Trapdoor Hash Functions and PIR from Low-Noise LPN

Damiano Abram
Bocconi University

Giulio Malavolta
Bocconi University

Lawrence Roy
Aarhus University

Abstract

Trapdoor hash functions (TDHs) are compressing hash functions, with an additional trapdoor functionality: Given a encoding key for a function f, a hash on x together with a (small) input encoding allow one to recover $f(x)$. TDHs are a versatile tool and a useful building block for more complex cryptographic protocols.

In this work, we propose the first TDH construction assuming the (quasi-polynomial) hardness of the LPN problem with noise rate $\varepsilon = O(\log^{1+\beta} n/n)$ for $\beta > 0$, i.e., in the so-called low-noise regime. The construction achieves $2^{\Theta(\log^{1-\beta} \lambda)}$ compression factor. As an application, we obtain a private-information retrieval (PIR) with communication complexity $L/2^{\Theta(\log^{1-\beta} L)}$, for a database of size L. This is the first PIR scheme with non-trivial communication complexity (asymptotically smaller than L) from any code-based assumption.

1 Introduction

A *trapdoor hash function* (TDH) [DGI+19] $\mathsf{Hash} : \mathbb{Z}_2^\ell \to \mathbb{Z}_2^\lambda$ is a compressing (and deterministic) hash function, with an additional trapdoor-like functionality. Specifically, it is equipped with a generation algorithm Gen that takes as input a function f and allows one to generate a (public) encoding key ek and a (private) trapdoor td. Then, given $\mathsf{Hash}(x)$ and $\mathsf{Enc}(\mathsf{ek}, x)$, the trapdoor allows one to recover $f(x)$. Furthermore, the encoding key ek hides any information about the function f (function privacy). An important property of TDHs is that the size of the output of $\mathsf{Enc}(\mathsf{ek}, x)$ should be approximately the same as that of $f(x)$, i.e., the encoding algorithm should have a good *information rate*.

Besides being a cryptographic primitive of general interest, TDHs played a central role in recent works developing new cryptographic constructions, such as oblivious transfer [DGI+19, BBDP22, BDS23, BDS24], private information retrieval [DGI+19], fully-homomorphic encryption [BDGM19], lossy trapdoor functions [DGI+19], non-committing encryption [BBD+20, YKT19], correlation intractable hash functions and non-interactive zero-knowledge proofs [BKM20, JJ21], succinct arguments [CGJ+23], and homomorphic secret sharing [ARS24].

Given the wide array of applications of this primitive, it is natural to ask under which computational assumption one can construct TDHs with good encoding rate. The work of Döttling et al. [DGI+19] proposed constructions of TDHs under group-based assumptions (such as DDH, QR, DCR) or lattice-based assumptions (LWE). However, at present no construction of TDHs from code-based assumptions is known, highlight our lack of a full understanding for this primitive.

Code-based assumptions, such as the learning parity with noise (LPN) are one of the foundations for cryptography and have recently seen a surge in popularity in the context of post-quantum cryptography. The LPN problem [BFKL94] asserts that it is hard to distinguish between the two distributions

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx (\mathbf{A}, \mathbf{w})$$

where $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$, $\mathbf{s} \in \mathbb{Z}_2^n$, $\mathbf{w} \in \mathbb{Z}_2^m$ are uniformly sampled, whereas $\mathbf{e}$ is sampled from an i.i.d. Bernoulli distribution. The LPN problem has a worst-case to average-case reduction [KS06] and it is known that solving LPN also implies the existence of learning algorithms for 2-DNF formulas, juntas, and any function with a sparse Fourier spectrum [FGKP06].

The LPN noise is often regarded as more conservative than the LWE problem[1], due to the *lack of mathematical structure* that is instead present in general q-ary lattices. LPN has withstood an extensive cryptanalytic effort (more discussion on this later), which helps us in building confidence in the security of this assumption. Unfortunately, the aforementioned lack of structure makes it also more challenging to build cryptographic primitives from LPN, since many of the available construction techniques do not apply in the code settings. The lack of TDHs from code assumptions can be seen as a manifestation of this phenomenon. Besides TDHs, it is also a long standing open problem to build private information retrieval with non-trivial complexity from *any variant* of LPN.

1.1 Our Results

In this work, we propose the first TDH construction where the security assumes only the hardness of the LPN problem, in the low-noise regime. Existing constructions of TDHs [DGI+19] assume the intractability of a variety of different problems (such as DDH, DCR, QR, or LWE) all of which are structurally different from the LPN problem, with no reduction known one way or another. Our TDH only achieves a weaker notion of compactness compared to prior work, which we nevertheless show to be sufficient for some applications. Our main result can be summarized in the following theorem.

Theorem 1.1. (Informal). *For any $0 < \beta < 1$, there exists a TDH construction for linear functions secure against the hardness of the LPN problem with noise rate $\Theta(\log^{1+\beta} n/n)$. The TDH satisfies the following properties:*

- *(Correctness) It is correct with probability $1/2 + 1/\mathsf{poly}(\lambda)$.*
- *(Weak Compactness) The the size of the digest is smaller than the size of the input by a factor of $2^{\Theta(\log^{1-\beta}(\lambda))}$.*
- *(Rate) The size of the encoding for a 1-bit output is exactly 1 bit, i.e., it has rate 1.*
- *(Linear Decoding) Has a linear decoding algorithm.*

Our construction (Section 3) is obtained in two steps: We first build a simple base TDH from LPN with linear decoding, then we use the latter property to recursively compose the scheme with itself, in order to obtain better parameters. In the main body, we also explore different tradeoffs between the compactness of the hash and choosing a more aggressive parameter regime for the LPN problem.

We emphasize that the version of the LPN problem that we consider can only be quasi-polynomially secure (since there is an algorithm solving the problem running in quasi-polynomial

[1]Although the variant of LPN that we consider in this work has security that is at most quasi-polynomial, i.e., the so-called low-noise regime.

time), and therefore we recommend caution. Nevertheless, we also stress the hardness of LPN is well-studied [BK02, BKW03, Lyu05, MMT11, BLP11, BJMM12, EKM17] and the low-noise version of LPN has already been used in prior work to construct other cryptographic primitives, such as identity-based encryption [BLSV18, DGHM18]. Furthermore, the only construction for a (compressing) collision-resistant hash function from LPN [YZW⁺19, BLVW19] is also in the low-noise regime, and specifically with noise rate $\Theta(\log^2 n/n)$, which for us would correspond to setting $\beta = 1$. Since a TDH is also (among other things) a compressing collision-resistant hash, one cannot expect to obtain a TDH from a more noisy version of LPN, without first improving on the simpler primitive.

As an application of this primitive, we propose the first non-trivial PIR protocol from LPN (Section 4). Prior to our work, no PIR with sublinear communication complexity (in the size of the database) was known. We summarize this in the following theorem.

Theorem 1.2. (Informal). *Assuming the hardness of the LPN problem with noise rate* $\Theta(\log^{1+\beta} n/n)$, *for any* $0 < \beta < 1$, *there exists a two-message PIR protocol with communication* $L/2^{\Theta(\log^{1-\beta} L)}$, *for a database of size* L.

1.2 Technical Outline

We give a brief and informal overview of our approach, and we refer to the technical sections for more precise technical statements.

Trapdoor Hashing. First, let us recall the syntax of trapdoor hashing. A TDH consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where the setup procedure Setup is used to generate the hash key hk, which can be used as a regular hash function $h \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x})$ to produce a digest h. Additionally, a TDH is equipped with a generation algorithm $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{hk}, f)$, which takes as an additional input a function f that, in this work, we assume to be a linear function. One can then run an encoding algorithm on the same input to produce an encoding $t \leftarrow \mathsf{Enc}(\mathsf{hk}, \mathbf{x})$. Given the trapdoor td and a the digest h, the corresponding decoding algorithm returns a pair of encodings (t_0, t_1) where $t_0 \neq t_1$. For correctness, we require that $t_{f(\mathbf{x})} = t$ with sufficiently high probability. Note that this allows one to correctly decode $f(\mathbf{x})$, given the trapdoor and the hash, by simply comparing it with t.

