thus, $\pi^*(\mathsf{Graph}(u))$ for a fixed π^* which respects the partition Φ, and a random u, is distributed identically to $\pi(\mathsf{Graph}(g))$ for a fixed g and a random π that respects Φ. □

Claim 2. For any partition Φ, any functions $g, g' : \mathbb{F} \to \mathbb{F}$, and any fixed permutation π^* that respects Φ, the distribution of $\left(\pi(\mathsf{Graph}(g)), \pi(\mathsf{Graph}(g'))\right)$ under $\Pr_\Phi$ is identical to the distribution of $\left(\pi^*(\mathsf{Graph}(u)), \pi^*(\mathsf{Graph}(u'))\right)$, where $u, u' : \mathbb{F} \to \mathbb{F}$ are jointly uniformly random conditioned on $\Delta(u, u') = \Delta(g, g')$.

Proof. We first consider the distribution under $\Pr_\Phi$ of $(x, x') = \left(\pi(\mathsf{Graph}(g)), \pi(\mathsf{Graph}(g'))\right)$, where g and g' are fixed. Because g and g' are functions, both x and x' will consist mostly of zeros, but for each $j \in \mathbb{F}$, they will contain a 1 in exactly one position in Φ_j. Recall from the proof of Claim 1 that π can be sampled by sampling a uniformly random permutation $\sigma : \mathbb{F} \to \mathbb{F}$ and q independent bijections $\pi_i : \mathbb{F} \to \Phi_{\sigma(i)}$, and defining $\pi(j, k) = \pi_j(k)$. Therefore, for any $j \in \mathbb{F}$ if $g(j) = g'(j)$ then x and x' will agree on the position within $\Phi_{\sigma(j)}$ at which they contain a 1 entry. Otherwise, they will disagree. Other than that, the positions are uniformly random within $\Phi_{\sigma(j)}$ because π_j is a random bijection. Moreover, since σ is a random permutation, the set of Φ_i's for which x, x' agree on the 1-entry is a random subset of size $\Delta(g, g')$.

Now consider the distribution of $(y, y') = \left(\pi^*(\mathsf{Graph}(u)), \pi^*(\mathsf{Graph}(u'))\right)$ where π^* is fixed and defined by σ^* and $\{\pi_i^*\}_{i \in \mathbb{F}}$. The same arguments show that for every $j \in \mathbb{F}$, y, y' agree on the positions within $\Phi_{\sigma^*(j)}$ at which they contain a 1 if and only if $u(j) = u'(j)$. Since u, u' are random and independent, the positions in $\Phi_{\sigma^*(j)}$ in which y, y' have a 1 are otherwise random because these positions are $\pi_j^*(u(j))$ and $\pi_j^*(u'(j))$, respectively. Additionally, the Φ_i's for which y, y' agree on the position of the 1 entry is a uniformly random subset of size $\Delta(g, g') = \Delta(u, u')$, because this set is $\{\sigma^*(j) : u(j) = u'(j)\}$, and u, u' are random and independent.

□

Claim 3. If $g_0, g_1 : \mathbb{F} \to \mathbb{F}$ are two independent uniformly random degree-λ polynomials, then $\Delta(g_0, g_1)$ is $e^{-\Omega(\lambda)}$-close to $\Delta(g_0', g_1')$ for uniformly random $g_0', g_1' : \mathbb{F} \to \mathbb{F}$.

Proof. For $i \in \mathbb{F}$, let X_i (respectively, Y_i) be indicator of the event that $g_0(i) = g_1(i)$ (respectively, $g_0'(i) = g_1'(i)$). Then X_i, Y_i are λ-wise independent with $\mathbb{E}[X_i] = \mathbb{E}[Y_i] = |\mathbb{F}|^{-1}$. The claim now follows from Lemma 4 below for $n = |\mathbb{F}|$. □

We now state the lemma used in the proof of Claim 3, the proof is deferred to the full version.

Lemma 4. *Let $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ be t-wise independent $\{0,1\}$-valued random variables with $t \geq 2e^2$, such that for all $i \in [n]$, $\mathbb{E}[Y_i] = \mathbb{E}[X_i] \stackrel{\mathsf{def}}{=} p_i$, let p denote $\frac{1}{n} \cdot \sum_i p_i$, and suppose that $p \leq \frac{t}{4n}$. Then the total variation distance $d_{\mathsf{TV}}(X, Y)$ is at most*

$$(n+3) \cdot \frac{(4pn/t)^{t/2}}{\prod_{i \in [n]}(1 - p_i)}$$

Now, we will show that $\mathbb{E}_{x \leftarrow \mathcal{D}_0}[f(\pi(x))]$ and $\mathbb{E}_{x \leftarrow \mathcal{D}_1}[f(\pi(x))]$, viewed as random variables that depend on π, have the same expectation and also have very small (negligible) variance.

