Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes

Geoffroy Couteau[1], Peter Rindal[2], Srinivasan Raghuraman[2]

[1] CNRS, IRIF, Université de Paris
couteau@irif.fr
[2] Visa Research
peterrindal@gmail.com, srini131293@gmail.com

Abstract. We put forth new protocols for oblivious transfer extension and vector OLE, called *Silver*, for SILent Vole and oblivious transfER. Silver offers extremely high performances: generating 10 million random OTs on one core of a standard laptop requires only 300ms of computation and 122KB of communication. This represents 37% less computation and $\sim 1300\times$ less communication than the standard IKNP protocol, as well as $\sim 4\times$ less computation and $\sim 14\times$ less communication than the recent protocol of Yang et al. (CCS 2020). Silver is *silent*: after a one-time cheap interaction, two parties can store small seeds, from which they can later *locally* generate a large number of OTs *while remaining offline*. Neither IKNP nor Yang et al. enjoys this feature; compared to the best known silent OT extension protocol of Boyle et al. (CCS 2019), upon which we build up, Silver has $19\times$ less computation, and the same communication. Due to its attractive efficiency features, Silver yields major efficiency improvements in numerous MPC protocols.
Our approach is a radical departure from the standard paradigm for building MPC protocols, in that we do *not* attempt to base our constructions on a well-studied assumption. Rather, we follow an approach closer in spirit to the standard paradigm in the design of symmetric primitives: we identify a set of fundamental structural properties that allow us to withstand all known attacks, and put forth a candidate design, guided by our analysis. We also rely on extensive experimentations to analyze our candidate and experimentally validate their properties. In essence, our approach boils down to constructing new families of linear codes with (plausibly) high minimum distance and extremely low encoding time. While further analysis is of course welcomed in order to gain total confidence in the security of Silver, we hope and believe that initiating this approach to the design of MPC primitives will pave the way to new secure primitives with extremely attractive efficiency features.

1 Introduction

Secure multiparty computation (MPC) allows n parties to jointly evaluate a function f, while leaking no information on their own input beyond the output of the function. It is a fundamental problem in cryptography, which has received considerable attention since its introduction in the seminal works of Yao [Yao86], and Goldreich, Micali, and Wigderson [GMW87b, GMW87a]. While early feasibility results for MPC were mainly of theoretical interest, MPC protocols have enjoyed tremendous improvements in the past decade.

Oblivious transfers (OT) are perhaps the most fundamental building block for MPC protocols. In a random OT, two parties receive respectively (s_0, s_1) and (s_b, b), where (s_0, s_1) are two random *strings*, and b is a random *selection bit*. Random OT is a complete primitive for secure computation [Kil88] (in particular, it implies chosen OT, where the sender picks (s_0, s_1) and the receiver picks b themselves), and modern MPC protocols rely on it. Efficiency improvements in protocols for generating OTs directly translate into improvements for a plethora of MPC protocols.

OT Extension. A long line of work, initiated with the breakthrough work of [IKNP03], has therefore sought to develop increasingly efficient protocols for generating a large number of random OTs. At a high level, OT extension protocols [IKNP03, KOS15, KKRT16, OOS17] turns a small number of base OTs into a near-arbitrary number of OTs, using only cheap operations. The latest generation of these protocols, initiated in [BCG+17], leverages the notion of pseudorandom correlation generators (PCGs) [BCGI18, BCG+19b] to enable the construction of extremely efficient OT extension protocols. This line of work recently culminated with the protocols of [BCG+19a, SGRR19, WYKW20, YWL+20].

Silent OT Extension. While PCGs allow for very efficient constructions of OT extension, this is not their main *claim to fame*: perhaps their most remarkable feature is that they allow the construction of *silent* OT extension protocols. A silent protocol has the property that: after a short interaction, with communication and computation essentially independent of the target number of OTs, the parties can locally store small correlated seeds. Then, the parties can later retrieve these seeds, and *without any further interaction* stretch them into a very large number of OTs. Unfortunately, while the protocols of [BCG+19a] enjoy the silent feature, the running time improvements in [SGRR19, WYKW20, YWL+20] were achieved at the cost of sacrificing this crucial property.

1.1 Our Results

In this work, we design new protocols for silent oblivious transfer extension and silent vector oblivious linear evaluation (VOLE). The latter is defined over a field $\mathbb{F}$ and allows a receiver with input $x \in \mathbb{F}$ to obtain $x \cdot \mathbf{a} + \mathbf{b}$ from a sender with input vectors $(\mathbf{a}, \mathbf{b})$ over $\mathbb{F}$. VOLE is another important building block in some of the most prominent secure computation tasks; e.g. the current most efficient private set intersection [RS21]. We call our (family of) protocols *Silver*, which stands for SILent Vole and oblivious transfER. In addition its silent feature, Silver exhibits extremely good performances, significantly outperforming the most efficient OT extension protocols [IKNP03, YWL+20] on all fronts.

At the heart of our results is a radical departure from previous works on secure computation. To put it bluntly, *we decidedly give up on provable security reductions to any well-studied assumption.* Instead, our protocols are based on the conjectured hardness of decoding new, *heuristically designed* linear codes (or, equivalently, the hardness of a new learning parity with noise (LPN) variant). Our approach for building these new linear codes is much closer in spirit to the de facto standard approach for building efficient block ciphers and hash functions in symmetric cryptography: using a general framework that encompasses essentially all known attacks on LPN and syndrome decoding, we identify the core properties that guarantee resistance of our new assumptions against existing attacks. Then, we extract a number of fundamental design criteria which guide the design of codes with these properties. Eventually, we rely on these design criteria together with extensive simulations to experimentally identify, with good confidence, the codes that exhibit the best properties for our constructions, while plausibly leading to very hard instances of the syndrome decoding problem.

1.2 Philosophy of our Approach

The construction of a cryptographic primitive or protocol can follow two complementary design strategies: a *top-down* approach, which starts from well-established cryptographic assumptions and aims at finding the most efficient construction whose security provably reduces to these assumptions, or a *bottom-up* approach, which tries to find the minimal construction that resists all known attacks, and relies on heuristic design criteria to build an intuition about the concrete security. Traditionally, secure computation has focused on the former, while symmetric cryptography (e.g. block cipher design) followed the latter.

The top-down approach has many attractive features – it deepens our theoretical understanding of the feasibility of cryptographic primitives, enlightens their relation to other primitives, and allows us to spend cryptanalytic efforts on a small set of assumptions. However, this sometimes comes at a huge cost in terms of efficiency: there is often a large gap between the best efficiency which can be achieved from well-established assumptions, and the efficiency which can be achieved with heuristic designs (consider the efficiency gap between SHA-256 and discrete-logarithm-based hash functions). When (our theoretical understanding of) a cryptographic primitive reaches a sufficient level of maturity, it is natural to envision the alternative bottom-up approach, in order to achieve real-world efficiency. This is the position that we advocate for in this work.

In the same way that symmetric cryptography has identified core families of attacks (e.g. linear and differential) and extracted a set of design principles for constructing primitives which plausibly resists them (e.g. substitution-permutation networks), our aim is to initiate the study of the most fundamental MPC primitives, oblivious transfer and its variants, under this angle. Pursuing this approach has the potential of yielding considerable efficiency improvements for MPC and strikes us as a natural next step for putting the efficiency of MPC primitives on par with that of symmetric primitives.

Our work being the first (to our knowledge) to study OT under the lens of heuristic cryptographic design, our constructions should of course be treated with the necessary caution. We invested a considerable effort in developing a rigorous understanding of which design criteria are likely to yield secure and efficient constructions, and relied on extensive experimental simulations to validate that our candidates satisfy these criteria; however, further study is welcomed in order to gain total confidence in their security. Given that Silver withstands the test of time, it will allow for significant improvements for numerous MPC protocols. And if not, we are confident that our analysis will motivate further constructions and analyses from which secure and efficient candidates will emerge.

1.3 Overview of our Methodology

Our starting point is the recent line of work on pseudorandom correlation generators (PCG) [BCG+17, BCGI18, BCG+19b]. PCGs allow to securely generate long, pseudorandom correlated strings, using minimal communication. Among the most remarkable achievements of this line of work is *silent oblivious transfer extensions* (SOT extension)[BCG+19a, SGRR19]. These protocols have two phases: (1) the two parties interact to distributively generate short correlated seeds with communication and computation essentially independent of the target number of OTs; (2) the parties locally expand the seeds, without any interaction, into a large number of pseudorandom OTs. Afterwards, these OTs can be converted into chosen-input OTs using standard methods. Very recently, efficiency improvements were obtained by [YWL+20, WYKW20]. However, this came at the cost of sacrificing the silent feature. In practice, the ability to confine the bulk of the computation to an entirely offline phase, is a crucial efficiency feature.

The SOT Protocol of [BCG+19a]. Our approach builds upon the protocol of [BCG+19a]. Let us briefly recall its high level intuition:

1. the parties generate additive shares of $x \cdot \mathbf{e}$, where $x \in \mathbb{F}$ is known to the sender, and $\mathbf{e} \in \mathbb{F}^n$ is a random *sparse* vector, known to the receiver. This can be done with minimal communication using the GGM puncturable pseudorandom function.
2. they multiply the shares of $x \cdot \mathbf{e}$ with a public matrix G, obtaining shares of $x \cdot \mathbf{a}$, for $\mathbf{a} = \mathbf{e} \cdot G^{\intercal}$. Given a uniform G, $\mathbf{a}$ is pseudorandom under LPN.
3. Optionally, the shares can be hashed to generate pseudorandom OTs.

Generating additive shares of $x \cdot \mathbf{e}$ is extremely efficient, requiring merely two calls to AES for each entry of the vector. The matrix-vector multiplication, however, is the bulk of the computation: in [BCG+19a], G is a matrix over $\mathbb{F}_2^{k \times n}$, where k is the target number of OTs and $n = c \cdot k$ for some small constant $c > 1$. MPC protocols commonly require a number of OTs in the millions (if not more), making this step impractical unless G has some structure that allows for fast matrix-vector multiplication. This leads to a tradeoff between efficiency and confidence in the security: when H is a truly random matrix, the multiplication is impractical, but security reduces to the standard syndrome decoding/LPN assumption. Structured matrices give better efficiency, but security reduces to syndrome decoding variants which are less well-understood.

[BCG+19a] settled for a reasonable middle ground, by letting G be a random matrix with a quasi-cyclic structure. On the one hand, this structure allows for matrix-vector multiplication in quasilinear time using fast Fourier transform; on the other hand, the underlying assumption (hardness of decoding quasi-cyclic linear codes) has been used in candidate post-quantum code-based cryptographic primitives submitted to the NIST competition, and are therefore relatively well studied. While this choice leads to a reasonably efficient construction, it remains somewhat unsatisfying: it seems very likely that there exists alternative choices for G which have significantly better efficiency, yet still are secure.

However, the particular set of constraints of silent OT extension is very different from all previous coding theory primitives: typically the dimension of the code is minimized, allow high noise rate, and rely on codes with a hidden structure to enable efficient decoding given a secret. In contrast, in the SOT application, the code dimension scales with the target number of OTs (hence typically millions), the noise rate must remain very low, and no hidden structure or efficient decoding property is required. As a result, there exists no well-established assumption regarding codes tailored for our unusual set of constraints.

Our Approach: a Design Methodology for Constructing G. In this work, we choose to approach the problem differently. Let us call a public matrix $G \in \mathbb{F}_2^{k \times n}$ *SOT-friendly* if it satisfies the following two properties:

- Security: it is infeasible to distinguish $\mathbf{e} \cdot G^\intercal$ from uniform (for sparse $\mathbf{e}$).
- Efficiency: the mapping $\mathbf{x} \to \mathbf{x} \cdot G^\intercal$ can be computed in *strict linear time*.

We develop a methodology for constructing SOT-friendly matrices by directly identifying some core structural properties of G which guarantee that distinguishing $\mathbf{e} \cdot G^\intercal$ from random cannot be done using essentially all known attacks on LPN and code-based cryptographic primitives. Yet the mapping $\mathbf{x} \to \mathbf{x} \cdot G^\intercal$ can be computed in strict linear time. Our methodology does not "start from zero": it builds upon well-known results related to breaking these assumptions.

1.4 Our Design Criteria

The first property can be stated in one sentence: *G should generate a code with large minimum distance*. For the second property, we focus on the following sufficient condition: we restrict our attention to matrices G which have a *sparse parity-check matrix* H (i.e., H is a sparse matrix in $\mathbb{F}_2^{m \times n}$ such that $H^\intercal G = 0$) such that H are in *approximate lower triangular form*.

Large minimum distance and security. Given a matrix G, the problem of distinguishing $\mathbf{e} \cdot G^\intercal$ from random (for a random sparse vector $\mathbf{e}$) is the *decisional syndrome decoding problem* with respect to $G^\intercal$. The name LPN is commonly used to denote the syndrome decoding assumption in the cryptographic community. As such, we will use both terms interchangeably. It is well-known that distinguishing $\mathbf{e} \cdot G^\intercal$ reduces to the following problem: given a parity-check matrix H of G, distinguish the distribution $\{\mathbf{b} = \mathbf{x} \cdot H + \mathbf{e}\}$ (where $\mathbf{x}$ is a uniformly random vector over $\mathbb{F}_2^m$ and $\mathbf{e}$ is a random length-n sparse vector) from the uniform distribution (indeed, if $\mathbf{b}$ is indistinguishable from random, then so is $\mathbf{b} \cdot G^\intercal = (\mathbf{x} \cdot H + \mathbf{e}) \cdot G^\intercal = \mathbf{e} \cdot G^\intercal$), which is the learning parity with noise assumption, with dimension n and number of samples m, for the code matrix H. Both LPN and the syndrome decoding problem have been heavily studied in the past decades, and many attacks have been developed. A core observation (which is folklore, and was made explicitly e.g. in [BCG+20]) is that essentially all known attacks share a common high level structure: *the distinguisher computes a linear function in the samples* $\mathbf{b}$ (but can depend arbitrarily on the matrix H). But if the code generated by G has large minimum distance d, the distribution $H \cdot \mathbf{x}$ for random $\mathbf{x}$ must be d-wise independent, which implies that no weight-$t \leq d$ linear function $\mathbf{v}^\intercal \cdot \mathbf{b}$ of $\mathbf{b} = \mathbf{x} \cdot H + \mathbf{e}$ can possibly distinguish it from random. However, if $\mathbf{v}$ has high weight, then the distribution of $\mathbf{v}^\intercal \cdot \mathbf{e}$ for a random sparse vector $\mathbf{e}$ is close to uniform, and so is $\mathbf{v}^\intercal \cdot \mathbf{b}$. In this work, we formalize this folklore observation, and use it to derive a concrete heuristic for choosing the parameters of an SOT-friendly matrix. Our concrete heuristic is the following:

If two codes have the same minimum distance & dimensions, their decision syndrome decoding problems likely have the same level of security.