We require that a TDH is *compact*, meaning that the size of the digest h is smaller than that of the input $\mathbf{x}$, and has good *rate*, meaning that the encoding t is efficient in terms of information. In this work we consider rate-1 schemes, where $|t| = |f(\mathbf{x})|$. Notice that, in these constructions, it holds that $t \oplus t_0 = f(\mathbf{x}) \oplus \varepsilon$, where ε is an error term. As for security, a TDH must be *function private*, which requires that for all f and f' the distributions

$$\{\mathsf{ek} | (\mathsf{ek}, \mathsf{td}) \overset{\$}{\leftarrow} \mathsf{Gen}(\mathsf{hk}, f)\} \approx \{\mathsf{ek} | (\mathsf{ek}, \mathsf{td}) \overset{\$}{\leftarrow} \mathsf{Gen}(\mathsf{hk}, f')\}$$

are computationally indistinguishable.

A Simple TDH from LPN. Before describing our construction, let us introduce the notion of a *sparsifier*, a function that takes as input a vector $\mathbf{x}$ and returns a low Hamming weight vector $\mathbf{x}'$. Sparsification is obtained by dividing the input vector into blocks and substituting each block with the corresponding unit vector. For this overview, we denote this sparsification function by $\mathbf{D}^{-1}(\cdot)$.

Note that the *inverse* of this sparsification procedure is a linear function, which implies that there exists a matrix $\mathbf{D}$ that satisfies the following identity

$$\mathbf{D} \cdot \mathbf{D}^{-1}(\mathbf{x}) = \mathbf{x}$$

for all vectors $\mathbf{x}$. Equipped with this tool, we can describe the standard LPN-based hash function [AHI+17] as

$$\mathsf{Hash}(\mathsf{hk} = \mathbf{A}, \mathbf{x}) = \mathbf{A}\mathbf{D}^{-1}(\mathbf{x})$$

where $\mathsf{hk} = \mathbf{A}$ is a uniformly sampled matrix. We are now in the position to describe our base TDH construction. For a given linear function f (in its coefficient representation $\mathbf{y}$), the Gen algorithm computes a key

$$\mathsf{ek} = \mathbf{A}^T\mathbf{s} \oplus \mathbf{e} \oplus \mathbf{D}^T \cdot \mathbf{y}$$

where $\mathbf{s}$ is uniformly sampled and set to be the trapdoor, whereas $\mathbf{e}$ is sampled from a Bernoulli distribution, with appropriate rate. The encoding algorithm simply computes $\mathsf{ek}^T \cdot \mathbf{D}^{-1}(\mathbf{x})$, whereas the decoding algorithm computes $t_0 = \mathbf{s}^T\mathsf{Hash}(\mathbf{A}, \mathbf{x})$ and $t_1 = t_0 \oplus 1$. To establish correctness, let us pretend for the moment that $\mathbf{e} = \mathbf{0}$, then it is clear that

$$\begin{aligned} \mathsf{ek}^T \cdot \mathbf{D}^{-1}(\mathbf{x}) &= \mathbf{s}^T\mathbf{A}\mathbf{D}^{-1}(\mathbf{x}) \oplus \mathbf{y}^T\mathbf{D}\mathbf{D}^{-1}(\mathbf{x}) \\ &= \mathbf{s}^T\mathbf{A}\mathbf{D}^{-1}(\mathbf{x}) \oplus \mathbf{y}^T\mathbf{x} \\ &= \mathbf{s}^T\mathsf{Hash}(\mathbf{A}, \mathbf{x}) \oplus f(\mathbf{x}) \\ &= t_{f(\mathbf{x})} \end{aligned}$$

which is exactly what the decoder expects. To take into account the presence of the noise, one must deal with the correctness error introduced by the extra term $\mathbf{e}^T\mathbf{D}^{-1}(\mathbf{x})$. Recalling that $\mathbf{D}^{-1}(\mathbf{x})$ has low Hamming weight and setting the parameters of the scheme carefully, one can ensure that the decoder maintains a bias towards the correct bit, while keeping the hash function (slightly) compressing. Specifically, the entries of $\mathbf{e}^T\mathbf{D}^{-1}(\mathbf{x})$ will be distributed according to Bernoulli distributions of parameter $\frac{1}{2} - \exp\left(-\Omega\left(\log^{\beta}(\lambda) \cdot |\mathbf{x}| \cdot |h|^{-1}\right)\right)$. This means that any improvement in compactness is paid *exponentially* in correctness! We therefore achieve only $o(\log^{1-\beta}\lambda)$-weak compactness, i.e. $|h| = \frac{|\mathbf{x}|}{o(\log^{1-\beta}\lambda)}$.

Finally, function privacy comes from a straightforward application of the LPN assumption, which ensures that switching the encoding key to a uniformly sampled vector is computationally indistinguishable.

Improved Efficiency via Recursive Composition. Our next observation is that the (noisy) decoding procedure is a *linear function* and there is therefore hope to attain better parameters by recursively composing the TDH with itself. In more details, we define our new hash function Hash^* as

$$\mathsf{Hash}^*(\mathbf{x}) = \mathsf{Hash}_{T-1} \circ \cdots \circ \mathsf{Hash}_0(\mathbf{x})$$

where Hash_i is the previously defined scheme, with appropriately chosen parameters. To sample an encoding key, we start with a coefficient encoding of the function $\mathbf{y}$ as before, and compute $(\mathsf{ek}_1, \mathsf{td}_1) \leftarrow \mathsf{Gen}_0(\mathsf{hk}, \mathbf{y})$. Using the linearity of decoding, we set $\mathbf{y}_1 := \mathsf{td}_1$ to be the function for the next iteration and we proceed this way until we generate $(\mathsf{ek}_T, \mathsf{td}_T)$. On the other hand, the new encoding function is defined as

$$t \leftarrow \mathsf{Enc}^*(\mathbf{x}) = \bigoplus_i \mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i)$$

where $\mathbf{x}_i$ is the input of Hash_i. It is easy to verify that

$$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i) \oplus t_{i,0} = \mathbf{y}_i^T \mathbf{x}_i \oplus \varepsilon_i$$

where $t_{i,0}$ is the first output of $\mathsf{Dec}(\mathbf{y}_{i+1}, \mathbf{x}_{i+1})$ and ε_i is an error term. Recalling that $\mathbf{y}_i^T \mathbf{x}_i = t_{0,i-1}$, we can conclude that

$$t \oplus \underbrace{\mathsf{Dec}(\mathbf{y}_T, \mathbf{x}_T)}_{\mathsf{Dec}^*} = \mathbf{y}^T \mathbf{x} \oplus \bigoplus_i \varepsilon_i$$

defining our new decoding algorithm Dec^*. It is possible to prove that, if ε_i is described by a Bernoulli distribution of parameter $\frac{1}{2} - \frac{\delta_i}{2}$, then the final error term $\bigoplus_i \varepsilon_i$ is described by a Bernoulli distribution of parameter $\frac{1}{2} - \frac{\prod_i \delta_i}{2}$. In other words, correctness degrades exponentially in T, however, at the same time, compactness improves exponentially! To conclude, any improvement in compactness is paid *polynomially* in correctness. This leads to significantly better efficiency.

Application: Private Information Retrieval. A TDH suggest a natural PIR protocol: Pass the database as an input $\mathbf{x}$ to the hash function to compute hk, and let the client compute the encoding key ek corresponding to a function f, which indexes the entry that the client is interested in. Privacy follows immediately from the function privacy of the TDH.

Unfortunately our TDH supports only linear functions, which in particular means that the size of f (in its coefficient encoding), and consequently the size of ek, would be linear in the size of the database. To solve this, we use a standard rebalancing algorithm, where we partition the database in blocks and we let the client retrieve an entry for each block. So long as the client sets the index of the "correct" block to be the one corresponding the the desired entry, correctness holds.

Finally, we have to contend with the fact that the TDH correctness is only approximate. This is solved by a standard parallel amplification, along with a majority vote in the end. Plugging in our newly constructed TDH, we obtain our final PIR protocol.

A Failed Attempt at Obtaining NIZKs from LPN via Correlation Intractability. A common approach to build non-interactive zero-knowledge proofs (NIZKs) without relying on idealised settings, such as the random oracle model, is to apply the Fiat-Shamir transform [FS87] using a correlation intractable hash functions [CGH04]. This is a particular type of hash function $\mathsf{CI.Hash}$ tackling a sparse relation $\mathcal{R}$: it guarantees the hardness of finding a value x such that $(x, \mathsf{CI.Hash}(x)) \in \mathcal{R}$.