Claim 4. For any $f : \{0,1\}^n \to \{0,1\}$ and any partition Φ,

$$\mathop{\mathbb{E}}_{\Phi}\Big[\mathop{\mathbb{E}}_{x \leftarrow \mathcal{D}_0}[f(\pi(x))]\Big] = \mathop{\mathbb{E}}_{\Phi}\Big[\mathop{\mathbb{E}}_{x \leftarrow \mathcal{D}_1}[f(\pi(x))]\Big].$$

Proof. Consider any $f : \{0,1\}^n \to \{0,1\}$ and any partition Φ. By Claim 1, there is a distribution $\mathcal{U}$ that is equal to the distribution (in $\Pr_\Phi$) of $\pi(\mathsf{Graph}(g))$ for all functions $g : \mathbb{F} \to \mathbb{F}$. Let μ denote $\mathbb{E}_{x' \leftarrow \mathcal{U}}[f(x')]$. Let P_b denote the probability mass function of $\mathcal{D}_b$. Then for any $b \in \{0,1\}$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\Phi}\left[\mathop{\mathbb{E}}_{x \leftarrow \mathcal{D}_b}[f(\pi(x))]\right] &= \mathop{\mathbb{E}}_{\Phi}\left[\sum_x P_b(x) \cdot f(\pi(x))\right] \\
&= \sum_x P_b(x) \cdot \mathop{\mathbb{E}}_{\Phi}[f(\pi(x))] \\
&= \sum_x P_b(x) \cdot \mu \\
&= \mu,
\end{aligned}$$

which does not depend on b. □

Now we analyze the variance. Recall that our goal is to show that $\mathrm{Var}_\Phi\left[\mathbb{E}_{x \leftarrow \mathcal{D}_b}[f(\pi(x))]\right]$ is negligible for $b \in \{0,1\}$. Because of Claim 3, this follows from the following more general claim.

Claim 5. Let $\mathcal{D}$ be any distribution on functions mapping $\mathbb{F}$ to $\mathbb{F}$. Suppose that when g and g' are sampled independently from $\mathcal{D}$ and $u, u' : \mathbb{F} \to \mathbb{F}$ are independent uniformly random functions, the distribution of $\Delta(g, g')$ is statistically ϵ-close to that of $\Delta(u, u')$.

Then, for any $f : \{0,1\}^n \to \{0,1\}$, any partition Φ,

$$\mathop{\mathrm{Var}}_{\Phi}\Big[\mathop{\mathbb{E}}_{g \leftarrow \mathcal{D}}\big[f\big(\pi(\mathsf{Graph}(g))\big)\big]\Big] \le \epsilon.$$

Proof. Let P denote the probability mass function of $\mathcal{D}$, and let π^* be an arbitrary permutation in S_n such that $\{\pi^*(\{i\} \times \mathbb{F})\}_i = \Phi$. By the definition of variance,

$$\mathop{\mathrm{Var}}_{\Phi}\left[\mathop{\mathbb{E}}_{g \leftarrow \mathcal{D}}[f(\pi(\mathsf{Graph}(g)))]\right] = \mathop{\mathbb{E}}_{\Phi}\left[\mathop{\mathbb{E}}_{g \leftarrow \mathcal{D}}[f(\pi(\mathsf{Graph}(g)))]^2\right] - \mathop{\mathbb{E}}_{\Phi}\left[\mathop{\mathbb{E}}_{g \leftarrow \mathcal{D}}[f(\pi(\mathsf{Graph}(g)))]\right]^2.$$

For the first term, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\Phi}[\mathop{\mathbb{E}}_{g\leftarrow\mathcal{D}}[f(\pi(\mathsf{Graph}(g)))]^2] &= \mathop{\mathbb{E}}_{\Phi}\left[\left(\sum_g P(g)\cdot f(\pi(\mathsf{Graph}(g)))\right)^2\right] \\
&= \sum_{g,h} P(g)\cdot P(h)\cdot \mathop{\mathbb{E}}_{\Phi}[f(\pi(\mathsf{Graph}(g)))\cdot f(\pi(\mathsf{Graph}(h)))] \qquad \text{(Claim 2)} \\
&= \mathop{\mathbb{E}}_{g,h\leftarrow\mathcal{D}}\left[\mathop{\mathbb{E}}_{\substack{u,v:\mathbb{F}\to\mathbb{F}\\ \Delta(u,v)=\Delta(g,h)}}\left[f(\pi^*(\mathsf{Graph}(u)))\cdot f(\pi^*(\mathsf{Graph}(v)))\right]\right].
\end{aligned}$$