Therefore, when choosing concrete parameters, we will use as a baseline the codes underlying well-studied syndrome decoding variants (e.g. random linear codes in syndrome decoding, or LDPC codes in Alekhnovich's assumption [Ale03]) and set parameters to achieve the same minimum distance that these codes exhibit. We make two additional comments before moving on to the second property:

- In practice, it is generally very hard to compute the minimum distance of a family of codes. We will provide some efficient concrete choices where provable bounds exists. However, in our most efficient instantiations, we will instead rely on extensive simulations to analyze the minimum distance of the code family using an optimized *minimum distance estimator*, from which we will heuristically derive the minimum distance on large dimensions.
- In existing attacks against LPN/syndrome decoding, the number of noisy coordinates plays a crucial role. However, it has a small impact on the overall efficiency of the SOT: scaling the noise by some factor increases the (very small) amount of communication and computation in the first phase, but has no impact on the second phase. Therefore, even if our hypothesis turns out to be too aggressive, we can actually significantly increase the security level, by increasing the number of coordinates, at a minor cost.

Linear-time encodable LDPC codes. Low-density parity-check codes (LDPC) have a sparse parity-check H, were introduced in the seminal work of Gallagher [Gal62], and are among the most well-studied objects in coding theory. Certain random LDPC codes are known to exhibit a good minimum distance [Gal62] and can be decoded efficiently. On the other hand, their encoding time (i.e., the time to evaluate the mapping $\mathbf{x} \to \mathbf{x} \cdot G$) grows quadratically with the dimension in general. Due to the transposition principle (Section 4), our linear map $\mathbf{x} \to \mathbf{x} \cdot G^\intercal$ is efficient if and only if LDPC encoding $\mathbf{x} \to \mathbf{x} \cdot G$ is. Hence, finding LDPC codes whose generating matrix is SOT-friendly boils down to finding linear-time LDPC codes with large distance.

The seminal work of Richardson and Urbanke [RU01] identified a sufficient condition for efficient encoding of LDPC codes: if the parity-check matrix of the code can be brought, using row and column permutations only, into an approximate lower triangular form with a gap g (see Figure 9), then the encoding time can be brought down to $O(n + g^2)$ (instead of $O(n^2)$). Hence, whenever the gap g is below $O(\sqrt{n})$, encoding can be done in linear time using the *g-ALT encoder* of Richardson and Urbanke.

Achieving Fast Encoding and High Minimum Distance. Guided by the above, we therefore seek to construct new families of structured LDPC codes which simultaneously appear to achieve high minimum distance, yet can be encoded extremely efficiently with (our optimized variant of) the g-ALT encoder of Richardson and Urbanke [RU01] as presented in Section 4. Here, we use as a starting point the Tillich-Zémor (TZ) family of codes [TZ06]. TZ codes have appealing features in our setting: they provably achieve *almost* linear minimum distance, and can be encoded in linear time. However, their structure is also suboptimal in our specific setting: their code is not cache friendly and has sublinear distance due to degree-2 variable nodes. In [TZ06], the presence of these degree-2 variable nodes is motivated by the fact that they allow for high performance iterative decoding. In contrast, our application does not require any decoding property whatsoever. Hence, in Section 6 we refine the TZ codes to tailor them to our setting, improving the concrete minimum distance and encoding time.

We achieve this by iteratively refining our design, using extensive simulations to track the presence of *bad local structures* which, when they show up, lead to worse minimum distance guarantees. We also develop a method to increase the gap g of the code, which significantly improve the average-minimum distance properties and reduce the minimum distance variance (concretely, TZ codes are in approximate lower triangular form with gap $g = 1$, while we increase the gap to $g = 24$ without harming encoding efficiency). We fine-tune the structure of the matrix to minimize the number of cache misses in the encoding algorithm, which have a significant performance impact. To fine-tune the best possible choices of parameters in the low cache-misses setting, we compute, for many randomly generated choices of parameters, the average minimum distance and worst-case minimum distance over 10,000+ random samples of the code matrix.

1.5 Efficiency

After performing this iterative sequence of refinements, we end up with a variety of candidate new LDPC codes, which we call *Silver codes*. We use our Silver codes to instantiate the code matrix in the silent OT extension protocol of [BCG+19a], which we also generalize to the setting of VOLE. We implemented Silver, our protocol for SILent Vole and oblivious transfER, using our most optimized code; our implementation is available at libOTe[Rin]. We compare Silver to the best existing OT extension protocols: the standard IKNP protocol [IKNP03], which remains to date the most efficient protocol in the "unlimited bandwidth" setting, the recent protocol of Yang et al. [YWL+20], which provides the best concrete performance in natural bandwidth settings (from 10Mbps to 5Gbps), and the silent OT extension protocol of Boyle et al. [BCG+19a], which is the most efficient protocol that enjoys the silent feature. When generating 10^7 OTs on one core of a standard laptop, our protocol requires only 300ms of computation and 122KB of communication. In comparison, IKNP requires 58% more computation and $\sim 1300\times$ more communication, [YWL+20] requires $\sim 4\times$ more computation and $\sim 14\times$ more communication, and [BCG+19a] requires $19\times$ more computation (since our protocol is essentially their SOT with a Silver code, the communication is identical). In a setting with 100Mbps of bandwidth, Silver is at least 50 times more efficient than IKNP even when ignoring all costs beyond those of communication, and at least $4\times$ and $19\times$ more efficient than [YWL+20] and [BCG+19a] respectively, even when ignoring all communication costs.

2 Preliminaries

Throughout the work we will using $[a, b]$ to denote the set $\{a, ..., b\}$. $[n]$ is shorthand for $[1, n]$. $=$ will denote mathematical equality while $x := y$ denotes defining x to be equal to y. $|\mathbf{v}|$ denotes the Hamming weight of vector $\mathbf{v}$. Matrix and vector horizontal concatenation is denoted as $[X|Y]$. Due to space restriction, we defer preliminaries on the silent OT extension protocol of [BCG+19a] to Appendix A of the Supplementary Material.

2.1 Preliminaries on Bias

Definition 1 (Bias of a Distribution). *Given a distribution $\mathcal{D}$ over $\mathbb{F}^n$ and a vector $\mathbf{u} \in \mathbb{F}^n$, the bias of $\mathcal{D}$ with respect to $\mathbf{u}$, denoted $\mathsf{bias}_{\mathbf{u}}(\mathcal{D})$, is equal to*

$$\mathsf{bias}_{\mathbf{u}}(\mathcal{D}) = |\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{u}^\intercal \cdot \mathbf{x}] - \mathbb{E}_{\mathbf{x}\sim\mathcal{U}_n}[\mathbf{u}^\intercal \cdot \mathbf{x}]| = \left|\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{u}^\intercal \cdot \mathbf{x}] - \frac{1}{|\mathbb{F}|}\right|,$$

where $\mathcal{U}_n$ denotes the uniform distribution over $\mathbb{F}^n$. The bias of $\mathcal{D}$, denoted $\mathsf{bias}(\mathcal{D})$, is the maximum bias of $\mathcal{D}$ with respect to any nonzero vector $\mathbf{u}$.

Given t distributions $(\mathcal{D}_1, \cdots, \mathcal{D}_t)$ over $\mathbb{F}_2^n$, we denote by $\bigoplus_{i\leq t} \mathcal{D}_i$ the distribution obtained by *independently* sampling $\mathbf{v}_i \xleftarrow{\$} \mathcal{D}_i$ for $i = 1$ to t and outputting $\mathbf{v} \leftarrow \mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_t$. We will use the following bias of the exclusive-or (cf. [Shp09]).

Lemma 2. *Let $t \in \mathbb{N}$ be an integer, and let $(\mathcal{D}_1, \cdots, \mathcal{D}_t)$ be t independent distributions over $\mathbb{F}_2^n$. Then $\mathsf{bias}(\bigoplus_{i\leq t} \mathcal{D}_i) \leq 2^{t-1} \cdot \prod_{i=1}^t \mathsf{bias}(\mathcal{D}_i) \leq \mathsf{min}_{i\leq t}\mathsf{bias}(\mathcal{D}_i)$.*

Eventually, let $\mathsf{Ber}_r(\mathbb{F}_2)$ denote the Bernoulli distribution that outputs 1 with probability r, and 0 otherwise. More generally, we denote by $\mathsf{Ber}_r(\mathbb{F})$ the distribution that outputs a uniformly random element of $\mathbb{F}$ with probability r, and 0 otherwise. We will use a standard simple lemma for computing the bias of a XOR of Bernoulli samples:

Lemma 3 (Piling-up lemma). *For any $0 < r < 1/2$ and any integer n, given n random variables $X_1, \cdots, X_n$ i.i.d. to $\mathsf{Ber}_r(\mathbb{F}_2)$, it holds that $\Pr[\bigoplus_{i=1}^n X_i = 0] = 1/2 + (1-2r)^n/2$.*

2.2 Syndrome Decoding and Learning Parity with Noise

Our constructions will rely on new variants of the learning parity with noise (LPN) assumption (more accurately, a variant of the syndrome decoding assumption). The LPN assumption over a field $\mathbb{F}$ states, informally, that no adversary can distinguish $(A, A \cdot \mathbf{s} + \mathbf{e})$ from $(A, \mathbf{b})$, where A is sampled from the set of generating matrices of some linear code ensemble, $\mathbf{s}$ is a uniform secret vector over $\mathbb{F}$, $\mathbf{e}$ is a *noise vector* sampled from some distribution over $\mathbb{F}$-vectors and typically sparse. $\mathbf{b}$ is a uniform vector over $\mathbb{F}$. More formally, we define the LPN assumption over a ring $\mathcal{R}$ with dimension k, number of samples n, w.r.t. a code generation algorithm $\mathbf{C}$, and a noise distribution $\mathcal{D}$:

Definition 4 (Primal LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{k,n}(\mathcal{R})\}_{k,n\in\mathbb{N}}$ denote a family of efficiently sampleable distributions over a ring $\mathcal{R}$, such that for any $k, n \in \mathbb{N}$, $\mathsf{Im}(\mathcal{D}_{k,n}(\mathcal{R})) \subseteq \mathcal{R}^n$. Let $\mathbf{C}$ be a probabilistic code generation algorithm such that $\mathbf{C}(k, n, \mathcal{R})$ outputs a matrix $A \in \mathcal{R}^{n\times k}$. For dimension $k = k(\lambda)$, number of samples (or block length) $n = n(\lambda)$, and ring $\mathcal{R} = \mathcal{R}(\lambda)$, the (primal) $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{LPN}(k, n)$ assumption states that*

$$\{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathbf{C}(k, n, \mathcal{R}), \mathbf{e} \xleftarrow{\$} \mathcal{D}_{k,n}(\mathcal{R}), \mathbf{s} \xleftarrow{\$} \mathbb{F}^k, \mathbf{b} \leftarrow A \cdot \mathbf{s} + \mathbf{e}\}$$
$$\overset{c}{\approx} \{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathbf{C}(k, n, \mathcal{R}), \mathbf{b} \xleftarrow{\$} \mathcal{R}^n\}.$$

The above definition is very general, and captures in particular not only the standard LPN assumption and its variants, but also assumptions such as LWE or the multivariate quadratic assumption. However, we will typically restrict our attention to assumptions where the noise distribution outputs sparse vectors with high probability. The standard LPN assumption with dimension k, noise rate r, and n samples is obtained by setting A to be a uniformly random matrix over $\mathbb{F}_2^{n\times k}$, and the noise distribution to be the Bernoulli distribution $\mathsf{Ber}_r^n(\mathbb{F}_2)$, where each coordinate of $\mathbf{e}$ is independently

set to 1 with probability r and to 0 with probability $1-r$. The term "primal" in the above definition comes from the fact that the assumption can come in two equivalent form: the primal form as above, but also a *dual form*: viewing A as the transpose of the parity check matrix H of a linear code generated by G a matrix, i.e. $A = H^\intercal$, the hardness of distinguishing $H^\intercal \cdot \mathbf{x} + \mathbf{e}$ from random is equivalent to the hardness of distinguishing $G \cdot (H^\intercal \cdot \mathbf{x} + \mathbf{e}) = G \cdot \mathbf{e} = \mathbf{e} \cdot G^\intercal$ from random (since $G^\intercal \cdot H = 0$).

3 On the Hardness of LPN for Structured LDPC Codes

The learning parity with noise assumption is one of the most fundamental assumptions of cryptography, introduced in the work of [BFKL94]; related problems were used even earlier [McE78]. The hardness of syndrome decoding and its variants (which is equivalent to LPN under our definition – see above) has also been intensely studied in coding theory, starting with the seminal work of Prange [Pra62] (under the name the of syndrome decoding), in learning theory (see e.g. [FGKP09] and references therein), and in random CSP theory (starting with the seminal work of Feige [Fei02]) – all with many follow ups.

Over the past few decades, a tremendous number of attacks against LPN have been proposed. These attacks include, but are not limited to, attacks based on Gaussian elimination and the BKW algorithm [BKW00, Lyu05, LF06, EKM17] and variants based on covering codes [ZJW16, BV16, BTV16, GJL20], information set decoding attacks [Pra62, Ste88, FS09, BLP11, MMT11, BJMM12, MO15, EKM17, BM18], statistical decoding attacks [AJ01, FKI06, Ove06, DAT17], generalized birthday attacks [Wag02, Kir11], linearization attacks [BM97, Saa07], attacks based on finding low weight code vectors [Zic17], or on finding correlations with low-degree polynomials [ABG+14, BR17].

A Unified Framework for Attacks against LPN. In light of this situation, it would be excessively cumbersome, when introducing a new variant of LPN, to go over the entire literature of existing attacks and analyze their potential impact on the new variant. The crucial observation, however, is that this is not necessary, as *all the above attacks* (and more generally, essentially all known attacks against LPN and its variants) fit in a common framework, usually denoted the *linear test framework*. Furthermore, the asymptotic resistance of any LPN variant against any attack from the linear test framework can be deduced from two simple properties of the underlying code ensemble and noise distribution. Informally, if

- the code generated by G has high minimum distance, and
- for any large enough subset S of coordinates, with high probability over the choice of $\mathbf{e} \leftarrow \mathcal{D}$, at least one of the coordinates in S of $\mathbf{e}$ will be nonzero,

then the LPN assumption with code matrix G and noise distribution $\mathcal{D}$ cannot be broken by any attack from the linear test framework. We will formalize this and build on it to analyze the asymptotic security of our new LPN variants.

We stress that this crucial observation is not new to our work: a similar observation was explicitly made in previous works [ADI+17, BCG+20], where it was also used to analyze the security of new LPN variants. Even long before these works, distributions whose outputs look random to linear tests, called *low-bias sample spaces*, have been the subject of a rich and fruitful line of work which was initiated in the seminal work of Naor and Naor [NN90], and the relevance of linear tests to the security analysis LPN assumptions seems to have been at least somewhat folklore. Still, we believe that it will be beneficial and instructive to the reader to present this argument in a unified way with explicit bounds.