In [BKM20], Brakerski, Koppula and Mour showed how to build correlation intractable hash functions using any rate-1, δ-correct, trapdoor hashing scheme for linear functions with particular efficiency properties. Their argument is the following: suppose that $\mathcal{R}$ is described by all pairs $(\mathbf{x}, \mathbf{y})$ where $\mathbf{y} = f(\mathbf{x})$ for some linear function f. The hash key of the correlation intractable scheme is a trapdoor hash encoding key obtained as $(\mathsf{ek}, \mathsf{td}) \overset{\$}{\leftarrow} \mathsf{TDH.Gen}(\mathsf{hk}, f)$ along with a random shift $\mathbf{r}$. The digest of $\mathbf{x}$ consists of $\mathsf{TDH.Enc}(\mathsf{ek}, \mathbf{x}) \oplus \mathbf{r}$.

Their argument is based on the following observation. Let $(t_0, t_1) \leftarrow \mathsf{TDH.Dec}(\mathsf{td}, h)$ where $h \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}, \mathbf{x})$ and observe that, since the trapdoor hashing scheme has rate 1,

$$\mathsf{TDH.Enc}(\mathsf{ek}, \mathbf{x}) \oplus \mathbf{r} = f(\mathbf{x}) \oplus t_0 \oplus \mathbf{r} \oplus \varepsilon,$$

where ε is an error vector whose entries are distributed according to i.i.d. Bernoulli distributions $\mathsf{Ber}(1-\delta)$. In order for $(\mathbf{x}, \mathsf{CI.Hash}(\mathbf{x})) \in \mathcal{R}$, we would therefore need that $t_0 \oplus \varepsilon = \mathbf{r}$. However, there may be an entropy issue: the information in t_0 is upper-bounded by the size of the digest in the trapdoor hashing scheme. Therefore, if this is particularly small, and so is $1-\delta$, there is not enough entropy for the support of the random variable $t_0 \oplus \varepsilon$ to contain a uniformly sampled $\mathbf{r}$ (except with negligible probability).

Unfortunately, our LPN-based trapdoor hashing schemes do not seem to be efficient enough to obtain correlation intractability, or at least, not with the approach of [BKM20]. Specifically, suppose that the the domain of our correlation intractable function has size ℓ and let m be the size of the output, where $m < \ell$. If we consider our recursive construction, in T steps, we obtain a digest of size $\ell \cdot 2^{-T}$ and an error probability roughly $\frac{1}{2} - 2^{-T\cdot\omega(1)}$, where the $\omega(1)$ term depends on the LPN parameters. The number of m-bit vectors with weight smaller than $m \cdot (\frac{1}{2} - x)$ can be approximated, using the Chernoff bound, to $2^m \cdot \exp(-2m \cdot x^2)$. Therefore, we can expect ε to contain $m - m \cdot 2^{-T\cdot\omega(1)}$ bits of information. Notice that if we add the $\frac{\ell}{2^T}$ bits of information contained in t_0, we obtain

$$m - m \cdot 2^{-T\cdot\omega(1)} + \ell \cdot 2^{-T} \geq m.$$

It may therefore be that the support of $t_0 \oplus \varepsilon$ contains all m-bit strings, including $\mathbf{r}$.

Similar issues are encountered when we try to follow the blueprint of [DGI+19] to build lossy trapdoor functions, two-round statistically sender-private oblivious transfer [BF22] or even fully efficient PIR: for all of these applications, we would need a two-round oblivious transfer protocol with constant *download rate*[2]. Due to the $1/2 - \mathsf{poly}(\lambda)$ error probability and the weak compactness, our TDH does not trivially achieve this: in this work, we have only obtained a 1-out-of-L OT where the download message has sublinear size in L (this is sufficient for PIR). The download rate remains, however, high.

1.3 Related Work

The LPN problem (and variants thereof) have been used to construct basic cryptographic primtives, such as public-key encryption [Ale03, ABW10] and message authentication codes [KPC+11]. For more advanced primitives, we know how to use LPN to construct two-round oblivious transfer [DGH+20], trapdoor functions [YZ16], homomorphic secret sharing [BCG+22, DIJL23], oblivious linear evaluation [BCGI18] multi-party computation with sublinear communication [CM21, BCM23], to mention a few. Very recently, a new variant of LPN (called dense-sparse LPN) has been proposed [DJ24], enabling the first construction of lossy trapdoor functions from code problems. The problem of PIR from LPN was explicitly left open in [DJ24].

[2]The download rate is defined as the ratio between the size of the download message (i.e. the second OT message) and the size of the output.

2 Preliminaries

Notation. We denote the security parameter by λ. Let κ be a statistical security parameter. For any positive integer n, we use $[n]$ to denote the set $\{0, 1, \dots, n-1\}$. We adopt the convention where vectors are denoted using bold font and matrices are indicated by bold-font capital letters. We assume that entries of vectors and rows and columns of matrices are indexed starting from 0. We use $\mathsf{Ber}(\varepsilon)$ to denote the Bernoulli distribution of parameter ε. In other words, the random variable is equal to 1 with probability ε and 0 with probability $1-\varepsilon$. In a similar way, we use $\mathsf{Ber}^{n\times m}(\varepsilon)$ to denote the distribution over $n \times m$ matrices, where each entry is distributed according to i.i.d. (independent, identically distributed) Bernoulli random variables $\mathsf{Ber}(\varepsilon)$. If $m = 1$, we simply write $\mathsf{Ber}^n(\varepsilon)$. For any integer $x \in [\ell]$, we use $\mathbf{u}_x$ to denote the binary unit vector of dimension 2^ℓ having x as special position. In other words, all entries of $\mathbf{u}_x$ will be equal to 0 except the one in position x which will be equal to 1. We denote the concatenation operator by $\|$. Given any matrix $\mathbf{A}$, we represent the transposed matrix by $\mathbf{A}^\intercal$. For ease of notation, whenever we deal with functions $f(\lambda)$ of the security parameter, we often drop the dependency in λ, writing just f. All logarithms are in base 2, exp denotes exponentiation in the natural base e. For any binary string $\mathbf{v}$, we use $|\mathbf{v}|$ to denote its length. We denote the row-vector with n ones by 1^n, we use $\otimes$ to denote the Kronecker product between matrices. We use id_n to denote the identity matrix of dimension n. We use $\equiv_p$ to denote perfectly indistinguishable distributions.

2.1 Learning Parity with Noise

Definition 2.1. (Learning Parity with Noise). *Let $n = n(\lambda)$, $m = m(\lambda)$ and $\varepsilon = \varepsilon(\lambda)$ be efficiently computable functions of the security parameter, where $n(\lambda)$ and $m(\lambda)$ are polynomially bounded positive integers and $\varepsilon(\lambda)$ is a real value in $[0, 1]$. The $\mathsf{LPN}^\varepsilon_{n,m}$ assumption holds if, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that:*

$$\left| \Pr\left[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{w}) = 1 \,\middle|\, \begin{array}{l} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n} \\ \mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n \\ \mathbf{e} \xleftarrow{\$} \mathsf{Ber}(\varepsilon)^m \\ \mathbf{w} \leftarrow \mathbf{A}\cdot\mathbf{s} \oplus \mathbf{e} \end{array} \right] - \Pr\left[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{w}) = 1 \,\middle|\, \begin{array}{l} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n} \\ \mathbf{w} \xleftarrow{\$} \mathbb{Z}_2^m \end{array} \right] \right| \le \mathsf{negl}(\lambda).$$

In this paper, we rely on LPN with low noise rate $\varepsilon = O\left(\frac{\alpha(\lambda)\cdot\log n}{n}\right)$ where $\alpha(\lambda) = \omega(1)$. In this parameter regime, the assumption is still believed to hold for any polynomial $m(n)$, however, with rather weak security guarantees [EKM17]: Due to Gaussian elimination, we can hope to achieve security only against adversaries that run in quasi-polynomial time $o\left(n^{\alpha(\lambda)}\right)$.

Before continuing with other preliminaries, we recall the following fundamental lemma.