For the second term, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\Phi}\left[\mathop{\mathbb{E}}_{g\leftarrow\mathcal{D}}[f(\pi(\mathsf{Graph}(g)))]\right]^2 \\
&= \left(\sum_g P(g)\cdot \mathop{\mathbb{E}}_{\Phi}\left[f(\pi(\mathsf{Graph}(g)))\right]\right)^2 \\
&= \left(\sum_g P(g)\cdot \mathop{\mathbb{E}}_{u:\mathbb{F}\to\mathbb{F}}\left[f(\pi^*(\mathsf{Graph}(u)))\right]\right)^2 \qquad \text{(Claim 1)} \\
&= \mathop{\mathbb{E}}_{u:\mathbb{F}\to\mathbb{F}}\left[f(\pi^*(\mathsf{Graph}(u)))\right]^2 \\
&= \mathop{\mathbb{E}}_{u,v:\mathbb{F}\to\mathbb{F}}\left[f(\pi^*(\mathsf{Graph}(u)))\cdot f(\pi^*(\mathsf{Graph}(v)))\right] \\
&= \mathop{\mathbb{E}}_{g,h:\mathbb{F}\to\mathbb{F}}\left[\mathop{\mathbb{E}}_{\substack{u,v:\mathbb{F}\to\mathbb{F}\\ \Delta(u,v)=\Delta(g,h)}}\left[f(\pi^*(\mathsf{Graph}(u)))\cdot f(\pi^*(\mathsf{Graph}(v)))\right]\right] \quad \text{(law of total expectation).}
\end{aligned}$$

The difference between these two expressions is only in the distribution of g and h over which the (outer) expectation is taken. Furthermore, the value whose expectation is computed lies in $[0,1]$ and depends only on the Hamming distance between g and h. The claim follows. □

Theorem 8 follows from Claims 3, 4, and 5, and Chebyshev's inequality. □

Acknowledgments

We thank Yuval Ishai for many useful discussions. We thank Fermi Ma for helpful discussions, in particular for pointing out that the blueprint of the DDH-based permuted puzzle extends also to the LPN setting, for simplifying one step of the proof of Proposition 3, and for allowing us to include these observations in the current work. We thank the anonymous TCC reviewers for helpful comments.

References

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108, 1996.

Ale03. Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307, 2003.

BCC+17. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *J. Cryptology*, 30(4):989–1066, 2017.

BFKL93. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 278–291, 1993.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

BHHO08. Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 108–125, 2008.

BIM00. Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 55–73, 2000.

BIPW17. Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *TCC (2)*, volume 10678 of *Lecture Notes in Computer Science*, pages 662–693. Springer, 2017.

BKW00. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 435–440, 2000.

BPR15. Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498, 2015.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *ECCC 2011*, 18:109, 2011.

CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Rothblum Guy N, Ron D. Rothblum, and Daniel WIchs. Fiat-Shamir: From practice to theory. In *STOC*, 2019.

CCRR18. Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption.

In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018.

CGKS95. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50, 1995.

CHK+19. Arka Rai Choudhuri, Pavel Hubávcek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. *IACR Cryptology ePrint Archive*, 2019:158, 2019.

CHR17. Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *TCC (2)*, volume 10678 of *Lecture Notes in Computer Science*, pages 694–726. Springer, 2017.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

FKM+18. Jean-Charles Faugère, Eliane Koussa, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. PKP-based signature scheme. *IACR Cryptology ePrint Archive*, 2018:714, 2018.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009, Proceedings*, pages 169–178. ACM, 2009.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

GJ02. Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

GK16. Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *TCC (A1)*, volume 9562 of *Lecture Notes in Computer Science*, pages 505–522. Springer, 2016.

GKL88. Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators (extended abstract). In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 12–24, 1988.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108. ACM, 2011.

HL18. Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 850–858. IEEE, 2018.

HY17. Pavel Hubávcek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1352–1371, 2017.

KMP19. Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin. On the complexity of the permuted kernel problem. *IACR Cryptology ePrint Archive*, 2019:412, 2019.

KO97. Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 364–373, 1997.

KRR17. Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 224–251, 2017.

LP12. Rodolphe Lampe and Jacques Patarin. Analysis of some natural variants of the PKP algorithm. In *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 209–214, 2012.

Mul54. David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Trans. I.R.E. Prof. Group on Electronic Computers*, 3(3):6–12, 1954.

Nao03. Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

PS19. Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. *IACR Cryptology ePrint Archive*, 2019:158, 2019.

Rab79. Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. *Technical Report, MIT Laboratory for Computer Science*, 1979.

RAD78. Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation, Academia Press*, 1978.

Ree54. Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Trans. of the IRE Professional Group on Information Theory (TIT)*, 4:38–49, 1954.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

Sha89. Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 606–609, 1989.