3.1 The Linear Test Framework

The common feature of essentially all known attacks against LPN and its variants is that the distinguisher can be implemented as a (nonzero) *linear function of the samples* (the linear test), where the coefficients of the linear combination can depend arbitrarily on the code matrix. Therefore, all these attacks can be formulated as distinguishing LPN samples from random samples by checking whether the output of some linear test (with coefficients depending arbitrarily on the code matrix) is biased away from the uniform distribution. Formally,

Definition 5 (Security against Linear Test). *Let $\mathbb{F}$ be an arbitrary finite field, and let $\mathcal{D} = \{\mathcal{D}_{m,n}\}_{m,n\in\mathbb{N}}$ denote a family of noise distributions over $\mathbb{F}^n$. Let $\mathbf{C}$ be a probabilistic code generation algorithm such that $\mathbf{C}(m,n)$ outputs a matrix $A \in \mathbb{F}^{n\times m}$. Let $\varepsilon, \delta : \mathbb{N} \mapsto [0,1]$ be two functions. We say that the $(\mathcal{D}, \mathbf{C}, \mathbb{F})$-$\mathsf{LPN}(m,n)$ assumption with dimension $m = m(\lambda)$ and $n = n(\lambda)$ samples is* (ε,δ)-secure against linear tests *if for any (possibly inefficient) adversary $\mathcal{A}$ which, on input a matrix $A \in \mathbb{F}^{n\times m}$, outputs a nonzero $\mathbf{v} \in \mathbb{F}^n$, it holds that*

$$\Pr[A \xleftarrow{\$} \mathbf{C}(m,n), \mathbf{v} \xleftarrow{\$} \mathcal{A}(A) \ : \ \mathsf{bias}_{\mathbf{v}}(\mathcal{D}_A) \geq \varepsilon(\lambda)] \leq \delta(\lambda),$$

where $\mathcal{D}_A$ denotes the distribution induced by sampling $\mathbf{s} \xleftarrow{\$} \mathbb{F}_2^m$, $\mathbf{e} \leftarrow \mathcal{D}_{m,n}$, and outputting the LPN samples $A \cdot \mathbf{s} + \mathbf{e}$.

The following observation is folklore, and was made explicitly e.g. in [BCG+20]:

Observation 1 *Existing attacks against LPN (as listed above) can be cast as instances of the linear test framework. Therefore, none of these attacks can provide a polynomial-time distinguisher against any LPN assumption that is provably (ε,δ)-secure against linear tests, for any negligible functions (ε,δ).*

[ADI+17] went even further and explicitly conjectured that for any LPN variant with a sparse code matrix, the runtime of the best possible attack against LPN is essentially $\mathsf{poly}(1/\varepsilon)$, *i.e.*, the number of times a linear test attack must be repeated until the bias becomes noticeable. See Assumption 1.

3.2 Dual Distance and Security against Linear Tests

Following [ADI+17], we call *dual distance* of a matrix M, and write $\mathsf{dd}(M)$, the largest integer d such that every subset of d rows of M is linearly independent. The name "dual distance" stems from the fact that the $\mathsf{dd}(M)$ is also the minimum distance of the dual of the code generated by M (*i.e.*, the code generated by the left null space of M). The following lemma is folklore:

Lemma 6. *Let $\mathcal{D} = \{\mathcal{D}_{m,n}\}_{m,n\in\mathbb{N}}$ denote a family of noise distributions over $\mathbb{F}^n$. Let $\mathbf{C}$ be a probabilistic code generation algorithm s.t. $\mathbf{C}(m,n) \to A \in \mathbb{F}^{n\times m}$. Then for any $d \in \mathbb{N}$, the $(\mathcal{D}, \mathbf{C}, \mathbb{F})$-$\mathsf{LPN}(m,n)$ assumption with dimension $m = m(\lambda)$ and $n = n(\lambda)$ samples is $(\varepsilon_d, \delta_d)$-secure against linear tests, where*

$$\varepsilon_d = \max_{|\mathbf{v}|>d} \mathsf{bias}_{\mathbf{v}}(\mathcal{D}_{m,n}), \qquad \text{and} \qquad \delta_d = \Pr_{A\xleftarrow{\$}\mathbf{C}(m,n)}[\mathsf{dd}(A) \geq d].$$

Proof. The proof is straightforward: fix any integer d. Then with probability at least δ_d, $\mathsf{dd}(A) \geq d$. Consider any (possibly unbounded) adversary $\mathcal{A}$ outputting $\mathbf{v}$. Two cases can occur:

- Either $|\mathbf{v}| \leq d \leq \mathsf{dd}(A)$. In this case, the bias with respect to $\mathbf{v}$ of the distribution $\{A\cdot\mathbf{s} \mid \mathbf{s} \xleftarrow{\$} \mathbb{F}^m\}$ is 0 (since this distribution is d-wise independent). Since the bias of the XOR of two distribution is at most the smallest bias among them (see Lemma 2; the same holds for the bias with respect to any fixed $\mathbf{v}$), we get $\mathsf{bias}(\mathcal{D}_A) = 0$.
- Or $|\mathbf{v}| > d$; in which case, applying Lemma 2 again, $\mathsf{bias}(\mathcal{D}_A) \leq \mathsf{bias}(\mathcal{D}_{m,n})$.

Security of LPN with random codes. An instructive example is to consider the case of LPN with a uniformly random code matrix over $\mathbb{F}_2$, and a Bernoulli noise distribution $\mathcal{D}_{m,n} = \mathsf{Ber}_r^n(\mathbb{F}_2)$, for some noise rate r. The probability that d random vectors over $\mathbb{F}_2^m$ are linearly independent is at least

$$\prod_{i=0}^{d-1} \frac{2^m - 2^i}{2^m} \geq (1 - 2^{d-1-m})^d \geq 1 - 2^{2d-m}.$$

Therefore, by a union bound, the probability that a random matrix $A \xleftarrow{\$} \mathbb{F}_2^{n\times m}$ satisfies $\mathsf{dd}(A) \geq d$ is at least $1 - \binom{n}{d} \cdot 2^{2d-m} \geq 1 - 2^{(2+\log n)d-m}$. On the other hand, for any d and any $\mathbf{v}$ with $|\mathbf{v}| > d$, we have by Lemma 3:

$$\Pr[\mathbf{e} \leftarrow \mathsf{Ber}_r^n(\mathbb{F}_2) \ : \ \mathbf{v}^\intercal \cdot \mathbf{e} = 1] = \frac{1 - (1-2r)^d}{2},$$

hence $\mathsf{bias}_{\mathbf{v}}(\mathsf{Ber}_r^n(\mathbb{F}_2)) = (1-2r)^d \leq e^{-2rd}$. In particular, setting $d = O(m/\log n)$ suffices to guarantee that with probability at least $\delta_d = 1 - 2^{-O(m)}$, the LPN samples will have bias (with respect to any possible nonzero vector $\mathbf{v}$) ε_d at most $e^{-O(rm/\log n)}$. Hence, any attack that fits in the linear test framework against the standard LPN assumption with dimension m and noise rate r requires of the order of $e^{O(rm/\log n)}$ iterations. Note that this lower bound still leaves a gap with respect to the best known linear attacks, which require time of the order of $e^{O(rm)}$, $e^{O(rm/\log\log m)}$, and $e^{O(rm/\log m)}$ when $n = O(m)$, $n = \mathsf{poly}(m)$, and $n = 2^{O(m/\log m)}$ respectively [BKW00, Lyu05, EKM17].

3.3 SOT from Asymptotically Good Linear-Time Encodable Codes

Abstracting out the unnecessary details, recall that the construction of silent oblivious transfer extension introduced in [BCG+19b, BCG+19a] and recalled in Appendix A, relies on the following assumption: given a large public matrix $G \in F_2^{k\times n}$, is such that $n = c \cdot k$ for some small constant $c > 1$ (e.g. $c = 2$), it should be infeasible to distinguish $\mathbf{e} \cdot G^\intercal$ from random, where $\mathbf{e}$ is a uniformly random weight-t vector. This corresponds to the dual-LPN assumption, which is equivalent to the primal-LPN assumption with matrix $H \in F_2^{m\times n}$, where H is the parity check for generator G; *i.e.*, $G^\intercal \cdot H = 0$.

A selection principle for LPN with structured code. Based on the previous discussions, for any linear code ensemble $\mathbf{C}$ which outputs matrices $H \xleftarrow{\$} \mathbf{C}$ having a large distance w.h.p., it is reasonable to conjecture that the corresponding primal-LPN assumption will hold (since a contradiction would imply a fundamentally different type of attack than existing ones). This conjecture was formally stated in [ADI+17] for the case of all *sparse* code ensembles:

Assumption 1 (Assumption 6 in [ADI+17]) *For every prime-order field $\mathbb{F}$, every polynomial $m(\lambda), n(\lambda)$, every constant t, every real $r \in (0, 1/2)$, and every t-sparse matrix $A \in \mathbb{F}^{n\times m}$, the following holds: Any circuit of size $T = \exp(\Omega_r(\mathsf{dd}(A)))$ cannot distinguish $(A \cdot \mathbf{s} + \mathbf{e})$ for $\mathbf{s} \xleftarrow{\$} \mathbb{F}_2^m, \mathbf{e} \leftarrow \mathsf{Ber}_r(\mathbb{F})$ from the uniform distribution with advantage better than $1/T$ ($\Omega_r(x)$ denotes $\Omega(x)$, where the hidden constant may depend on the noise rate r).*

Noise weight versus minimum distance. The above discussions allows to make a simple, yet powerful observation: for typical noise distributions, including the Bernoulli distribution with parameter t/n, the *regular* noise distribution (concatenations of t length-n/t unit vectors), and the *exact* noise distribution (random t-sparse vectors), the running time of linear attacks is lower bounded by a term of the form $e^{c\cdot rd}$ for some constant c, where $r = t/n$ is the noise rate and d is the minimum distance. This suggests the following *safeguard*: if a SOT code exhibits a much worse typical minimum distance behavior than estimated (which in our case would be very surprising but theoretically possible), say, the true distance d is v times shorter than estimated, then same conjectured security level as before can be obtained by *scaling the number of noisy coordinates t by a factor v*. Crucially, in our SOT construction, the impact of this scaling vanishes when the number of OTs is large: it only impacts the complexity of distributing the seed (which increases by a factor v), but has no influence whatsoever, neither on the matrix multiplication part (which is the bulk of the computation) nor on the sparse vector expansion (which is the only other component whose cost scales with the target number of OTs).

Our approach: structured LDPC codes. Asymptotically good families of linear-time encodable codes have been studied in the literature, with probabilistic constructions given in [GDP73, Spi96]. However, these works only targeted asymptotic efficiency. Our aim, on the other hand, is to focus on concrete efficiency, and to find codes with a large concrete minimum distance, and extremely efficient encoding. We choose to focus on structured families of LDPC codes (*i.e.*, codes whose parity-check matrix is sparse), which have been widely studied in the coding theory literature. Our rationale is based on the following observations:

- Most LDPC codes have linear minimum distance;
- Some structured families of LDPC codes admit efficient encoding algorithms;
- Some structured families of LDPC codes provably achieve both fast linear time encoding and *almost* linear minimum distance;

– Structured families of LDPC codes in the literature which do *not* exhibit linear or close-to-linear minimum distance typically satisfy a specific constraint: their Tanner graph contains a large number of degree-2 variable nodes. In contrast, we suggest candidates which admit extremely fast encoding, but do *not* exhibit this structural weakness, and can be experimentally verified to exhibit a very good minimum distance growth.
– For *random* LDPC codes, the corresponding assumption (primal LPN with a random sparse code matrix) is the Alekhnovich assumption [Ale03], an important and well-studied assumption.

3.4 Most Sparse Matrices Have Linear Dual Distance

In this section, we show that for any integer $t > 2$, most matrices in $\mathbb{F}_2^{n\times m}$ with rows of Hamming weight t have dual distance linear in m; more precisely, the fraction of such matrices with dual distance at least $\gamma \cdot k$ (for some constant γ) is at least $1 - m^{2.1-t}$. In coding-theoretic terms, it says that most *column-regular* LDPC codes have linear minimum distance, where the parity check matrix has fixed column weight. Let $\mathsf{W}_t(\mathbb{F}_2^m)$ denote the set of all vectors in $\mathbb{F}_2^m$ with Hamming weight exactly t; we also denote by $\mathsf{W}_t(\mathbb{F}_2^{n\times m})$ the set of all matrices in $\mathbb{F}_2^{n\times m}$ with exactly t ones per column.

Theorem 7 (Most sparse matrices have dual distance $O(m)$). *For any constant $c > 1$ and integer $t > 2$, there is a constant $\gamma = \gamma(c,t)$ such that for any large enough m, denoting $n = c \cdot m$,*

$$\Pr\left[A \xleftarrow{\$} \mathsf{W}_t(\mathbb{F}_2^{n\times m}) \ : \ \frac{\mathsf{dd}(A)}{m} < \gamma\right] \leq m^{2.1-t}.$$

For completeness, we provide the proof in Appendix C of the Supplementary Material; the proof is a direct adaptation to our setting of the analysis of [MST03, Section 5.3]. Building upon our analysis, we also make a key observation: random LDPC codes over *large fields* have linear minimum distance with high probability, *even when their parity-check matrix is randomly sampled with* $\{0,1\}$ *entries*. We discuss the implications of this observation for one of our applications, as well as its relation to previous assumptions from the literature, in Appendix C.

4 Fast LDPC Encoding

We begin with an overview of how to perform fast encoding of LDPC codes by leveraging the sparsity/structure of H. Let us first review the naive encoding method. Recall H defines the code $\mathcal{C} = \{\mathbf{c} \mid H\mathbf{c}^\intercal = 0\}$. As such, define the systematic form H' for the same code by performing elementary row operations on H to obtain $H' = [-P^T | I_{n-k}]$. Since elementary row operations do not change the null-space, we have $\mathcal{C} = \{\mathbf{c} \mid H\mathbf{c}^\intercal = 0\} = \{\mathbf{c} \mid H'\mathbf{c}^\intercal = 0\}$. Although H is sparse, $P \in \mathbb{F}^{k\times m}$ is likely dense. Let $G := [I_k | P]$ be the symmetric form generator and then encoding can be achieved by computing $\mathbf{c} := \mathbf{x}G$ for $\mathbf{x} \in \mathbb{F}^k$. The cost of this is $O(n^3)$ time to compute P and $O(n^2)$ time to compute $\mathbf{x}P$.

However, we can also use the fact that $H\mathbf{c}^\intercal = 0$ to encode $\mathbf{x}$ into $\mathbf{c}$. Recall that $c = \mathbf{x}G = [\mathbf{x}|\mathbf{c}']$ for $\mathbf{c}' := \mathbf{x}P$. Therefore we can rewrite this as

$$0 = H\mathbf{c}^\intercal = H[\mathbf{x}|\mathbf{c}']^\intercal = T\mathbf{x}^\intercal + S\mathbf{c}'^\intercal \qquad \Longleftrightarrow \qquad -T\mathbf{x}^\intercal = S\mathbf{c}'^\intercal$$

where $H = [T|S]$ and $T \in \mathbb{F}^{m\times k}, S \in \mathbb{F}^{m\times m}$. Given $\mathbf{x}$, we can compute $\mathbf{y} := -T\mathbf{x}^\intercal$ in $O(k)$ time since T is sparse. We then solve the sparse system $\mathbf{y} = S\mathbf{c}'^\intercal$. Using Gaussian elimination, this would naively require $O(m^3) = O(n^3)$ time. However, we can try to leverage the sparsity of H to achieve better efficiency.