Lemma 2.2. (Piling-Up Lemma [Mat94]). *Let $X_0, \dots, X_{n-1}$ be independent Bernoulli random variables where X_i is distributed according to $\mathsf{Ber}(\varepsilon_i)$. Then, $X_0 \oplus \dots \oplus X_{n-1}$ is distributed according to*

$$\mathsf{Ber}\left(\frac{1}{2} - \frac{1}{2}\cdot\prod_{i\in[n]}(1-2\varepsilon_i)\right).$$

2.2 Trapdoor Hashing

We recall the definition of trapdoor hashing from [DGI+19].

Definition 2.3. (Trapdoor Hashing [DGI+19])**.** *A trapdoor hashing scheme for the function class* $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ *with input size* $\ell(\lambda)$ *consists of a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- Setup *is randomised and takes as input the security parameter* 1^λ*. The output is a hash key* hk*.*
- Hash *is deterministic and takes as input a hash key* hk *and a binary string* $\mathbf{x} \in \mathbb{Z}_2^{\ell(\lambda)}$*. The output is a digest* h*.*
- Gen *is randomised and takes as input an hash key* hk *and the description of a function* $f \in \mathcal{C}_\lambda$*. The output is an encoding key* ek *and a trapdoor* td*.*
- Enc *is deterministic and takes as input an encoding key* ek *and a binary string* $\mathbf{x} \in \mathbb{Z}_2^{\ell(\lambda)}$*. The output is an encoding* t*.*
- Dec *is deterministic and takes as input a trapdoor* td *and a digest* h*. The output is a pair of encodings* (t_0, t_1) *where* $t_0 \neq t_1$*.*

Definition 2.4. (Correctness)**.** *Let* $\delta(\lambda)$ *be a function of the security parameter. Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a trapdoor hashing scheme for the function class* $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ *with input size* $\ell(\lambda)$*. We say that the scheme is* $\delta(\lambda)$*-correct if, for every* $\lambda \in \mathbb{N}$, $\mathbf{x} \in \mathbb{Z}_2^{\ell(\lambda)}$, $f \in \mathcal{C}_\lambda$ *and hash key* $\mathsf{hk} \in \mathsf{Setup}(1^\lambda)$*, we have*

$$\Pr\left[t = t_{f(\mathbf{x})} \middle| \begin{array}{l} h \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \\ (\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{hk}, f) \\ t \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{x}) \\ (t_0, t_1) \leftarrow \mathsf{Dec}(\mathsf{td}, h) \end{array} \right] \geq \delta(\lambda).$$

We say that the trapdoor hashing scheme is fully correct if $\delta(\lambda)$ is negligible. Observe that the above probability is taken only over the randomness of Gen.

We recall the definition of function privacy, that says that the encoding key should hide all information about the function f.

Definition 2.5. (Function Privacy)**.** *Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a trapdoor hashing scheme for the function class* $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ *with input size* $\ell(\lambda)$*. We say that the scheme is function private if, for every PPT adversary* $\mathcal{A}$ *and sequence of functions* $(f_\lambda^0, f_\lambda^1)_{\lambda\in\mathbb{N}}$ *where* $f_\lambda^0, f_\lambda^1 \in \mathcal{C}_\lambda$ *for every* $\lambda \in \mathbb{N}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that, for every* $\lambda \in \mathbb{N}$*, we have*

$$\left| \Pr\left[\mathcal{A}(1^\lambda, \mathsf{hk}, \mathsf{ek}) = 1 \middle| \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{hk}, f_\lambda^0) \end{array} \right] - \right.$$
$$\left. \Pr\left[\mathcal{A}(1^\lambda, \mathsf{hk}, \mathsf{ek}) = 1 \middle| \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{hk}, f_\lambda^1) \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

Next, we define the rate for trapdoor hashing.

Definition 2.6. (Rate). *Let $\mu(\lambda)$ be a function of the security parameter. Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a trapdoor hashing scheme for the function class $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ with input size $\ell(\lambda)$. We say that the scheme has rate μ if, for every $\lambda \in \mathbb{N}$, $\mathbf{x} \in \mathbb{Z}_2^{\ell(\lambda)}$ and $f \in \mathcal{C}_\lambda$, it holds that*

$$\frac{|f(\mathbf{x})|}{|\mathsf{Enc}(\mathsf{ek}, \mathbf{x})|} \leq \mu(\lambda)$$

for all $\mathsf{hk} \in \mathsf{Setup}(1^\lambda)$ *and* $\mathsf{ek} \in \mathsf{Gen}(\mathsf{hk}, f)$.

Finally, we define a weaker version of the compactness property, than the one presented in [DGI+19].

Definition 2.7. (Weak Compactness). *Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a trapdoor hashing scheme for the function class $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ with input size $\ell(\lambda)$. Let $\gamma(\lambda)$ be a function of the security parameter. We say that the scheme is $\gamma(\lambda)$-weakly compact if, for every $\lambda \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{Z}_2^{\ell(\lambda)}$, we have*

$$|\mathsf{Hash}(\mathsf{hk}, \mathbf{x})| \leq \frac{\ell(\lambda)}{\gamma(\lambda)}$$

where $\mathsf{hk} \in \mathsf{Setup}(1^\lambda)$.

2.3 Two-Round Private Information Retrieval

We recall the standard definition of private information retrieval [KO97].

Definition 2.8. (Two-Round Private Information Retrieval). *A two-round private information retrieval (PIR) consists of a triple of PPT algorithms* $(\mathsf{Query}, \mathsf{Response}, \mathsf{Dec})$ *with the following syntax:*

- Query *takes as input the security parameter 1^λ, the database size 1^L and an index $i \in [L]$. The output is an encoding key* ek *and a secret-key* sk.
- $\mathsf{Response}$ *is deterministic and takes as input an encoding key* ek *and a database $\mathbf{v} \in \mathbb{Z}_2^L$. The output is an encoding e.*
- Dec *is deterministic and takes as input a secret key* sk *and an encoding e. The output is a value $z \in \mathbb{Z}_2$*

Definition 2.9. (Correctness). *A two-round PIR* $(\mathsf{Query}, \mathsf{Response}, \mathsf{Dec})$ *is correct if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that, for every $\lambda, L \in \mathbb{N}$, index $i \in [L]$ and database $\mathbf{v} \in \mathbb{Z}_2^L$, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, e) \neq \mathbf{v}[i] \middle| \begin{array}{l} (\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Query}(1^\lambda, 1^L, i) \\ e \leftarrow \mathsf{Response}(\mathsf{ek}, \mathbf{v}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.10. (Query-Privacy). *A two-round PIR* $(\mathsf{Query}, \mathsf{Response}, \mathsf{Dec})$ *is query-private if, for every PPT adversary $\mathcal{A}$, polynomial function $L(\lambda)$ and sequence of indexes $(i_\lambda^0, i_\lambda^1)_{\lambda \in \mathbb{N}}$ where $i_\lambda^0, i_\lambda^1 \in [L(\lambda)]$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\left| \Pr\left[\mathcal{A}(1^\lambda, 1^{L(\lambda)}, \mathsf{ek}) = 1 \middle| \begin{array}{l} b \xleftarrow{\$} \mathbb{Z}_2 \\ (\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Query}(1^\lambda, 1^{L(\lambda)}, i_\lambda^b) \end{array}\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Definition 2.11. (Compactness of Two-Round Private Information Retrieval). *Let $\alpha(\lambda)$ and $L(\lambda)$ be polynomial functions of the security parameter. A two-round private information retrieval (PIR)* $(\mathsf{Query}, \mathsf{Response}, \mathsf{Dec})$ *is $\alpha(\lambda)$-compact with respect to databases of size $L(\lambda)$ if, for all sufficiently large $\lambda \in \mathbb{N}$, for every index $i \in [L(\lambda)]$ and every database $\mathbf{v} \in \mathbb{Z}_2^{L(\lambda)}$, we have*

$$|\mathsf{ek}| + |\mathsf{Response}(\mathsf{ek}, \mathbf{v})| \leq \alpha(\lambda)$$

where $(\mathsf{ek}, \mathsf{sk}) \in \mathsf{Query}(1^\lambda, 1^{L(\lambda)}, i)$.

2.4 Sparsification with Linear Reconstruction

Definition 2.12. (Sparsification). *Let ℓ and c be positive values. We define the algorithm $\mathbf{D}_{\ell,c}^{-1}$ which, on input a vector $\mathbf{x} \in \mathbb{Z}_2^\ell$, computes the following operations:*

1. *Split $\mathbf{x}$ into $B := \frac{\ell}{c \cdot \log \ell}$ blocks $x_0, \ldots, x_{B-1}$ of size $c \cdot \log \ell$. View them as values in $[\ell^c]$.*
2. *For every $i \in [B]$: ${\mathbf{x}'}_i \leftarrow \mathbf{u}_{\mathbf{x}_i}$, the $\mathbf{x}_i$-th unit vector.*
3. *Output $\mathbf{x}' \leftarrow ({\mathbf{x}'}_0 \parallel \ldots \parallel \mathbf{x}'_{B-1})$.*

Definition 2.13. (Reconstruction Matrix). *Let ℓ and c be positive values. We define the matrix $\mathbf{D}'_{\ell,c}$ as the $(c \cdot \log \ell) \times \ell^c$ matrix where, for every $i \in [\ell^c]$, the i-th column consists of the binary representation of the positive integer i. We define $\mathbf{D}_{\ell,c}$ as $\mathbf{D}'_{\ell,c} \otimes \mathsf{id}_B$, where $B = \frac{\ell}{c \cdot \log \ell}$.*

Lemma 2.14. *Let ℓ and c be positive values, and consider any vector $\mathbf{x} \in \mathbb{Z}_2^\ell$. Then $\mathbf{D}_{\ell,c}^{-1}(\mathbf{x})$ is a binary vector of length $\frac{\ell^{1+c}}{c \cdot \log \ell}$ having Hamming weight exactly $\frac{\ell}{c \cdot \log \ell}$. Furthermore, we have*

$$\mathbf{D}_{\ell,c} \cdot \mathbf{D}_{\ell,c}^{-1}(\mathbf{x}) = \mathbf{x}.$$

Proof: We start by observing that $\mathbf{D}_{\ell,c}^{-1}(\mathbf{x})$ is obtained by concatenating $B := \frac{\ell}{c \cdot \log \ell}$ unit vectors of length ℓ^c. This ensures that $\mathbf{D}_{\ell,c}^{-1}(\mathbf{x})$ has length $\ell^{1+c}/c \cdot \log \ell$ and Hamming weight $\ell/c \cdot \log \ell$. To conclude the proof, we observe that

$$\mathbf{D}_{\ell,c} \cdot \mathbf{D}_{\ell,c}^{-1}(\mathbf{x}) = (\mathbf{D}'_{\ell,c} \cdot \mathbf{u}_{x_0} \parallel \ldots \parallel \mathbf{D}'_{\ell,c} \cdot \mathbf{u}_{x_{B-1}})$$

where x_i is the integer representation of the i-th $(c \cdot \log \ell)$-bit block in $\mathbf{x}$. It is easy to see that, by the way the matrix was defined, for any $y \in [\ell^c]$, we have that $\mathbf{D}'_{\ell,c} \cdot \mathbf{u}_y$ is equal to the binary representation of the integer y. In conclusion, $\mathbf{D}_{\ell,c} \cdot \mathbf{D}_{\ell,c}^{-1}(\mathbf{x}) = (x_0 \parallel \ldots \parallel x_{B-1}) = \mathbf{x}$. $\square$

3 Trapdoor Hashing for Linear Functions from Low-Noise LPN

In this work, we obtain a construction of trapdoor hash with a linear encoding and rate exactly 1, which we formally define below.

Definition 3.1. (Linear Decoding, Rate-1 Trapdoor Hashing). *We say that a rate-1 trapdoor hashing scheme* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has linear decoding if the trapdoor* td *output by* Gen *consists of a binary vector the same size as the digest and the decoding t_0 output by* Dec *is* $\mathsf{td}^\mathsf{T} \cdot h$.

We present our basic construction of trapdoor hash in Figure 1. We summarize the main result of this section in the following theorem.

Theorem 3.2. *Under the hardness of the* $\mathsf{LPN}^{\varepsilon}_{n,m}$ *assumption, the construction in Figure 1 is a* $\log \ell/\gamma(\lambda)$*-weakly compact, rate-1, function private,* $(1/2 + 1/2 \cdot \ell^{-4\eta\cdot\alpha(\lambda)/c\cdot\gamma(\lambda)})$*-correct trapdoor hashing scheme for linear functions. The construction has linear decoding. The size of the encoding key is* $\ell^{1+c}/c \cdot \log \ell$*.*

Proof: We start by proving $(1/2 + 1/2 \cdot \ell^{-4\frac{\eta\cdot\alpha(\lambda)}{c\cdot\gamma(\lambda)}})$-correctness. We observe that

$$\begin{aligned} t = \mathbf{v}^{\intercal} \cdot \mathbf{x}' &= \mathbf{s}^{\intercal} \cdot A \cdot \mathbf{x}' \oplus \mathbf{e}^{\intercal} \cdot \mathbf{x}' \oplus \mathbf{y}^{\intercal} \cdot \mathbf{D}_{\ell,c} \cdot \mathbf{x}' \\ &= \mathbf{s}^{\intercal} \cdot \mathbf{d} \oplus \mathbf{e}^{\intercal} \cdot \mathbf{x}' \oplus \mathbf{y}^{\intercal} \cdot \mathbf{x} \\ &= t_{\mathbf{y}^{\intercal}\cdot\mathbf{x}} \oplus \mathbf{e}^{\intercal} \cdot \mathbf{x}'. \end{aligned}$$

Since $\mathbf{x}'$ has Hamming weight $B = \frac{\ell}{c\cdot\log\ell}$, each entry of $\mathbf{e}^{\intercal} \cdot \mathbf{x}'$ can be rewritten as the sum of B i.i.d. Bernoulli random variables of parameter ε. By the piling-up lemma (Lemma 2.2), we conclude that each entry of $\mathbf{e}^{\intercal} \cdot \mathbf{x}'$ is described by an independent Bernoulli random variable of parameter $\frac{1}{2} - \frac{1}{2} \cdot (1 - 2\varepsilon)^B$. Now, we observe that

$$(1-2\varepsilon)^B \geq 2^{-2\cdot(2\varepsilon)\cdot B} = 2^{-4\cdot\eta\cdot\frac{\alpha(\lambda)\cdot\log n}{n}\cdot\frac{\ell}{c\cdot\log\ell}} \geq 2^{-4\cdot\eta\cdot\frac{\alpha(\lambda)\cdot\log^2\ell}{\gamma(\lambda)\cdot\ell}\cdot\frac{\ell}{c\cdot\log\ell}} = \ell^{-\frac{4\cdot\eta\cdot\alpha(\lambda)}{c\cdot\gamma(\lambda)}}.$$

In other words, $t = t_{\mathbf{y}^{\intercal}\cdot\mathbf{x}}$ with probability at least $(1/2 + 1/2 \cdot \ell^{-4\frac{\eta\cdot\alpha(\lambda)}{c\cdot\gamma(\lambda)}})$.

Next, we focus on weak compactness. This property can be easily verified by observing that the size of the digest is $n = \frac{\ell(\lambda)\cdot\gamma(\lambda)}{\log\ell}$. It is also straightforward that the scheme has rate 1.

Finally, we prove function privacy. We observe that, under the $\mathsf{LPN}^{\varepsilon}_{n,m}$ assumption, the pair $(\mathbf{A}^{\intercal}, \mathbf{A}^{\intercal} \cdot \mathbf{s} \oplus \mathbf{e})$ is computationally indistinguishable from $(\mathbf{A}^{\intercal}, \mathbf{u})$, where $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_2^m$. In other words, no PPT adversary can distinguish $\mathbf{v}$ from a random vector of size m, even if we provide it with $\mathbf{A}$. This ensures the privacy of $\mathbf{y}$. $\square$

We highlight the following implications of Theorem 3.2.