Our starting point is the somewhat standard LDPC solving technique known as g-Approximate Lower Triangularization (g-ALT) [RU01, DP15, KS12]. The basic intuition is that this system can be solved in linear time if S is a lower triangular matrix. In particular, the entries along the diagonal should all be set to one. Later we will discuss how to ensure this is the case. The system can be solved by solving each row "independently" starting with row 1 and working down. This idea can be generalized to allow all but the last g rows of H to be triangular (see Figure 9 in Appendix D). The last g rows are said to be part of the *gap*. As discussed in Appendix D, a parity check matrix with this form allows for encoding $\mathbf{x}$ as $\mathbf{c} = \mathbf{x}G$ in $O(n + g^2)$ time. Therefore, this remains linear so long

as $g = O(\sqrt{n})$. Additionally, we present an optimization in Appendix E which reduces this to $O(n)$ at the expense of $O(g)$ communication in the protocol.

Recall that in dual LPN we wish to compute $\mathbf{u} := \mathbf{e} \cdot G^\intercal$ which is equivalent to primal LPN where $\mathbf{u} := \mathbf{x} \cdot H + \mathbf{e}$. Yet, the encoding algorithm described above is for computing $\mathbf{x} \cdot G$. By the transposition principle [Bor57, IKOS08], we can achieve our goal at effectively the same cost. Roughly, the transformation works by first expressing the circuit which computes $\mathbf{x} \cdot G$ as a series of matrix multiplication, $\mathbf{s}_1 := M_1 \cdot \mathbf{x}, \mathbf{s}_2 := M_2 \cdot \mathbf{s}_1, ..., \mathbf{e} := M_n \cdot \mathbf{s}_{n-1}$ such that $\mathbf{e}$ is the final output. Any circuit can be expressed in this way. Then $\mathbf{e} \cdot G^\intercal$ can be computed as $\mathbf{s}_{n-1} := M_n^\intercal \cdot \mathbf{e}, \mathbf{s}_{n-2} := M_{n-1}^\intercal \cdot \mathbf{s}_{n-1}, ..., \mathbf{x} := M_1^\intercal \cdot \mathbf{s}_1$. Refer to Appendix D for a detailed description of all the algorithms discussed in this section.

5 Estimating the Minimum Distance Empirically

Crucial to our construction is the ability to accurately determine the minimum distance of the LDPC matrix H that is employed. Computing the exact minimum distance is known to be NP-Complete [Var97] and typically infeasible for our parameter region, e.g. $n = 2^{20}$. For some LDPC distributions, it is possible to derive an asymptotic bound on the minimum distance; however, many of these have drawbacks in efficiency or distance.

To overcome this, we resort to computational approaches for estimating the minimum distance of an LDPC code ensemble. For relatively small values of n, say less than 200, we compute the exact minimum distance using the approach presented in [HIQO19]. For larger n, say less than 4000, we fall back to a standard heuristic, the noisy impulse method of [BVJD02], for upper bounding the minimum distance (which we have verified does in fact closely agree with exact minimum distance for smaller values of n). We then extrapolate the asymptotic behavior of the minimum distance for larger values of n.

5.1 Exact Minimum Distance

For computing the exact minimum distance we make use of the so-called Brouwer-Zimmerman algorithm as described in [Gra06], and implemented in [HIQO19]. Loosely speaking, this approach iteratively refines a lower and upper bound until they are equal. First, the generator matrix G is placed in systematic form $G' = [I_k|P]$. Recall that for all $\mathbf{x} \in \mathbb{F}^k \setminus \{0\}$, the corresponding codeword is $\mathbf{c} = [\mathbf{x}|\mathbf{x}P]$ and therefore clearly $|\mathbf{c}| \geq |\mathbf{x}|$. Using this observation, the algorithm proceeds by initializing the lower bound $\ell = 1$ and upper bound $u = m + 1$. All x with $|\mathbf{x}| = \ell$ are encoded as $\mathbf{c} = \mathbf{x}G$ and the upper bound u is replaced as the minimum weight over all codewords considered. While $u \neq \ell$, ℓ is incremented and the process is repeated. See [Gra06, HIQO19] for details.

The running time remains exponential in the size of the code. With careful optimizations, the implementation of [HIQO19] is capable of computing the minimum distance up to about $n = 160$. Since this is relatively small compared to the codes our protocol employs, this approach is primarily used to validate the accuracy of the so-called *noisy impulse method* which we describe next.

5.2 Upper Bounding the Minimum Distance

Our second approach for evaluating the minimum distance of a LDPC family is known as the noise impulse method [BVJD02]. Very roughly speaking, this approach tries to decode the zero codeword when one or more of the bits have been flipped. The intuition is that if the right bits are flipped, then the next closest codeword will correspond to a close-to-minimum weight codeword.

In more details, and including improvements from [XFE04], this approach considers all vectors $\mathbf{c} \in \{0,1\}^n$ with $|\mathbf{c}| \leq w$ for some small constant w, e.g. 1 or 2. Each $\mathbf{c}$ is input into a belief propagation decoder, typically Min-Sum, which output the decoders estimates on the likelihood that each bit of $\mathbf{c}$ should be error corrected to zero or one. Since at most w bits in $\mathbf{c}$ are one and the actually minimum distance d is almost certainly more than twice w, the most likely codeword will in fact be the original all zero codeword.

However, the likelihood information contained in the decoder output can be leveraged to aid in the search of nearby non-zero codewords. Loosely speaking, belief propagation (BP) decoders work by assigning each bit of $\mathbf{c}$ a likelihood of being zero or one and updating these likelihoods in an

iterative process. The initial likelihood values could be that the decoder is 95% certain that each bit is as specified by $\mathbf{c}$, i.e. an error rate of 0.05. Intuitively, at each iteration the likelihood information for each bit of $\mathbf{c}$ is updated based on how many of the corresponding parity checks pass or fail. An interpretation of this is that it reduces the likelihood values for the zero positions of $\mathbf{c}$ when they are closely related to the positions of $\mathbf{c}$ which were set to one.

The idea is then to sort the positions of $\mathbf{c}$ such that the positions which are most confidently zero are to the right. The same permutation is applied to the columns of G. Partial Gaussian elimination is applied to G s.t. the left $k \times k$ submatrix is lower triangular with ones along the diagonal. Some of the first k columns are likely linearly dependent, preventing us from making the left k columns of G lower diagonal. In this case, the dependent columns are permuted right and replaced with the next left most column of G. We then consider all (permuted) codewords with the form $\mathbf{c}^* = [\mathbf{c}_1|\mathbf{c}_2|0^{n-k-t}]$ where $\mathbf{c}_2 \in \{0,1\}^t$ has some maximum weight u, e.g. $t = 50, u = 10$. For each choice of $\mathbf{c}_2$, it is possible to compute $\mathbf{c}_1 \in \{0,1\}^k$ via the left lower diagonal submatrix of G. The estimated upper bound the distance as the minimum weight over all $\mathbf{c}^*$.

Table 1: The minimum d_{min}, average d_{avg} and maximum d_{max} minimum distance obtained over 100 trials for weight 5 uniform LDPC codes.

m	method	w	d_{min}	d_{avg}	d_{max}
20	impulse	1	2	5	6
	exact	-	2	5	6
40	impulse	1	4	8.3	10
	exact	-	4	8.3	10
60	impulse	1	8	11.44	14
	impulse	2	8	11.38	14
	exact	-	8	11.38	14
80	impulse	1	12	16.48	20
	impulse	2	12	14.86	18
	exact	-	12	14.86	18
100	impulse	1	16	22.28	26
	impulse	2	14	18.2	20

6 Code Design

Designing an efficient LDPC code for large dimension LPN offers many unique challenges. Our primary two design goals are to achieve large minimum distance, ideally linear in n, and linear time encoding. However, unlike many existing codes from the coding community, we do not care about its decoding performance or other error correcting properties. All our codes have rate $1/2$, i.e. $n/2 = m = k$.

In this section we review two existing LDPC codes, namely uniform and Tillich-Zémor Codes. After describing various benefits and drawbacks of each, we design a new highly efficient LDPC code which achieves an extremely fast linear encoding time and plausibly linear minimum distance.

6.1 Uniform LDPC

As described in Section 3.4, the family $\mathsf{W}_t(\mathbb{F}_2^{n\times m})$ of uniform LDPC codes with fixed column weight t are known to have linear minimum distance with good probability. We consider the family of codes parameters by $t \in \{5, 11\}$. While the theoretical bound applies to all $t > 2$, we observe that $t = 3$ experiences very poor concrete minimum distance performance and often do not correspond to a code that can be made systematic. For $t = 5$ we observe a concrete linear minimum distance growth rate of $d_{\mathsf{avg}} = 0.28m, d_{\mathsf{min}} = 0.19m$ over 100 trials. These growth rates were obtained for $n \in [200, 800]$. Since we are interested in the worse case performance, we are mostly interested in d_{min}. By increasing the weight to $t = 11$ we obtain a minimum distance growth rate of $d_{\mathsf{avg}} = 0.38m, d_{\mathsf{min}} = 0.36m$.

The hardness of syndrome decoding for uniform LDPC codes was the basis of recent proposals [BCGI18, YWL+20], and corresponds to the well-established Alekhnovich assumption [Ale03]. While these codes turn out to be inappropriate efficiency-wise in our setting (see below), we will rely on the following heuristic to select the concrete parameters of our new codes: when we experimentally observe, with high confidence, that a distribution over codes achieves a similar average minimum distance, with a similar variance, compared to uniform LDPC codes, we heuristically estimate that the

corresponding assumptions should provide a comparable level of hardness. We note that, if it turns out that this heuristic is too optimistic (which would intuitively require finding new attacks radically different from all known attacks), increasing the noise (as described in Section 3.3, Noise weight versus minimum distance) can be used to adjust the hardness level of the underlying assumption without significantly harming efficiency.

Shortcoming of uniform LDPC codes. The choice of basing security on the hardness of decoding uniform LDPC codes was motivated in previous works [BCGI18, YWL+20] by the fact that they correspond to the relatively well-established Alekhnovich assumption. However, they turn out to be a relatively poor choice in our setting. At a high level, the reason is that for distributions over random LDPC codes which do not enforce any particular structure beyond guaranteeing some conditions on the number of ones per row and column (*i.e.*, which sample the parity-check matrix uniformly conditioned on constraints on the fractions of variable and check nodes from the Tanner graph which must have a given degree), having a high minimum distance with good probability, and being linear-time encodable with the g-approximate lower triangularization algorithm, appear to be at odd, according to a conjecture of Richardson and Urbanke [RU01] (we will discuss this conjecture in more details in Section 6.2). This effectively justifies moving towards *structured* ensembles of LDPC codes, which also enforce some structure on the *shape* of the parity-check matrix. A prime example of codes achieving a sweet spot between having high minimum distance with good probability, and very fast encoding, is given by the Tillich-Zémor code ensemble.

6.2 Tillich-Zémor Codes

As discussed before, in order to design codes that have efficient encoders as well as good minimum distance, one must move away from random codes and consider more structured codes. As such, structured codes would offer an immediate handle on efficient encoding purely by design. On the other hand, this approaches leaves much for the desire of a more rigorous theoretical understanding of the minimum distance of such structured codes. Such questions have been posed in the past amongst the members of the coding theory and communications communities. One such work is that of Tillich and Zémor [TZ06]. They investigate the minimum distance of structured LDPC codes with two variable nodes of degree-2 per parity-check equation. In the design of LDPC codes with high iterative decoding performance the variable nodes of degree-2 play a very important role, and this is their motivation for investigating the minimum distance of such codes. Concretely, they investigate codes with $m \times n$ parity check matrices of the form $H = [L|R]$, where R is the $m \times m$ matrix defined by

$$R = \begin{pmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & 1 \\ 1 & & & & & 1 \end{pmatrix}$$

and L is an $m \times k$ matrix such that all of its columns/rows have weight constant t. As such, H is in g-ALT form for $g = 1$. Tillich and Zémor prove that if H was generated at random subject to these structural constraints, then the minimum distance of the corresponding code is at most $\alpha n^{1-\frac{2}{t}}$ with probability $\mathcal{O}(n^{\frac{2}{t}-1}) + \mathcal{O}(\alpha^{\frac{t}{2}})$ for even t, and $\mathcal{O}(n^{\frac{2}{t}-1}) + \mathcal{O}(\alpha^{t})$ for odd t. In fact, they also show that for *any* H that has the aforementioned structure, the corresponding code will have minimum distance upper bounded by a quantity of order $\mathcal{O}(n^{1-\frac{1}{t}})$. Thus, such codes always have sub-linear distance.

Looking at the structure of H, there are a few observations we can make. Let n_2 denote the number of variable nodes of degree 2. A principal quantity of interest is the ratio n_2/m. It can be shown that if $n_2/m > 1$, then the minimum distance of the corresponding LDPC code cannot be larger than a logarithmic function of n. If $n_2/m < 1$, then it is possible for the minimum distance to be a linear function of n. The codes considered by Tillich and Zémor achieve $n_2/m = 1$ (for $t \neq 2$) and offer readily a simple linear-time encoding algorithm for the corresponding code. Yet, as mentioned before, these codes always have sub-linear (albiet, close to linear) minimum distance. In the next section we will empirically verify that the sub-linear growth is in fact the case.

Other works, e.g. [OTA07, DRU06], have looked at the the sub-graph $\mathcal{G}_2$ of the Tanner graph formed by only the degree-2 variable nodes (columns of H with weight 2) and certain structural properties can lead to poor minimum distance. For example, it is a common practice to ensure that

there are no cycles in the Tanner graph involving only variable nodes of degree 2. Also, Otmani, Tillich and Andriyanova [OTA07] proved that if $\mathcal{G}_2$ is slightly dense (has average degree greater than 2), then the minimum distance is only at most logarithmic in n. They also consider several other conditions for ensuring sub-linear distance.

Another work regarding $\mathcal{G}_2$ is that of Di, Richardson and Urbanke [RU01, DRU06] and regard the quantity $Q = \lambda'(0)\rho'(1)$ (λ and ρ are polynomials describing specific weight distributions of the rows/columns respectively) and how it impacts the minimum distance. $\lambda'(0)$ is the fraction of edges in the Tanner graph connecting to degree-2 variable nodes. They show that if $Q > 1$, then the minimum distance grows sub-linearly with n and linear time encoding. A question that is left open and remains to be answered is whether a linear encoding complexity necessarily implies sub-linear minimal distance.

We end this section with a few concluding remarks regarding our new codes and how they compare against the several techniques laid out in this section. Firstly, our codes have designed to ensure fast/linear encoding complexity while also having high (potentially linear) minimum distance. However, the approach to ensure linear encoding complexity is different from the works described in this section, since we actually have zero columns with weight 2. Thus, none the sufficient conditions for sub-linear minimal distance described above are satisfied by our codes. Based on these observations, we conclude that our codes do not provably have sub-linear minimal distance. We leave open the task of formally proving claims regarding the minimum distance of our codes.