- (Corollary 1) Setting $\gamma(\lambda) = \beta \cdot \log \ell$ for $\beta > 0$, $\alpha(\lambda) = \log n$, and η and c so that $\frac{4\eta}{c} = \nu$ for $\nu > 0$, we obtain a β-weakly compact, rate-1, function private, $(1/2 - \Omega(\ell^{-\nu}))$-correct trapdoor hashing scheme for linear functions. This assumes the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ problem, where $\varepsilon = \Theta(\frac{\log^2 n}{n})$.

- (Corollary 2) Let $\ell = \lambda^{\Theta(1)}$ and let $1 > \beta > 0$ and $\nu > 0$. Set $\gamma(\lambda) = \log^{\beta} \ell$, $\alpha(\lambda) = \log^{\beta} n$, and choose η and c so that $\frac{4\eta}{c} = \nu$. There exists a $(\log^{1-\beta} \ell)$-weakly compact, rate-1, function private, $(1/2 + \Omega(\ell^{-\nu}))$-correct trapdoor hashing scheme for linear functions, assuming the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ problem, where $\varepsilon = \Theta(\frac{\log^{1+\beta} n}{n})$.

- (Corollary 3) Let $\nu > 0$ and let $\ell = \lambda^{\Theta(1)}$. Setting $\gamma(\lambda) = \log\log\ell$, $\alpha(\lambda) = \log\log n$ and η and c so that $\frac{4\eta}{c} = \nu$, we obtain a $(\log\ell/\log\log\ell)$-weakly compact, rate-1, function private, $(1/2 + \Omega(\ell^{-\nu}))$-correct trapdoor hashing scheme for linear functions. This assumes the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ assumption, where $\varepsilon = \Theta(\frac{\log n\cdot\log\log n}{n})$.

Trapdoor Hashing for Linear Functions from Low-Noise LPN

Parameters: Let $\ell(\lambda) = \lambda^{\Theta(1)}$ be the size of the hash function input. Let the linear function be described a row vector in $\mathbb{Z}_2^{\ell}$. Consider any functions $\alpha(\lambda) = \omega(1)$ and $\gamma(\lambda) = \Omega(\alpha(\lambda))$. Pick arbitrary positive constants c and η. We rely on the $\mathsf{LPN}_{n,m}^{\varepsilon}$ assumption, where $n = \frac{\ell(\lambda)\cdot\gamma(\lambda)}{\log \ell}$ and $\varepsilon = \eta \cdot \frac{\alpha(\lambda)\cdot \log n}{n}$. Let m be $\frac{\ell^{1+c}}{c\cdot \log \ell}$.

```
Setup($1^\lambda$)
    1. $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n\times m}$
    2. Output $\mathsf{hk} := \mathbf{A}$
Hash($\mathsf{hk} = \mathbf{A}, \mathbf{x}$)
    1. $\mathbf{x}' \leftarrow \mathbf{D}_{\ell,c}^{-1}(\mathbf{x})$
    2. $\mathbf{d} \leftarrow \mathbf{A} \cdot \mathbf{x}'$
    3. Output $h := \mathbf{d}$
Gen($\mathsf{hk} = \mathbf{A}, \mathbf{y}$)
    1. $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{n}$
    2. $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}^{m}(\varepsilon)$
    3. $\mathbf{v} \leftarrow \mathbf{A}^{\intercal} \cdot \mathbf{s} \oplus \mathbf{e} \oplus \mathbf{D}_{\ell,c}^{\intercal} \cdot \mathbf{y}$
    4. Output $\mathsf{ek} := \mathbf{v}$ and $\mathsf{td} := \mathbf{s}$
Enc($\mathsf{ek} = \mathbf{v}, \mathbf{x}$)
    1. $\mathbf{x}' \leftarrow \mathbf{D}_{\ell,c}^{-1}(\mathbf{x})$
    2. $t \leftarrow \mathbf{v}^{\intercal} \cdot \mathbf{x}'$
    3. Output $t$
Dec($\mathsf{td} = \mathbf{s}, h = \mathbf{d}$)
    1. $t_0 \leftarrow \mathbf{s}^{\intercal} \cdot \mathbf{d}$
    2. Output $t_0$ and $t_1 := t_0 \oplus 1$.
```

Figure 1: Trapdoor hashing for linear functions from low-noise LPN

A More Efficient Trapdoor Hashing Scheme

Parameters: For every $i \in [T]$, let $\mathsf{LDTH}_i = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a $\gamma_i(\lambda)$-weakly compact, rate-1, $(1/2 + 1/2 \cdot \delta_i(\lambda))$-correct trapdoor hashing scheme with input size $\ell_i(\lambda)$ where $\ell_i(\lambda) := \frac{\ell_{i-1}(\lambda)}{\gamma_{i-1}(\lambda)}$, if $i > 0$, and $\ell_0(\lambda) := \ell(\lambda)$. Assume that all these schemes have linear decoding. Suppose that $\prod_{i \in [t]} \delta_i(\lambda) = \lambda^{O(1)}$.

```
Setup(1^λ)
    1. For every $i \in [T]$:
        (a) $\mathsf{hk}_i \xleftarrow{\$} \mathsf{LDTH}_i.\mathsf{Setup}(1^\lambda)$
    2. Output $\mathsf{hk} := (\mathsf{hk}_0, \ldots, \mathsf{hk}_{T-1})$

$\mathsf{Hash}(\mathsf{hk} = (\mathsf{hk}_0, \ldots, \mathsf{hk}_{T-1}), \mathbf{x})$
    1. $\mathbf{x}_0 \leftarrow \mathbf{x}$
    2. For $i = 0, \ldots, T-1$:
        (a) $\mathbf{x}_{i+1} \leftarrow \mathsf{LDTH}_i.\mathsf{Hash}(\mathsf{hk}_i, \mathbf{x}_i)$
    3. Output $h := \mathbf{x}_T$

$\mathsf{Gen}(\mathsf{hk} = (\mathsf{hk}_0, \ldots, \mathsf{hk}_{T-1}), \mathbf{y})$
    1. $\mathbf{y}_0 \leftarrow \mathbf{y}$
    2. For $i = 0, 1, \ldots, T-1$:
        (a) $(\mathsf{ek}_i, \mathbf{y}_{i+1}) \xleftarrow{\$} \mathsf{LDTH}_i.\mathsf{Gen}(\mathsf{hk}_i, \mathbf{y}_i)$
    3. Output $\mathsf{ek} := (\mathsf{hk}_0, \ldots, \mathsf{hk}_{T-1}, \mathsf{ek}_0, \ldots, \mathsf{ek}_{T-1})$ and $\mathsf{td} := \mathbf{y}_T$

$\mathsf{Enc}(\mathsf{ek} = (\mathsf{hk}_0, \ldots, \mathsf{hk}_{T-1}, \mathsf{ek}_0, \ldots, \mathsf{ek}_{T-1}), \mathbf{x})$
    1. $\mathbf{x}_0 \leftarrow \mathbf{x}$
    2. For $i = 0, \ldots, T-1$:
        (a) $\mathbf{x}_{i+1} \leftarrow \mathsf{LDTH}_i.\mathsf{Hash}(\mathsf{hk}_i, \mathbf{x}_i)$
    3. $t \leftarrow \bigoplus_{i \in [T]} \mathsf{LDTH}_i.\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i)$
    4. Output $t$

$\mathsf{Dec}(\mathsf{td} = \mathbf{y}_T, h = \mathbf{x}_T)$
    1. $(t_0, t_1) \leftarrow \mathsf{LDTH}_T.\mathsf{Decode}(\mathbf{y}_T, \mathbf{x}_T)$
    2. Output $t_0$ and $t_1$
```

Figure 2: A more efficient trapdoor hashing scheme

Improved Efficiency with Self-Composition. Here we show how to improve the efficiency of our trapdoor hash construction by composing our construction recursively with itself. In fact, we prove a somewhat more general statement that holds for any trapdoor hash function with linear decoding. The scheme is presented in Figure 2.

Theorem 3.3. *The construction in Figure 2 is a $(\prod_{i\in[t]}\gamma_i(\lambda))$-weakly compact, rate-1, function private, $(1/2+1/2\cdot\prod_{i\in[T]}\delta_i)$-correct trapdoor hashing scheme for linear functions. The construction has linear decoding.*

Proof: We start by proving correctness. For every $i \in [T]$, let (t_0^{i+1}, t_1^{i+1}) be the output of $\mathsf{LDTH}_i.\mathsf{Dec}(\mathbf{y}_{i+1}, \mathbf{x}_{i+1})$. By the linear decoding property, we have that

$$\mathsf{LDTH}_i.\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i) \oplus t_0^{i+1} = \mathbf{y}_i^\intercal \cdot \mathbf{x}_i \oplus e_i$$

where e_i is a small noise term where distributed according to $\mathsf{Ber}(\frac{1}{2} - \frac{1}{2}\cdot\delta_i')$ and $\delta_i' \geq \delta_i$. Remember that, for every $i > 0$, we have $\mathbf{y}_i^\intercal \cdot \mathbf{x}_i = t_0^i$. We conclude that

$$t \oplus t_0 = \mathbf{y}^\intercal \cdot \mathbf{x} \oplus \bigoplus_{i\in[T]} e_i.$$

By the piling-up lemma (Lemma 2.2), we observe that $\bigoplus_{i\in[T]} e_i$ is distributed according to $\mathsf{Ber}(1/2-1/2\cdot\prod_{i\in[T]}\delta_i')$. We also observe that

$$\frac{1}{2} - \frac{1}{2}\cdot\prod_{i\in[T]}\delta_i' \leq \frac{1}{2} - \frac{1}{2}\cdot\prod_{i\in[T]}\delta_i.$$

This proves that the construction is $(1/2+1/2\cdot\prod_{i\in[T]}\delta_i)$-correct.

Next, we focus on weak compactness. This property can be easily verified by observing that, for every $i > 0$ the size of $\mathbf{x}_i$ is $\ell_{i-1}(\lambda)/\gamma_{i-1}(\lambda)$. Given that $\ell_0(\lambda) = \ell(\lambda)$, we conclude that the size of the digest $\mathbf{x}_T$ is

$$|\mathbf{x}_T| = \frac{\ell(\lambda)}{\prod_{i\in[t]}\gamma_i(\lambda)}.$$

It is immediate to observe that the scheme has rate 1.

Finally, we prove function privacy. This easily follows from the function privacy of $(\mathsf{LDTH}_i)_{i\in[t]}$. Specifically, we follow a hybrid argument: for every $i \in [t+1]$, we consider Hybrid i, in which $\mathsf{ek}_i, \ldots, \mathsf{ek}_{t-1}$ are all generated by encoding the $\ell_i(\lambda)$-dimensional vector where all the entries are 0. Thanks to the function privacy of LDTH_i, Hybrid i is computationally indistinguishable from Hybrid $i+1$. Notice also that in Hybrid 0, ek contains no information about the encoded linear function $\mathbf{y}$. Hybrid T on the other hand corresponds to the usual encoding of $\mathbf{y}$. $\square$ We highlight a few corollaries of Theorem 3.3.

- (Corollary 1) Let $T(\lambda) = \log^{1-\beta}\ell$. For every $i \in [T]$, instantiate the linear decoding hash functions LDTH_i using the scheme in Figure 1 with parameters $\gamma(\lambda) = \frac{1}{2}\cdot\log\ell_i$ and $\alpha(\lambda) = \log^\beta n$. Choose η and c so that $\frac{4\eta}{c} = \nu$, where $\nu > 0$, and set β to be some positive constant $1 > \beta > 0$. There exists a $(2^{\Theta(\log^{1-\beta}\ell)})$-weakly compact, rate-1, and $(1/2+\Omega(\ell^{-\nu}))$-correct trapdoor hashing scheme for linear functions. The construction has linear decoding and is function private assuming the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ problem, where $\varepsilon = \Theta(\frac{\log^{1+\beta} n}{n})$.

- (Corollary 2) Let $T(\lambda) = \frac{\log \ell}{\log\log \ell}$. For every $i \in [T]$, instantiate the linear decoding hash functions LDTH_i using the scheme in Figure 1 with parameters $\gamma(\lambda) = \frac{1}{2} \cdot \log \ell_i$ and $\alpha(\lambda) = \log\log n$. Choose η and c so that $\frac{4\eta}{c} = \nu$. Let $\nu > 0$ and let $\ell = \lambda^{\Theta(1)}$. There exists a $2^{\frac{\Theta(\log \ell)}{\log\log \ell}}$-weakly compact, rate-1, $(1/2 + \Omega(\ell^{-\nu}))$-correct trapdoor hashing scheme for linear functions. The construction has linear decoding and is function private assuming the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ problem, where $\varepsilon = \Theta(\frac{\log n \cdot \log\log n}{n})$.

4 Private Information Retrieval from Trapdoor Hashing

Next, we show how to use our trapdoor hash construction to build a two-round PIR protocol with sublinear communication complexity. The protocol is described in Figure 3.

Theorem 4.1. *Let $\kappa = \omega(\log \lambda)$. The construction in Figure 3 is a correct, query private, two-round PIR protocol. Furthermore, consider any $L = \lambda^{\Theta(1)}$. The construction in Figure 3 is $\alpha(\lambda)$-compact with respect to $L(\lambda)$, where*

$$\alpha(\lambda) = M_0(\lambda) + \kappa \cdot \delta^{-2}(\lambda) \cdot M_1(\lambda) + \frac{L(\lambda)}{\gamma(\lambda)} + \mu \cdot \frac{L(\lambda)}{\ell(\lambda)} \cdot \kappa \cdot \delta^{-2}(\lambda).$$

Proof: We start by proving correctness. We observe that, due to the $\left(\frac{1}{2} + \delta(\lambda)\right)$-correctness of the trapdoor hash function, for every $k \in [T]$, we have

$$\Pr\Big[z_k = \mathbf{v}[i]\Big] \geq \frac{1}{2} + \delta(\lambda).$$

Now, let ε_k be the random variable described by $z_k \oplus \mathbf{v}[i]$. We observe that $\varepsilon_0, \dots, \varepsilon_{T-1}$ are described by T independent Bernoulli distributions of parameter smaller than $\frac{1}{2} - \delta(\lambda)$. By the Chernoff bound, we observe that the probability that $\mathsf{Maj}(\varepsilon_0, \dots, \varepsilon_{T-1}) \neq 0$ is bounded from above by $\exp(-2\kappa)$. Since this is a negligible function in λ, we conclude that $\mathsf{Maj}(z_0, \dots, z_{T-1}) = \mathbf{v}[i]$ with overwhelming probability.

Next, we prove query privacy. This is an immediate consequence of function privacy of the trapdoor hashing scheme. We can prove this by relying on a hybrid argument. Specifically, we rely on $T+1$ hybrids. Rewrite i^0_λ as $\iota^0_\lambda \cdot \ell(\lambda) + \bar{\iota}^0_\lambda$ and i^1_λ as $\iota^1_\lambda \cdot \ell(\lambda) + \bar{\iota}^1_\lambda$, where $\bar{\iota}^0_\lambda, \bar{\iota}^1_\lambda \in [\ell(\lambda)]$. In Hybrid 0, we generate the encoding key ek using $\mathsf{Query}(1^\lambda, 1^{L(\lambda)}, i^0_\lambda)$. In Hybrid T, on the other hand, we generate ek using $\mathsf{Query}(1^\lambda, 1^{L(\lambda)}, i^1_\lambda)$. Finally, for every intermediate Hybrid j, we generate the first j trapdoor hashing encoding keys in ek using $\mathsf{TDH.Gen}(\mathsf{hk}, \mathbf{u}_{\bar{\iota}^1_\lambda})$, whereas we generate the rest using $\mathsf{TDH.Gen}(\mathsf{hk}, \mathbf{u}_{\bar{\iota}^0_\lambda})$. It is easy to observe that any pair of consecutive Hybrids are indistinguishable thanks to function privacy of TDH. □

Instantiations. We can use different trapdoor hashing constructions, in order to obtain different PIR protocols trading compactness for a more aggressive choice of parameters of the underlying computational assumption. In what follows we always assume that the database size $L = \lambda^{\Theta(1)}$ is a polynomial in the security parameter.