6.3 LDPC Silver Codes

We now present our new LDPC constructions, which we dub Silver Codes (codes for SILent Vole and oblivious transfER). The goal of these codes is to obtain (plausible) linear minimum distance and extremely efficient encoding. Unlike in the traditional setting, our codes need to perform well (encoding-wise) when n is on the order of millions (but do not need to admit efficient decoding algorithms). Ideally our code would have a very compact representation. If a large preprocessing/sampling procedure must be performed, then the codes will likely need to be stored in memory, possibly requiring more memory than the rest of the protocol. Therefore we aim to design codes with a very succinct description.

Our second goal is to have a very efficient memory access structure. Recall that the encoding algorithm will have to access "memory locations" j and i whenever there is a 1 located at $H_{j,i}$. Therefore we would ideally like H to have some additional structure which maintains some memory locality. For example, having a bounded distance between sequential memory accesses. In the case of TZ codes, for example, the left matrix is uniformly distributed, which significantly harms the performances in terms of memory access. When n is on the order of millions, performing random access into an array of length n can quickly dominate the running time as we will see in Section 7.

Despite this shortcoming, we take TZ as our starting point and iteratively improve it (sacrificing decoding performance, but trying to optimize minimum distance and encoding time) with the (heuristic) guidance of our minimum distance estimators. It will be useful to partition H into left and right halves $[L|R] := H$ which are each of size $m \times m$. For TZ, L is therefore a uniform column/row weight t matrix while R has column/row weight 2 where all ones are effectively on a diagonal band. Recall that we only consider rate 1/2 codes where $k = m = n/2$.

It is also a well known phenomenon that odd column weight t LDPC codes achieve better minimum distance performance (for examples, the bounds on the minimum distance achieved in [TZ06] are much better for odd t). Hence, we restrict ourselves to odd values of t. In particular, we focus on $t \in \{5, 11\}$.

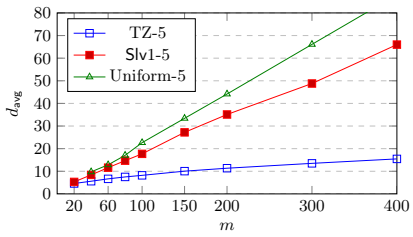
Slv1. Our first observations is that the structure of R in TZ plays a crucial role in the proof of sub-linear distance. For TZ, this structure was desirable as it enables a very efficient linear time encoder. However, using the more general g-ALT encoder we are still able to have linear time encoding for any $g = O(\sqrt{n})$. Our first alteration is then to increase the gap g and ensure all columns of R have weight t. There are several possible values for g and we experimentally settle on $g \in \{24, 32\}$ as they will provide good concrete performance.

The next question is how should the ones be distributed in R. Our g-ALT encoder require ones along the diagonal which leaves $t - 1$ degrees of freedom per column. While one could distribute these uniformly over the lower half of R, we opt to place them uniformly in the g positions below the main diagonal. An example of $g = 2, t = 2$ is shown in Figure 1a. We consider two choices of

these parameters, $(g, t) \in \{(24, 5), (32, 11)\}$, which are respectively used in our weight 5 and 11 codes. We note that other parameter choices are possible and that we settled on these as a good trade off between efficiency and distance.

$$\begin{pmatrix} 1 & 1 & 1 & & & & \\ & 0 & 1 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 0 & 0 & 1 & \\ 1 & & & & 1 & 1 & \\ 0 & 1 & & & & 0 & \end{pmatrix}$$

(a) Sample R matrix for Slv1 with $g = t = 2$.



(b) Average minimum distance of weight $t = 5$ codes.

Fig. 1: Example of R and distance of Slv1, uniform, TZ for weight $t = 5$.

We denote this code family as Slv1-t. Slv1 immediately gives a significant improvement over TZ as shown in Figure 1b. Consider the structure of minimum codewords in TZ: They are often composed of several columns from L which when added together result in small distances between the non-zero elements, e.g. 1001000...000100010 which has distances 3,4 between the ones. The small distances can then be "bridged" by including the corresponding columns of R, e.g. 7 columns in the case above. However, this strategy does not work for our codes due to R having larger column weight which are randomly distributed.

Moreover, this code performs remarkably similar to uniform of the same column weight $t = 5$. With $m = n/2 = 200$ rows, the average (estimated) minimum distance over 100 trials of this code is 35 while uniform is 45.

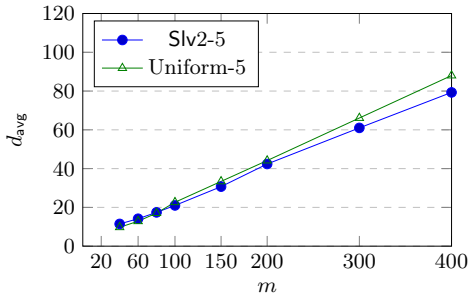
Although this code represents a significant improvement over TZ for our particular application, we observe that some samples of the Slv1 code have significantly lower distance that others. In particular, the variance in this code can result in samples with as low as $d_{\text{min}} \approx 0.55 d_{\text{avg}}$ while uniform has a much smaller variance, with $d_{\text{min}} \approx 0.95 d_{\text{avg}}$ over 100 samples.

Slv2. Through experimentation and inspecting the Slv1 instances which perform unusually poorly, we identified that key contributors are bad local structures in the main diagonal of R which can at times result in low weight codewords. To prevent this, we observed that adding additional weight one diagonals below the main diagonal prevents these structures. Intuitively they work by increasing the expanding property of each column by guaranteeing they span more than g rows. Moreover, these structures add almost no computational overhead.

Additionally, we remove the first g columns of R such that its a $m \times m - g$ matrix and the portion of the band which wraps around is removed. In Appendix E we will use a different technique to restore R to being square. An example of the Slv2 distribution of R is in Figure 2a with $g = t = 2$ and a single diagonal.

$$\begin{pmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 1 & 1 & 1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ 1 & & 0 & 1 & 1 & & \\ & \ddots & & 0 & 0 & 1 & \\ & & 1 & & 1 & 1 & \\ & & & 1 & & 0 & \end{pmatrix}$$

(a) Sample R matrix for Slv2 with $t = 2$.



(b) d_{avg} of column weight 5 codes.

Fig. 2: The alterations in Slv2 along with the minimum distance performance.

Through experimentation we observe that adding two weight 1 diagonal bands at distances 5 and 31 below the main diagonal significantly reduces the variance and improves the average distance. As shown in Figure 2b the performance of the second code which we denote as Slv2. We note that the

uniform code also had the n dimension reduced by g in order to maintain a fair comparison. Remarkably, the Slv2 code has average performance almost identical to that of uniform codes. Moreover, the variance of Slv2 is significantly reduced, with $d_{\mathsf{min}} = 0.88 d_{\mathsf{avg}}$ compared to $d_{\mathsf{min}} = 0.91 d_{\mathsf{avg}}$ for uniform over 100 trials.

Slv3. Next we turn our attention to the distribution of L after which we will further optimize R. While the current distribution of L gives good minimum distance, its memory locality properties are extremely poor since it is uniform. For each non-zero $L_{i,j}$, the g-ALT encoder must access two arrays at i, j respectively. This effectively means one of them is always a cache miss and can quickly dominate the running time as see in Figure 5 of Section 7.

We investigated numerous methods of improving the memory locality of L. For instances, an L consisting of random non-zero submatrixes with various dimensions. However, for the most part this line of thinking was ineffective. Core to a high performing L is an expanding property. In particular, each column of L should have non-zero locations which are somewhat unique and spread out. This is particularly important since the distribution of R is more or less a single band along the diagonal. If both L and R consists of clumps of ones, then it is more likely that cancellation can occur.

However, we identified a surprisingly simple and highly efficient structure which can possess the exact properties we desire. In particular, we will distribute L such that each column is a cyclic shift of exactly one over the previous. This effectively results in t weight one diagonals wrapping around L.

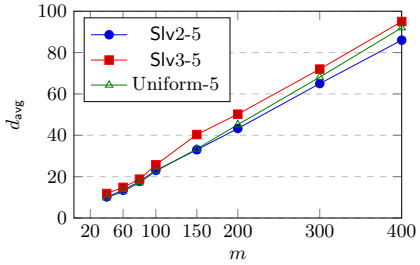
We observe that the exact distribution of the diagonal plays a very crucial role in the minimum distance performance of L. For instance, if they are sampled uniformly, then with some noticeable probability the diagonals can be clumped together. In these cases the code can perform extremely poorly due to L and R being too similarly distributed. One also might think that to achieve a good expanding property that distributing the diagonals evenly over L would be optimal. However, in this case it is possible for two columns of L to equal.

We have experimentally identified that a compromise between these two extremes achieves very good minimum distance performance (both in terms of average distance and variance). In particular, the diagonals should be somewhat evenly distributed while still being irregularly spaced. To identify such distributions we sampled many L at random and evaluate the resulting minimum distance over hundreds of trials and various values of n. An instances of a well performing L with weight $t = 5$ is to distribute the ones of the first column as $\{0m, 0.049m, 0.43m, 0.60m, 0.73m\}$. Other well performing instances have a similar distribution where some diagonals are relatively close while overall they are evenly distributed over the range.

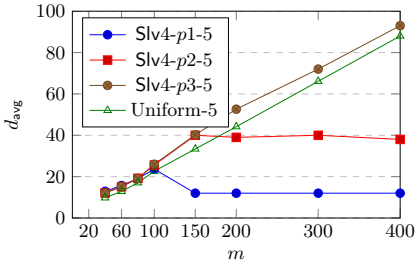
Our methodology for selecting the exact parameters was to evaluate 10,000 random choices at $m \in \{40, 60, 80, 100, 150, 200, 300, 400\}$ and select the top 100 best performing. Out of these, we then ran 100 trials for each $m \in [40, 400]$ with independently sampled R and selected the parameters which maximized $d_{\mathsf{min}}/d_{\mathsf{avg}}$ for each m. As such, our selection didn't achieve the highest average distance d_{avg} but instead was "consistently well performing." We note that one has to be careful with the selection of L as a poorly chosen one can result in bad/erratic minimum distance performance. That being said, we observed that most randomly sampled chooses performed well.

The minimum distance performance of this code is depicted in Figure 3a. Interestingly, this code out performs uniform with an average (estimated) minimum distance of $d_{\mathsf{avg}} = 94$ at $m = 400$ compared to $d_{\mathsf{avg}} = 91$ for uniform. Moreover, the variance of this code is quite low, with $d_{\mathsf{min}} = 0.94 d_{\mathsf{avg}}$ at $m = 400$ compared to $d_{\mathsf{min}} = 0.91 d_{\mathsf{avg}}$ for uniform over 100 trials.

Slv4. We now return our attention to improving the distribution of R. Generating R is effectively sampling $O(m)$ random sets of $\binom{g}{t}$, which correspond to the location of the ones on the main diagonal. While linear time, this sampling can be quite expensive. We therefore experiment with the idea of letting the diagonal repeat ever p columns. While one has to be careful with repeated structures in a code, for a sufficiently large p we conjecture and experimentally confirm that it should not harm the minimum distance. We consider a repeat of $p \in \{1, 2, 3, ..., g\}$ and observe the repeating structure only introduces a weakness for $p \in \{1, 2\}$. The case of of $p = 1$ is clearly problematic due to R now effectively being $t + 2$ diagonal lines of width one which structurally is too similar to L. Our experiments reflect this with minimum distances being effectively upper bounded by 12 as seen in Figure 4. For $p = 2$ we observe a similar trend with the distance being upper bounded by 40. However,



(a) Average minimum distance of Slv2, Slv3 vs uniform with $t = 5$.



(b) The negative effect on average minimum distance of having repeats of $p \in \{1, 2\}$ vs $p \geq 3$.

Fig. 3: Performance of Slv3 and Slv4 with $p \in \{1, 2, 3\}$ vs uniform.

for $p \geq 3$ we observe no negative effects over all of the trials. To be slightly conservative, we opt to set $p = g$ which for our weight 5 code results in $p = 24$.

We further propose selecting a concrete instance of the diagonal, and validating its performance on the range of experimentally testable values of n. This can in turn give us confidence that the repeating structure does not happen to correspond to a weak instances, e.g. a $p = 1$ instance. Moreover, by selecting a concrete instance, it is possible to hardcode the indices into the program and get a very significant performance improvement.

Slv5. This leads us to our final modification. For the case of $p = g$ we restrict our selection of R such that each row[3] and column has fixed weight $t - 1$ with respect to these random indices. The reason for this alteration is purely to improve the computational efficiency of computing $xG^{\intercal}$ via the transposed circuit. In particular, the encoding algorithm will process R in a row by row manner. This alteration allows the weight of each row to be not be hard coded improved the performance of the branch predictor, etc. We observe that restricting R to be row regular does not decrease the minimum distance performs. See Appendix F for a detailed description.

Eventually, we further consider a variant of Slv5, called Slv5'. This variant is entirely identical, with the sole exception that the parity-check matrix is now viewed as the parity-check matrix *over a field* $\mathbb{F}$ which might not be equal to $\mathbb{F}_2$ – while the parity-check matrix still has $\{0, 1\}$ entries. We do not use this variant in our main application to silent OT, but it can be used to provide strong efficiency improvements for VOLE over larger fields. We provide support for this modification in Appendix C of the Supplementary Material.

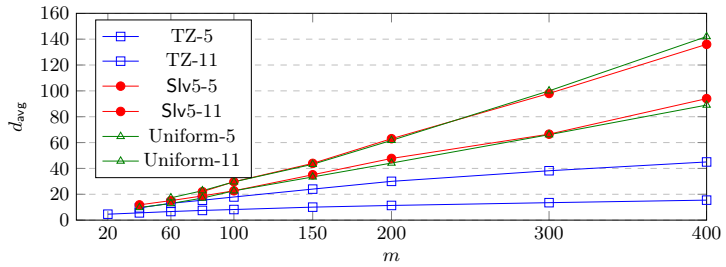


Fig. 4: Average minimum distance of uniform, Slv5 and TZ for weight $t \in \{5, 11\}$.

7 Performance Evaluation

We now evaluate the concrete running times of our LDPC codes along with our Silent OT and Vole implementations (available at [Rin]). With respect to our OT protocol we compare with [IKNP03, BCG+19a, YWL+20]. We also compare our Vole implementation (a direct generalization of [BCG+19a] with our LDPC code) with the implementation of [WYKW20]. All implementations target $\kappa = 128$ bits of computational security and $\lambda = 40$ bits of statistical security.

All performance evaluations were perform on a single consumer laptop with an i7 9750H CPU and 16GB of RAM. Networking is performed via localhost. Each party is restricted to a single thread. We

[3] Excluding edge cases for the first and last set of g rows.

note that due to silent property of our protocol, it is very conducive to a multi-threaded implementation but that we only consider single thread performing for simplicity. All numbers reported exclude a setup phase where 128 base OTs are perform.

LDPC Encoding Performance. In previous protocols for silent OT and Vole, the running time was dominated by the compression of the noisy vectors generated in the setup. We now compare our new algorithms with the bit polynomial multiplication encoding used in [BCG+19a].

For $n = 2^{20}$, our most optimized code is 31× faster than [BCG+19a, CCK+18]. This improved running time is not merely due to using an LDPC code as demonstrate by the running time of TZ, which is only between 1.1 and 2× faster than [BCG+19a, CCK+18]. Moreover, the initial strengthening of the TZ minimum distance by the Silv1 code results in a significant running time increase of 1.5×.

The first major performance improvement is achieved by the Silv3 code which changes the distribution of L to have an extremely efficient memory access structure. This change reduces the running time of the L encoding by around 25×. The Silv5 code then optimizes the distribution of R to have a repeating structure along with ensuring that it is row regular. These changes allow for very significant memory and system level optimization.

Encoder	weight t	n 2^{16}	2^{20}	2^{24}
[BCG+19a]	-	10.2	194.2	4180
TZ	5	4.5	77.4	1943
	11	4.9	145.2	3971
Silv1	5	7.3	153.2	3632
	11	7.9	241.7	5414
Silv3	5	5.8	88.2	1688
	11	6.1	97	1792
Silv5	5	0.2	6.3	134
	11	0.5	11.3	234

Fig. 5: Running times (ms) of encoding algorithms for LPN with n length vector.

Oblivious Transfer Performance. We now turn our attention to analysing the concrete performance of our OT protocol in comparison to [BCG+19a, YWL+20, IKNP03] as shown in Figure 6. All protocols output m instances of correlated OT where the receiver obtains a per instance bit b and message $m_b \in \{0,1\}^{128}$ while the sender obtains a global $\Delta \in \{0,1\}^{128}$ and a per instance message $m_0 \in \{0,1\}^{128}$ such that $m_b = m_0 + b\Delta$. Random and chosen message OTs can then be obtained via standard techniques. Our protocol is based on that of [BCG+19a] and we inherit their $O(\log m)$ communication overhead.

We observe that both our weight 5 and 11 Silv5 codes out perform *all* existing protocol in terms of computational overhead while matching the best communication overhead of [BCG+19a]. In particular, our protocol is as much as 1.5× faster than the highly optimized [IKNP03, Rin] protocol which has stood as the most computationally efficient protocol for almost two decades. All this is achieved while communicating exponentially less data. We argue that this is a landmark achievement given the central role OT plays in countless protocols.

The next most efficient protocol is that of Yang et al. [YWL+20] which also achieves a sub-linear (but not logarithmic) communication overhead. This protocol is based on Primal LPN and therefore requires a one time setup sub-protocol in which correlated randomness is constructed. Given this, their protocol can then generate correlated OTs on demand. In Figure 6 we distinguish their setup and online protocols as $x + y$ respectively. However, even if only the online protocol is considered, our protocol is more than 4× more efficient in terms of running time and communication. If their setup phase is included then our protocol requires 13× less communication for $m = 10^7$. What is more, their setup phase requires a relatively complicated parameter select procedure which limited us to only performing $m = 10^7$ OTs with their implementation. One reason their only implement this size is that their setup phase has a relatively fixed cost regardless of m. On the other hand, our protocol can easily be executed with any value of m with running times that scales proportionally.

		Time (ms) m				Comm (KB) m			
Protocol:	weight t:	2^{16}	2^{20}	10^7	2^{24}	2^{16}	2^{20}	10^7	2^{24}
[BCG+19a]	-	25	510	5,121	1,0432	**75**	**94**	**122**	**126**
This	5	**1**	**29**	**268**	**488**				
	11	2	33	324	591				
[YWL+20]	10	-	-	44+1,134	-	-	-	1,130+550	-
[IKNP03]	-	4	45	423	692	1,048	16,777	160,038	268,435

Fig. 6: Single thread running time (ms) and communication (KB) to perform m correlated oblivious transfers in the LAN setting.

Vole Performance. We implement the generalization of [BCG+19a] for performing vole. The protocol is largely the same as the OT variant except that f more OTs on strings of length $O(\kappa f)$ need to be performed where f is the log of the field size, i.e. $f = 128$. For our protocol we use the binary Slv5 code while the noise vector is distributed over the while field. The security of this optimization is discussed in Section C.2. We compare with the vole protocol of [WYKW20] (a generalization of [YWL+20]) which is based on Primal LPN and therefore requires a one time setup sub-protocol. We also compare to the 1-out-of-N OT protocol of [OOS17] due to vole also supporting this functionality via hashing.

We observe that our protocol significantly out performs both of these works. Moreover, [WYKW20] performs a vole over a field of size $2^{61}-1$ while our implementation is for the Galois field of size 2^{128}. As such, this effectively halves their communication. Similarly, [OOS17] has an analogous field size of 2^{79}. Despite working over a larger field, the running time of our protocol $4\times$ faster than [WYKW20] at $m = 4\times10^7$ and $22\times$ faster than [OOS17]. Similarly, at $m = 4\times10^7$ our protocols requires between 5 to $8\times$ less communication than [WYKW20] depending on if their setup is include and $6200\times$ less than [OOS17].

			Time (ms) m				Comm (KB) m			
Protocol:	weight t:	$\log \lvert\mathbb{F}\rvert$	2^{16}	2^{20}	2^{24}	$4\cdot10^7$	2^{16}	2^{20}	2^{24}	$4\cdot10^7$
This	**5**	128	**10**	**50**	**616**	**1,390**	**339**	**373**	**405**	**409**
	11	128	11	53	750	1,660				
[WYKW20]	10	61	-	-	-	260+5,699	-	-	-	1,130+2,101
[OOS17]	-	79	15	218	3,499	31,219	4,219	67,137	1,073,832	2,561,551

Fig. 7: Single thread running time (ms) and communication (KB) to perform m voles (or 1-out-of-N OTs for [OOS17]) in the LAN setting.

Applications. The applications of our new protocol are extremely broad. Two of the most compelling are binary triple generation for the GMW[GMW87b] protocol and private set intersection. The former allows generic secure computation of binary circuit at the expense of performing $2|C|$ OTs and sending $2|C|$ bits where $|C|$ is the number of AND gates in the circuit. Due to the extreme efficiency of our protocol, the cost of the OTs is like dominated by the other costs in the GMW protocol, i.e. simply sending the bits. More generally, since our OT protocol is faster than all prior works in effectively all metrics, our protocol should be the de facto choice for generating OTs and binary triples.

The recent semi-hones/malicious secure PSI protocol of [RS21] directly builds on vole and achieved the lowest communication and very fast running times compared to all prior works. This protocol performs a vole of size $2.4n$ where the sets are of size n. Their implementation makes use of the vole protocol of [SGRR19] along with optimizations of [WYKW20]. As such, integrating our vole protocol gives a good example of the speed ups that can be obtained.

For sets of size $n = 2^{16}$ to $n = 2^{24}$ we observe that our vole protocol improves the running time of [RS21] by between 40 and 45 percent and 25 to 1 percent reduction in communication. Concretely, the semi-honest variant of their PSI protocol for $n = 2^{20}$ with our vole implementation would require 3.1 seconds compared to 2.4 seconds of [KKRT16] while at the same time sending $2.5\times$ less data than

[KKRT16]. As such, in effectively all real world situation the PSI protocol of [RS21] with our vole is the optimal protocol to use. Moreover, the malicious variant of [RS21] with our vole achieves the fastest running time and lowest communication by a factor of 1.7× and 4× respectively compared to then next most efficient protocol[PRTY20].

Acknowledgement.

We thank Quang Dao for pointing out a typo in the statement of Theorem 7.

References

ABG+14. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC^0 o MOD_2. pages 251–260, 2014.

ADI+17. Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. pages 223–254, 2017.

AJ01. Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. 2001.

Ale03. Michael Alekhnovich. More on average case vs approximation complexity. pages 298–307, 2003.

BCG+17. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. pages 2105–2122, 2017.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. pages 291–308, 2019.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. pages 489–518, 2019.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. pages 1069–1080, 2020.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. pages 896–912, 2018.

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. pages 278–291, 1994.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. pages 501–519, 2014.

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. pages 520–536, 2012.

BKW00. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. pages 435–440, 2000.

BLP11. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. pages 743–760, 2011.

BM97. Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. pages 163–192, 1997.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. pages 25–46, 2018.

Bor57. Jan Lourens Bordewijk. Inter-reciprocity applied to electrical networks. 1957.

BR17. Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. Cryptology ePrint Archive, Report 2017/652, 2017. http://eprint.iacr.org/2017/652.

BTV16. Sonia Bogos, Florian Tramer, and Serge Vaudenay. On solving lpn using bkw and variants. 2016.

BV16. Sonia Bogos and Serge Vaudenay. Optimization of LPN solving algorithms. pages 703–728, 2016.

BVJD02. C. Berrou, S. Vaton, M. Jezequel, and C. Douillard. Computing the minimum distance of linear codes by the error impulse method. 2002.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. pages 280–300, 2013.

CCK+18. Ming-Shing Chen, Chen-Mou Cheng, Po-Chun Kuo, Wen-Ding Li, and Bo-Yin Yang. Multiplying boolean polynomials with frobenius partitions in additive fast fourier transform, 2018.

CG90. John T. Coffey and Rodney M. Goodman. The complexity of information set decoding. 1990.

DAT17. Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. 2017.

DP15. A. Dutta and A. Pramanik. Modified approximate lower triangular encoding of ldpc codes. 2015.

DRU06. Changyan Di, Thomas J. Richardson, and Rüdiger L. Urbanke. Weight distribution of low-density parity-check codes. 2006.

EKM17. Andre Esser, Robert Kübler, and Alexander May. LPN decoded. pages 486–514, 2017.

Fei02. Uriel Feige. Relations between average case complexity and approximation complexity. pages 534–543, 2002.

FGKP09. Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. 2009.
FKI06. Marc PC Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. 2006.
FS09. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. pages 88–105, 2009.
Gal62. Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.
Gal13. Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors. 2013.
GDP73. S. I. Gelfand, R. L. Dobrushin, and M. S. Pinsker. On the complexity of coding. 1973.
GJL20. Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. 33(1):1–33, January 2020.
GMW87a. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. pages 218–229, 1987.
GMW87b. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. pages 171–185, 1987.
Gra06. Markus Grassl. *Searching for linear codes with large minimum distance*. 2006.
HIQO19. Fernando Hernando, Francisco D. Igual, and Gregorio Quintana-Ortí. Algorithm 994: Fast implementations of the brouwer-zimmermann algorithm for the computation of the minimum distance of a random linear code. 2019.
HM17. Gottfried Herold and Alexander May. Lp solutions of vectorial integer subset sums–cryptanalysis of galbraith's binary matrix lwe. 2017.
HS13. Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. 2013.
IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. pages 145–161, 2003.
IKOS08. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. pages 433–442, 2008.
Kil88. Joe Kilian. Founding crytpography on oblivious transfer. 1988.
Kir11. Paul Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. http://eprint.iacr.org/2011/377.
KKRT16. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. pages 818–829, 2016.
KOS15. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. pages 724–741, 2015.
KPTZ13. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. pages 669–684, 2013.
KS12. K. Kobayashi and T. Shibuya. Generalization of lu's linear time encoding algorithm for ldpc codes. 2012.
KT17. Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto*, 2017.
LF06. Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. pages 348–359, 2006.
Lyu05. Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. 2005.
McE78. Robert J McEliece. A public-key cryptosystem based on algebraic. 1978.
MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. pages 107–124, 2011.
MO15. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. pages 203–228, 2015.
MST03. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. pages 136–145, 2003.
NN90. Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. pages 213–223, 1990.
NPC+17. Robert Niebuhr, Edoardo Persichetti, Pierre-Louis Cayrel, Stanislav Bulygin, and Johannes A. Buchmann. On lower bounds for information set decoding over $\mathbb{F}_q$ and on the effect of partial knowledge. 2017.
OOS17. Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. pages 381–396, 2017.
OTA07. Ayoub Otmani, Jean-Pierre Tillich, and Iryna Andriyanova. On the minimum distance of generalized LDPC codes. 2007.
Ove06. Raphael Overbeck. Statistical decoding revisited. pages 283–294, 2006.

Pet10. Christiane Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In Nicolas Sendrier, editor, *PQCrypto*, 2010.
Pra62. Eugene Prange. The use of information sets in decoding cyclic codes. 1962.
PRTY20. Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. pages 739–767, 2020.
Rin. Peter Rindal. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.
RS21. Peter Rindal and Phillipp SChoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In *Eurocrypt*, 2021.
RU01. Thomas J. Richardson and Rüdiger L. Urbanke. Efficient encoding of low-density parity-check codes. 2001.
Saa07. Markku-Juhani Olavi Saarinen. Linearization attacks against syndrome based hashes. pages 1–9, 2007.
SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. pages 1055–1072, 2019.
Shp09. Amir Shpilka. Constructions of low-degree and error-correcting ε-biased generators. 2009.
SNDM18. Tikaram Sanyashi, Sreyans Nahata, Rushang Dhanesha, and Bernard Menezes. Learning plaintext in galbraith's lwe cryptosystem. 2018.
Spi96. Daniel A Spielman. Linear-time encodable and decodable error-correcting codes. 1996.
Ste88. Jacques Stern. A method for finding codewords of small weight. 1988.
SVA+19. Tikaram Sanyashi, M Venkatesh, Kapil Agarwal, Manish Verma, and Bernard Menezes. A new hybrid lattice attack on galbraith's binary lwe cryptosystem. 2019.
TS16. Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *PQCrypto*, 2016.
TZ06. Jean-Pierre Tillich and Gilles Zémor. On the minimum distance of structured LDPC codes with two variable nodes of degree 2 per parity-check equation. 2006.
Var97. A. Vardy. The intractability of computing the minimum distance of a code. 1997.
Wag02. David Wagner. A generalized birthday problem. pages 288–303, 2002.
WYKW20. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits, 2020.
XFE04. Xiao-Yu Hu, M. P. C. Fossorier, and E. Eleftheriou. On the computation of the minimum distance of low-density parity-check codes. 2004.
Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). pages 162–167, 1986.
YWL+20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. pages 1607–1626, 2020.
Zic17. Lior Zichron. Locally computable arithmetic pseudorandom generators, 2017.
ZJW16. Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster algorithms for solving LPN. pages 168–195, 2016.

Supplementary Material

A Preliminaries on Silent OT Extension

In this section, we recall the silent OT extension protocol of [BCG+19a], on top of which our protocol is built (some descriptions are taken almost verbatim from [BCG+19a]).