For instance, instantiate the trapdoor hashing with the scheme from Theorem 3.3 (Corollary 1), with input length $\ell(\lambda) = L^{\frac{1}{c+1}}$. We choose c as a constant such that ℓ^c upper bounds $M_0(\lambda) +$

Private Information Retrieval from Trapdoor Hashing

Parameters: Let $\mathsf{TDH} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a $\gamma(\lambda)$-weakly compact, rate-$\mu(\lambda)$, function private, $\left(\frac{1}{2} + \delta(\lambda)\right)$-correct trapdoor hashing scheme for linear functions with input size $\ell(\lambda)$. Suppose that the size of hk is $M_0(\lambda)$, whereas the length of an encoding key is $M_1(\lambda)$. Let T be $\kappa \cdot \delta(\lambda)^{-2}$.

```
$\mathsf{Query}(1^\lambda, 1^L, i)$
    1. Rewrite $i$ as $\iota \cdot \ell + \bar{\iota}$ where $\bar{\iota} \in [\ell]$.
    2. $\mathsf{hk} \xleftarrow{\$} \mathsf{TDH.Setup}(1^\lambda)$
    3. For every $k \in [T]$: $(\mathsf{ek}_k, \mathsf{td}_k) \xleftarrow{\$} \mathsf{TDH.Gen}(\mathsf{hk}, \mathbf{u}_{\bar{\iota}})$
    4. Output $\mathsf{ek} := (\mathsf{hk}, \mathsf{ek}_0, \ldots, \mathsf{ek}_{T-1})$, $\mathsf{sk} = (\iota, \mathsf{td}_0, \ldots, \mathsf{td}_{T-1})$
$\mathsf{Response}(\mathsf{ek} = (\mathsf{hk}, \mathsf{ek}_0, \ldots, \mathsf{ek}_{T-1}), \mathbf{v})$
    1. Split $\mathbf{v}$ into $B := L/\ell$ blocks $\mathbf{v_0}, \ldots, \mathbf{v_{B-1}}$
    2. For every $j \in [B]$: $h_j \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}, \mathbf{v_j})$
    3. For every $j \in [B]$ and $k \in [T]$: $t_{j,k} \leftarrow \mathsf{TDH.Enc}(\mathsf{ek}_k, \mathbf{v_j})$
    4. Output $e := (h_0, \ldots, h_{B-1}, (t_{0,k})_{k\in[T]}, \ldots, (t_{B-1,k})_{k\in[T]})$
$\mathsf{Dec}(\mathsf{sk} = (\iota, \mathsf{td}_0, \ldots, \mathsf{td}_{T-1}), e = (h_0, \ldots, h_{B-1}, (t_{0,k})_{k\in[T]}, \ldots, (t_{B-1,k})_{k\in[T]}))$
    1. For every $k \in [T]$: $(t^0_{\iota,k}, t^1_{\iota,k}) \leftarrow \mathsf{TDH.Dec}(\mathsf{td}_k, h_\iota)$
    2. For every $k \in [T]$: set $z_k \leftarrow 0$ if $t_{\iota,k} = t^0_{\iota,k}$. Otherwise, set $z_k \leftarrow 1$.
    3. Output $z := \mathsf{Maj}(z_0, \ldots, z_{T-1})$.
```

Figure 3: Private information retrieval from trapdoor hashing

$\kappa \cdot \delta^{-2}(\lambda) \cdot M_1(\lambda)$. Choose any constant $\nu < 1/2$, any $1 > \beta > 0$, and set $\kappa = \ell^r$ where r is a positive constant smaller than $1 - 2\nu$. We obtain a two-round $(L(\lambda) \cdot 2^{-\Theta(\log^{1-\beta} L)})$-compact PIR protocol, assuming the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ problem, where $\varepsilon = \Theta(\frac{\log^{1+\beta} n}{n})$.

On the other hand, we can also instantiate the trapdoor hashing with the scheme from Theorem 3.3 (Corollary 2) having input length $\ell(\lambda) = L^{\frac{1}{c+1}}$. We choose c as a constant such that ℓ^c upper bounds $M_0(\lambda) + \kappa \cdot \delta^{-2}(\lambda) \cdot M_1(\lambda)$. Choose any constant $\nu < \frac{1}{2}$ and $\kappa = \ell^r$ where r is a positive constant smaller than $1 - 2\nu$. Under the hardness of the $\mathsf{LPN}^{\varepsilon}_{n,m}$ assumption, where $\varepsilon = \Theta(\frac{\log n \cdot \log \log n}{n})$, there exists a $L(\lambda) \cdot 2^{-\frac{\Theta(\log L)}{\log \log L}}$-compact, query private, two-round PIR protocol.

Can Recursion Help? One may notice that in our PIR protocol, the server sends more information than what the client actually needs: The decoding procedure never uses any of $h_j, t_{j,0}, \dots, t_{j,T-1}$ except when $j = \iota$. A standard technique to further reduce the communication complexity of two-round PIR protocols is to recurse the scheme. Specifically, we can run another instance of the PIR protocol on a database that consists of the bit representation of e, where the client's goal is to retrieve the entries corresponding to $h_\iota, t_{\iota,0}, \dots, t_{\iota,T-1}$.

Unfortunately this idea runs quickly into problems: At each step of the recursion, the number of bits that the client needs to receive gets is raised by a constant greater than 1. Thus, we can recurse at most a constant number of times before the communication complexity becomes super-polynomial. To summarise, the number of bits the client needs to retrieve grows so fast that soon will exceed the size of the whole database.

Acknowledgments

D.A. and G.M. are supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). G.M. is also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972. L.R. is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement number 101124977 (DECRYPSIS) and the Danish Independent Research Council under Grant-ID DFF-0165-00107B (C3PO).

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 171–180. ACM Press, June 2010.

[AHI+17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 7:1–7:31, 67, January 2017. LIPIcs.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 301–330. Springer, Cham, May 2024.

[BBD$^+$20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 58–87. Springer, Cham, November 2020.

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 157–186. Springer, Cham, May / June 2022.

[BCG$^+$22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Cham, August 2022.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[BCM23] Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 159–189. Springer, Cham, April 2023.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437. Springer, Cham, December 2019.

[BDS23] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. A framework for statistically sender private OT with optimal rate. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 548–576. Springer, Cham, August 2023.

[BDS24] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. Two-round maliciously-secure oblivious transfer with optimal rate. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 271–300. Springer, Cham, May 2024.

[BF22] Nir Bitansky and Sapir Freizeit. Statistically sender-private OT from LPN and derandomization. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 625–653. Springer, Cham, August 2022.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Berlin, Heidelberg, August 1994.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, Berlin, Heidelberg, April 2012.

[BK02] Piotr Berman and Marek Karpinski. Approximating minimum unsatisfiability of linear equations. In David Eppstein, editor, *13th SODA*, pages 514–516. ACM-SIAM, January 2002.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 738–767. Springer, Cham, August 2020.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 743–760. Springer, Berlin, Heidelberg, August 2011.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Cham, April / May 2018.

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 619–635. Springer, Cham, May 2019.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CGJ$^+$23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 635–668. Springer, Cham, August 2023.

[CM21] Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 842–870. Springer, Cham, October 2021.

[DGH$^+$20] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 768–797. Springer, Cham, May 2020.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Cham, March 2018.

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Cham, August 2019.

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 315–348. Springer, Cham, August 2023.

[DJ24] Quang Dao and Aayush Jain. Lossy cryptography from code-based assumptions. In *Annual International Cryptology Conference*, pages 34–75. Springer, 2024.

[EKM17] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 486–514. Springer, Cham, August 2017.

[FGKP06] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *47th FOCS*, pages 563–574. IEEE Computer Society Press, October 2006.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from subexponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 3–32. Springer, Cham, October 2021.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997.

[KPC$^+$11] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, Berlin, Heidelberg, May 2011.

[KS06] Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the HB and HB+ protocols. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 73–87. Springer, Berlin, Heidelberg, May / June 2006.

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 378–389. Springer, 2005.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 386–397. Springer, Berlin, Heidelberg, May 1994.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, Berlin, Heidelberg, December 2011.

[YKT19] Yusuke Yoshida, Fuyuki Kitagawa, and Keisuke Tanaka. Non-committing encryption with quasi-optimal ciphertext-rate based on the DDH problem. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 128–158. Springer, Cham, December 2019.

[YZ16] Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 214–243. Springer, Berlin, Heidelberg, August 2016.

[YZW$^+$19] Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from sub-exponential learning parity with noise. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 3–24. Springer, Cham, December 2019.