A.1 Pseudorandom Correlation Generator for VOLE

First, we recall the high-level idea of the pseudorandom correlation generators (PCG) for vector-OLE from [BCGI18, BCG+19b]. A PCG is a pair of algorithms $(\mathsf{Gen}, \mathsf{Expand})$, where Gen distributes a pair of seeds to a sender and a receiver. Later, the seeds can be locally expanded with Expand into long pseudorandom correlations, for a target class of correlation (here, subfield vector OLE). The construction of [BCGI18, BCG+19b] uses two main ingredients: a variant of the LPN assumption, and a method for the two parties to obtain a compressed form of (pseudo)random secret shares $\mathbf{s}_0, \mathbf{s}_1$, satisfying

$$\mathbf{s}_1 = \mathbf{s}_0 + \mathbf{e} \cdot x \in \mathbb{F}_{2^\lambda}^n \tag{1}$$

where $\mathbf{e} \in \{0,1\}^n$ is a random, sparse vector held by one party, and $x \in \mathbb{F}_{2^\lambda}$ is a random field element held by the other party. Given this, the shares can be randomized using a public binary matrix $G^\intercal$ that compresses from n down to $m < n$ elements, and locally multiplying each share with $G^\intercal$. The underlying assumption is that $\mathbf{a} = G^\intercal \cdot \mathbf{e}$ is pseudorandom under a suitable variant of LPN. Now, writing $\mathbf{b} = G^\intercal \cdot \mathbf{s}_0$ and $\mathbf{c} = G^\intercal \cdot \mathbf{s}_1$, from (1) we then get $\mathbf{c} = \mathbf{b} + \mathbf{a}x$. This can be seen as a set of random *correlated OTs*, where $a_i \in \{0,1\}$ are the receiver's choice bits, and $(b_i, b_i + x)$ are the sender's strings, of which the receiver learns c_i. These can be locally converted into random string-OTs with a standard hashing technique [IKNP03].

To obtain a compressed form of the shares in (1), the constructions of [BCG+19a] uses a *puncturable pseudorandom function* (PPRF) primitive. A PPRF is a PRF F such that given an input x, and a PRF key K, one can generate a punctured key $\mathsf{K}\{x\}$ which allows evaluating F at every point except for x, and does not conceal any information about the value $F(\mathsf{K}, x)$. A PPRF can be built from any length-doubling pseudorandom generator, using a binary tree-based construction [KPTZ13, BW13, BGI14]. The construction works as follows: Gen will give the sender a random key K and x, and give to the receiver a random point $\alpha \in \{1, \ldots, N\}$, a punctured key $\mathsf{K}\{\alpha\}$, and the value $z = F(\mathsf{K}, \alpha) + x$. Given these seeds, the sender and receiver can now define the expanded outputs, for $i = 1, \ldots, n$:

$$\mathbf{s}_0[i] = F(\mathsf{K}, i), \quad \mathbf{s}_1[i] = \begin{cases} F(\mathsf{K}, i) & i \neq \alpha \\ z & \text{otherwise} \end{cases}$$

These immediately satisfy (1), with $\mathbf{e}$ as the α-th unit vector. Then, to obtain shares of sparse $\mathbf{e}$ with t non-zero coordinates, simply repeat the above t times and XOR together all t sets of outputs.

A.2 Distributing the Setup

To obtain a silent OT out of the above PCG for VOLE and OT correlations, it remains to distributively implement the setup protocol (i.e., securely compute and distribute the outputs of the Gen algorithm). The work of [BCG+19a] gives a simple two-round protocol for this task. At a high level, it works as follows: for each of t secret LPN noise coordinates $\alpha_j \in [N]$ known to the receiver, the sender generates a fresh PRF key K_j, and obliviously communicates a punctured key $\mathsf{K}_j\{\alpha_j\}$ and hardcoded punctured output $z_j = \mathsf{PRF}(\mathsf{K}_j, \alpha) + x$ to the receiver. Overall, this gives shares of $x \cdot \mathbf{e}$.

To execute this oblivious communication of the punctured keys, for each $\mathsf{K}\{\alpha\}$, the parties rely on $\ell = \log N$ parallel OT executions: the sender's ℓ message pairs correspond to appropriate sums of partial evaluations from a consistent GGM PRF tree and his secret value x, and the receiver's ℓ selection bits correspond to the bits of his chosen path α. We defer the reader to [BCG+19a] for a full description of the exact protocol.

B Preliminaries on Coding Theory

Generator Matrix. Let $k < n \in \mathbb{N}$. If $G \in \mathbb{F}^{k \times n}$ is an $k \times n$ matrix over the field $\mathbb{F}$, it generates the codewords of a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ given by

$$\mathcal{C} := \{\mathbf{c} = \mathbf{x}G \mid \mathbf{x} \in \mathbb{F}^k\}$$

where $\mathbf{c}$ is a codeword of the linear code $\mathcal{C}$, and $\mathbf{x}$ is any input k-length vector. G is called the *generator matrix* of $\mathcal{C}$. Both $\mathbf{c}$ and $\mathbf{x}$ are assumed to be row vectors. If $|\mathbb{F}| = q$, $\mathcal{C}$ is called an $[n, k, d]_q$-code, where d is the minimum distance of the code (the distance of the linear code is the minimum weight of its nonzero codewords, or equivalently, the minimum distance between distinct codewords). It will be useful to define $m := n - k$.

Standard Form. The *standard form* for a generator matrix is,

$$G = [I_k|P]$$

where I_k is the $k \times k$ identity matrix and P is a $k \times m$ matrix. When the generator matrix is in standard form, the code $\mathcal{C}$ is said to be *systematic* in its first k coordinate positions. In this case, the message can be read directly from the first k positions codeword, i.e. $c = G\mathbf{x} = [\mathbf{x}|\mathbf{p}]$ for some parity information $\mathbf{p} \in \mathbb{F}^m$. An arbitrary generator matrix G' can be placed in standard form by performing elementary row operations. Depending on how G' is sampled, it is possible that some columns of G' are linearly dependent in which case permuting the columns of G' (and the resulting codewords) might be required.

Parity Check Matrix. The matrix $H \in \mathbb{F}^{m \times n}$ representing a linear function $\phi : \mathbb{F}^n \to \mathbb{F}^{n-k}$ whose kernel is $\mathcal{C}$ is called a *parity check matrix* of $\mathcal{C}$. Equivalently, H is a matrix whose null space is $\mathcal{C}$, i.e. $\mathcal{C} = \{\mathbf{c} \mid H\mathbf{c}^\intercal = 0\}$. It can be verified that H is a $m \times n$ matrix. The code generated by H is called the *dual code* of $\mathcal{C}$, denoted by $\mathcal{C}^\perp$.

Minimum Distance. The distance d of a linear code $\mathcal{C}$ can be defined as the minimum number of linearly dependent columns of the parity check matrix H. Equivalently, it is the minimum number of positions of a codeword $\mathbf{c} \in \mathcal{C}$ which must be modified to produce another codeword $\mathbf{c}'$. Due to the linearity of $\mathcal{C}$, it is easy to see that d is equal to the minimum weight non-zero codeword. Computing the minimum distance for an arbitrary linear code $\mathcal{C}$ is known to be NP-complete [Var97]. As explained in Section 5, various methods have been proposed for bounding the minimum distance of linear codes.

Relating the Generator and Parity Check Matrices A generator matrix can be used to construct the parity check matrix for a code (and vice versa). If the generator matrix G is in standard form,

$$G = [I_k|P]$$

then the parity check matrix for $\mathcal{C}$ is

$$H = [-P^\intercal|I_{n-k}]$$

where $P^\intercal$ is the transpose of the matrix P and I_{n-k} is the $m \times m$ identity matrix.

B.1 Decoding

Standard Array Decoding

Standard Array. A *standard array* $\mathsf{Arr} \in \mathbb{F}^{m \times k}$ is an array that lists all elements of a particular $\mathcal{F}^n$ vector space. Standard arrays can be used to decode linear codes over a noisy channel on which errors are made, i.e., to find the corresponding codeword for any received vector. A standard array can be created as follows:

1. List the codewords of $\mathcal{C}$, starting with 0, as the first row $\mathsf{Arr}_1 = \{\mathbf{c} \mid \mathbf{c} \in \mathcal{C}\}$.

2. Let i index the next row and choose any vector $\mathbf{v}$ of minimum weight not already in the array. Then define $\mathsf{Arr}_i = \mathsf{Arr}_1 + \mathbf{v}$. That is,
   (a) write this as the first entry of the next row i, i.e. $\mathsf{Arr}_{i,1} = \mathbf{v}$. This vector $\mathsf{Arr}_{i,j}$ is denoted the *coset leader*.
   (b) Fill out the row i by adding the coset leader to the codeword at the top of each column. The sum of the ith coset leader and the jth codeword becomes the entry in row i, column j, i.e. $\mathsf{Arr}_{i,j} = \mathsf{Arr}_{i,1} + \mathsf{Arr}_{1,j}$ for $j \in [2, 2^k]$.

   Repeat this step until all rows/cosets are listed and each vector appears exactly once.

Standard Array Decoding. To decode a vector using a standard array, for each coset leader $\mathsf{Arr}_{i,1}$, subtract it from the vector received. For some i, the result will be one of the codewords in $\mathcal{C}$. Decoding via a standard array is a form of *nearest neighbour decoding*. In practice, decoding via a standard array requires large amounts of storage. Other forms of decoding are more efficient. Decoding via standard array does not guarantee that all vectors are decoded correctly. In such a case some implementations might ask for the message to be resent, or the ambiguous bit may be marked as an erasure and a following outer code may correct it. This ambiguity is another reason that different decoding methods are sometimes used.

Syndrome Decoding

Syndrome. For any row vector $\mathbf{c} \in \mathbb{F}^n$, $\mathbf{s} = H\mathbf{c}^\intercal$ is called the *syndrome* of $\mathbf{c}$. The vector $\mathbf{c}$ is a codeword if and only if $\mathbf{s} = 0$.

Syndrome Decoding. The calculation of syndromes is the basis for the *syndrome decoding algorithm*. Syndrome decoding is a highly efficient method of decoding a linear code over a noisy channel. In essence, syndrome decoding is minimum distance decoding using a reduced lookup table. Note that linear code $\mathcal{C}$ is capable of correcting up to

$$t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$$

errors made by the channel (since if no more than t errors are made, then minimum distance decoding will still correctly decode the incorrectly transmitted codeword). Now suppose that a codeword $\mathbf{c} \in \mathbb{F}^n$ is sent over the channel and the error pattern $\mathbf{e} \in \mathbb{F}^n$ occurs. Then $\mathbf{z} = \mathbf{c} + \mathbf{e}$ is received. Ordinary minimum distance decoding would lookup the vector $\mathbf{z}$ in a table of size $|\mathcal{C}|$ for the nearest match, i.e., an element (not necessarily unique) $\mathbf{c} \in \mathcal{C}$ with

$$d(\mathbf{c}, \mathbf{z}) \leq d(\mathbf{y}, \mathbf{z})$$

for all $\mathbf{y} \in \mathcal{C}$. Syndrome decoding takes advantage of the property of the parity matrix that $H\mathbf{c}^\intercal = 0$ for all $\mathbf{c} \in \mathcal{C}$. The syndrome of the received $\mathbf{z} = \mathbf{c} + \mathbf{e}$ is $H\mathbf{z}^\intercal = H\mathbf{e}^\intercal$. To perform ML decoding in a binary symmetric channel, one has to look-up a pre-computed table of size q^{n-k}, mapping $H\mathbf{e}^\intercal$ to $\mathbf{e}$. Note that this is already of significantly less complexity than that of a standard array decoding. Knowing what $\mathbf{e}$ is, it is then trivial to decode $\mathbf{c}$ as $\mathbf{c} = \mathbf{z} - \mathbf{e}$.

Syndrome Decoding of Binary Codes. For *binary codes* ($q = 2$), under the assumption that no more than t errors were made during transmission, the receiver can look up the value $H\mathbf{e}^\intercal$ in a further reduced table of size

$$\sum_{i=0}^{t} \binom{n}{i} < |\mathcal{C}|$$

Another approach to decoding would be the following. If both k and $n-k$ are not too big, and assuming the generating matrix is in standard form, syndrome decoding can be computed using two precomputed lookup tables and two XORs only. Let $\mathbf{z}$ be the received noisy codeword, i.e., $\mathbf{z} = \mathbf{c} \oplus \mathbf{e}$. Using the encoding lookup table of size 2^k, the codeword $\mathbf{c}'$ that corresponds to the first k bits of $\mathbf{z}$ is found. The syndrome is then computed as the last $n-k$ bits of $\mathbf{s} = \mathbf{z} \oplus \mathbf{c}'$. Using the syndrome, the error $\mathbf{e}$ is computed using the syndrome lookup table of size 2^{n-k}, and the decoding is then computed via $\mathbf{c} = \mathbf{z} \oplus \mathbf{e}$. The number of entries in the two lookup tables is $2^k + 2^{n-k}$, which is significantly smaller than 2^n required for standard array decoding that requires only one lookup.

Information Set Decoding *Information set decoding* refers to a family of Las Vegas-probabilistic methods all based on the observation that it is easier to guess enough error-free positions, than it is to guess all the error-positions. The simplest form is the following: Select k columns of G at random, and denote by G' the corresponding sub-matrix of G. With reasonable probability G' will have full rank, which means that if we let $\mathbf{c}'$ be the sub-vector for the corresponding positions of any codeword $\mathbf{c} = \mathbf{x}G$ of $\mathcal{C}$ for a message $\mathbf{x}$, we can recover $\mathbf{x}$ as $\mathbf{x} = \mathbf{c}'G'^{-1}$. Hence, if we were lucky that these k positions of the received word $\mathbf{z}$ contained no errors, and hence equalled the positions of the sent codeword, then we may decode. If t errors occurred, the probability of such a fortunate selection of columns is given by

$$\frac{\binom{n-t}{k}}{\binom{n}{k}} = \frac{(m-t+1)\dots(m-t+t)}{(n-t+1)\dots(n-t+t)}$$

This method has been improved in various ways. For more on this, consider [CG90, Pet10, BLP11, BJMM12, HS13, MO15, TS16, KT17, NPC+17] and the references therein.

B.2 Tanner Graphs

A *Tanner graph* is a bipartite graph used to state constraints or equations which specify error correcting codes. Tanner graphs are partitioned into *constraint nodes* or *check nodes* and *digit nodes* or *variable nodes*. For linear codes, the constraint nodes denote rows of the parity-check matrix H. The digit nodes represent the columns of the matrix H. An edge connects a constraint node to a digit node if a non-zero entry exists in the intersection of the corresponding row and column.
As an example, consider the binary $[11, 6, d]_2$ code $\mathcal{C}$ with generator matrix

$$G = \left(\begin{array}{cccccc|ccccc} 1&0&0&0&0&0&1&0&1&0&0\\ 0&1&0&0&0&0&1&0&0&1&0\\ 0&0&1&0&0&0&1&0&0&0&1\\ 0&0&0&1&0&0&0&1&1&0&0\\ 0&0&0&0&1&0&0&1&0&1&0\\ 0&0&0&0&0&1&0&1&0&0&1 \end{array}\right) = [I_6|P]$$

in its standard form. Its parity check matrix is given by

$$H = \left(\begin{array}{cccccc|ccccc} 1&1&1&0&0&0&1&0&0&0&0\\ 0&0&0&1&1&1&0&1&0&0&0\\ 1&0&0&1&0&0&0&0&1&0&0\\ 0&1&0&0&1&0&0&0&0&1&0\\ 0&0&1&0&0&1&0&0&0&0&1 \end{array}\right) = [-P^T|I_5]$$

Let us name the rows of H as $c_1, \dots, c_5$ referring to the five parity checks that H performs. They will be the constraint nodes in the Tanner graph corresponding to H. Similarly, let us name the columns of H as $d_1, \dots, d_6, p_1, \dots, p_5$ referring to the eleven variable bits (six data bits and five parity bits) in the codeword that H will perform parity checks on. They will be the digit nodes in the Tanner graph corresponding to H. The Tanner graph corresponding to H is shown in Figure 8.

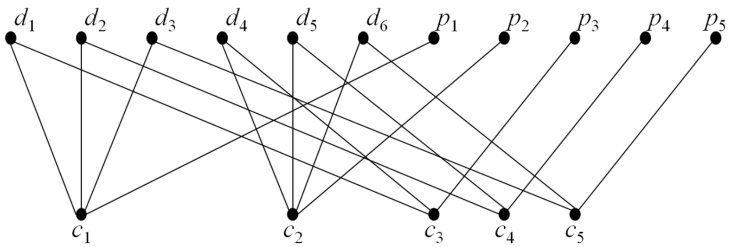


Fig. 8: Example Tanner Graph

B.3 LDPC Codes

A *low-density parity check code*, or *LDPC code*, is constructed using a sparse Tanner graph. The characteristic of LDPC codes is that H contains very few ones with respect to its total size, i.e., H is sparse (this is where the name low-density comes from). Let H be an $(n-k) \times n$ parity check matrix. Let $i \in [n-k]$ and $j \in [n]$ be indices to index into the rows and columns of H respectively. Let $\mathsf{wt}_{\mathsf{row},i}$ denote the number of non-zero entries in the ith row of H and let $\mathsf{wt}_{\mathsf{col},j}$ denote the number of non-zero entries in the jth column of H. For LDPC codes, we must have

$$\mathsf{wt}_{\mathsf{row},i} \ll n$$

and

$$\mathsf{wt}_{\mathsf{col},i} \ll n-k$$

for all i, j. If $\mathsf{wt}_{\mathsf{row},i}$ is a constant for all i and $\mathsf{wt}_{\mathsf{col},j}$ is a constant for all j, then the code is said to be a *regular code*. In such a case, it is easy to see that

$$\frac{\mathsf{wt}_{\mathsf{row},i}}{\mathsf{wt}_{\mathsf{col},j}} = \frac{n}{n-k}$$

Codes whose parity check matrices have a varying number of non-zero entries per row or column are called *irregular codes*.

C On the Dual Distance of Random Sparse Matrices

In this section, we prove Theorem 7, and discuss an interesting consequence of the proof regarding the minimum distance of LDPC codes over large fields.

C.1 Proof of Theorem 7

We prove Theorem 7, which we recall below:

Theorem 7 (Most sparse matrices have dual distance $O(k)$). *For any constant $c > 1$ and integer $t > 2$, there is a constant $\gamma = \gamma(c,t)$ such that for any large enough k, denoting $n = c \cdot m$,*

$$\Pr\left[A \xleftarrow{\$} \mathsf{W}_t(\mathbb{F}_2^{n\times m}) \;:\; \frac{\mathsf{dd}(A)}{m} < \gamma\right] \leq 1 - m^{2.1-t}.$$

The proof is a relatively simple adaptation to our setting of the analysis of [MST03, Section 5.3].

Proof. Given a matrix $A \in \mathbb{F}_2^{n\times m}$, we denote by $(A_1, \cdots, A_n)$ the rows of A, and by $(A^1, \cdots, A^m)$ its columns. For any subset $S \subseteq [n]$ of the rows of A, we call *column-neighbors of S in A*, and write $\mathsf{N}^{\mathsf{c}}_A(S) \subseteq [m]$, the subset of the columns of A which have at least one 1 in a row of S. For any integer d, we say that A has the *d-unique column-neighbor property* if for any set $S \subseteq [n]$ with $|S| \leq d$, there exists a column A^j of A such that $A^j[S]$ contains exactly a single 1.

Lemma 8 ([MST03]). *If $A \in \mathbb{F}_2^{n\times m}$ has d-unique column-neighbor, then $\mathsf{dd}(A) \geq d-1$.*

Proof. For any subset S of $[n]$ with $|S| \leq d$, there exists a column of A that has exactly one 1 in a row indexed by S, hence $\bigoplus_{i\in S} A_i \neq 0^m$.

In the following, we will show that a randomly sampled sparse matrix A will have d-right unique column-neighbor with high probability, for a certain d. This will follow from the fact that A has certain expansion properties. We say that a matrix A is (d,α)-expanding if for every subset $S \subseteq [n]$ with $|S| \leq d$, it holds that $|\mathsf{N}_A(S)| > \alpha \cdot |S|$.[4]

Lemma 9 ([MST03]). *Let $A \in \mathsf{W}_t(\mathbb{F}_2^{n\times m})$ be a t-sparse matrix. If A is $(d, t/2)$-expanding, then it has d-unique column-neighbor.*

[4] While expansion properties are most commonly given over graphs, the extension to matrices is straightforward: our definition simply says that a matrix A is (d,α)-expanding if it is the adjacency matrix of a (d,α)-expanding bipartite graph.

Proof. Assume that A does not have d-unique column-neighbor. Let $S \subseteq [n]$ be any subset with $|S| \leq d$. Then for every $j \in \mathsf{N}^{\mathsf{c}}_A(S)$, $A^j[S]$ contains at least two 1's. Since the rows in S contain $t \cdot |S|$ 1's in total, this implies that $|\mathsf{N}^{\mathsf{c}}_A(S)| \leq (t/2) \cdot |S|$.

To prove Theorem 7, it remains to show that whenever $n = c \cdot m$ for some constant c, a random sample from $\mathsf{W}_t(\mathbb{F}_2^{n \times m})$ is $(m/C, t/2)$-expanding with high probability, for some constant C.[5]

Lemma 10. *For any large enough $m = m(\lambda)$, $n = c \cdot m$ for a constant c, any $t > 2$, there is a constant C such that*

$$\Pr[A \xleftarrow{\$} \mathsf{W}_t(\mathbb{F}_2^{n \times m}) \ : \ A \text{ is } (m/C, t/2)\text{-expanding }] \geq 1 - \left(\frac{1}{O(m)}\right)^{t-2}.$$

Proof. For any subset $S \subset [n]$ and any size $t \cdot |S|/2$ subset $T \subset [m]$, the probability over the random choice of $A \xleftarrow{\$} \mathsf{W}_t(\mathbb{F}_2^{n \times m})$ that $\mathsf{N}^{\mathsf{c}}_A(S) \subseteq T$ is clearly at most $(t \cdot |S|/2m)^{t \cdot |S|}$. Since there are $\binom{n}{|S|}$ choices for S and $\binom{m}{t|S|/2}$ choices for T, the probability that A fails to be $(d, t/2)$-expanding is at most

$$\sum_{i=2}^{d} \binom{c \cdot m}{i} \cdot \binom{m}{t \cdot i/2} \cdot \left(\frac{t \cdot i}{2m}\right)^{t \cdot i},$$

where the sum starts at 2 because any single row j always satisfies $\mathsf{N}^{\mathsf{c}}_A(\{j\}) = t > t/2$. Using the inequality $\binom{a}{b} \leq (ae/b)^b$, we get

$$\begin{aligned}
\sum_{i=2}^{d} \left(\frac{ecm}{i}\right)^i \cdot \left(\frac{em}{ti/2}\right)^{ti/2} \cdot \left(\frac{ti}{2m}\right)^{ti} &= \sum_{i=2}^{d} \left(\frac{ecm}{i} \cdot \left(\frac{em}{ti/2}\right)^{t/2} \cdot \left(\frac{ti}{2m}\right)^{t}\right)^i \\
&= \sum_{i=2}^{d} \left(c \cdot \left(\frac{t}{2}\right)^{\frac{t/2}{t/2-1}} \cdot e^{\frac{t/2+1}{t/2-1}} \cdot \frac{i}{m}\right)^{i \cdot (t/2-1)} \\
&= \sum_{i=2}^{d} \left(\frac{C}{2} \cdot \frac{i}{m}\right)^{i \cdot (t/2-1)},
\end{aligned}$$

where C denotes the constant $2c \cdot (t/2)^{\frac{t/2}{t/2-1}} \cdot e^{\frac{t/2+1}{t/2-1}}$. Now, setting $d = m/C$, the above is upper bounded by

$$\left(\frac{C}{m}\right)^{t-2} + \left(\frac{3C}{2m}\right)^{3(t/2-1)} + \log^2 m \cdot \left(\frac{4C^2 \log^4 m}{m^2}\right)^{t-2} + \frac{m}{C} \cdot \left(\frac{1}{m^{\log m}}\right)^{t/2-1},$$

where the first two terms are the terms for $i = 2, 3$ in the sum, the third term bounds the terms $i = 4, \cdots, \log^2 m$ in the sum, and the last term bounds the sum of the remaining terms. For a large enough m and any integer $t > 2$, this sum is therefore dominated by its first term. The lemma follows.

Theorem 7 directly follows from Lemma 8, 9, and 10.

C.2 Dual Distance of Random Binary Sparse Matrices

We now make a crucial observation, which is the core motivation behind our candidate for the silent VOLE application: the previous proof that random sparse matrices have high dual distance over $\mathbb{F}_2$ generalizes directly to arbitrary fields. Even more, it also generalizes to the setting of codes over an arbitrary field, *even when the parity-check matrix is sampled over* $\{0,1\}^{n \times m}$. This is because the analysis relies solely on the expansion properties of the bipartite graph which has A as its adjacency matrix. In our framework, this means that with probability $1 - o(1)$ over the random choice of A from $\mathsf{W}_t(\{0,1\}^{n \times m})$, any linear attack against the LPN assumption with code matrix A *over* $\mathbb{F}$ has advantage at most $\exp(-\Omega(t))$ when $\mathbf{e}$ is randomly sampled from either a Bernoulli distribution over

[5] A similar statement actually holds for any $c(m) = o(\sqrt{m}/(\log m)^{3/4})$ (see [MST03]), but the weaker statement suffices in our context.

$\mathbb{F}$ with rate t/n, the uniform distribution over all t-sparse vectors over $\mathbb{F}^n$, or the regular distribution (*i.e.*, the concatenation of t random unit vectors over $\mathbb{F}^{n/t}$).

In light of known attacks against LPN, this suggests than Alekhnovich's assumption (primal LPN with a random sparse code matrix) remains plausibly hard over arbitrary fields, even when the matrix is restricted to have $\{0,1\}$ entries. We note that the hardness of Alekhnovich's assumption over large fields has been conjectured and use in previous works (see e.g. [ADI+17]), but to our knowledge, we are the first to observe that the underlying analysis extends to the case of binary matrices as well.

Looking ahead, while the protocol of [BCG+19a] for silent OT extension relies only on LPN over $\mathbb{F}_2$, it can directly be generalized over arbitrary fields, in which case it provides a silent protocol for vector OLE over $\mathbb{F}$. Silent protocols for vector OLE over large fields have recently found important applications; for example, they lead to the most efficient solutions known to date for private set intersection [RS21], one of the most actively researched areas of secure computation. In our candidate structured LDPC codes over general fields, the use of matrices with $\{0,1\}$ entries will result in (very) significant efficiency improvements, while (plausibly) not harming security.

Comparison to binary-matrix LWE. The knowledgeable reader might be aware that a similar proposal was made in [Gal13], in the setting of the *learning with error* assumption. The end goal for this proposal was superficially similar: increasing efficiency and reducing storage costs, by assuming the hardness of LWE over a field $\mathbb{F}_q$ with respect to a matrix with $\{0,1\}$ entries. Unfortunately, a number of devastating attacks against this proposal have been described in [HM17, SNDM18, SVA+19], using linear programming algorithms. The reader may wonder, then, why things should be any different in our setting, and whether this does not make our assumption at least highly suspicious.

We argue that, in spite of the superficial similarity, the situations are actually incomparable. In short, attacks against binary-matrix LWE crucially rely on the fact that the LWE error terms are guaranteed to have *small magnitude*. Since the entries in the matrix are also small, this implies that binary-matrix LWE can be reduced to the hardness of finding an integral solution of a set of equations *over the integers*, a problem which can be solved efficiently using linear programming. However, the situation is entirely different in our setting, because the nonzero entries of our noise vector are *uniformly random over the field* $\mathbb{F}$. This effectively prevents reducing the problem to that of solving a system over the integers.

D Detailed Circuit for $f(\mathbf{e}) = \mathbf{e} \cdot G^\intercal$

Recall that for dual LPN, we want to multiply the error vector with the transpose of $G \in \mathbb{F}^{k \times n}$. This can be handled with the circuit transposition principle [Bor57]. Moreover, we present the generalized algorithm where $H \in \mathbb{F}^{m \times n}$ is approximately triangular with gap g.

Linear time LDPC encoding We begin by presenting a detailed description of the circuit computing $f(\mathbf{m}) = \mathbf{m} \cdot G$ as shown in Figure 10. We begin by requiring H to be in approximate lower triangular form consisting of

$$H = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}$$

where $C \in \mathbb{F}^{\sigma \times \sigma}$ is a (strict) lower triangular with ones on the diagonal and $E \in \mathbb{F}^{g \times g}$. We will also require $E' = -FC^{-1}B + E$ to be invertible. Let $H' = [-P^\intercal | I_m]$ be the systematic form of H. We assume no column permutations are required to convert H into H' and as such a systematic codewords $\mathbf{c} = [\mathbf{x}|\mathbf{p}]$ for message $\mathbf{x} \in \mathbb{F}^k$ is also in the nullspace of H, i.e. $Hw^\intercal = 0$.

Since C is triangular, we can zero F out by multiply H with $[I|0 \mathbin{/\!\!/} -FC^{-1}|I]$ to obtain

$$\begin{bmatrix} A & B & C \\ D' & E' & 0 \end{bmatrix}$$

where $D' = -FC^{-1}A + D, E' = -FC^{-1}B + E$. Let $[\mathbf{p}_1, \mathbf{p}_2] := \mathbf{p}$ where $\mathbf{p}_1 \in \mathbb{F}^g$. Observe that

$$\begin{aligned} \mathbf{p}_1^\intercal &= -E'^{-1}(D'\mathbf{x}^\intercal) \\ &= -E'^{-1}(-F(C^{-1}(A\mathbf{x}^\intercal)) + D\mathbf{x}^\intercal) \\ \mathbf{p}_2^\intercal &= -C^{-1}(A\mathbf{x}^\intercal + B\mathbf{p}_1^\intercal) \end{aligned}$$

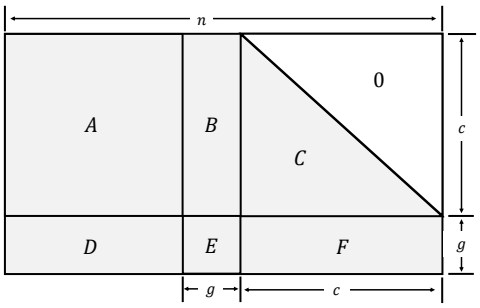


Fig. 9: The structure of an g approximate lower triangular matrix. The diagonal of C should all be ones.

In the case that $g = 0$, then this simplifies to $\mathbf{p} = -C^{\intercal}(A\mathbf{x}^{\intercal})$.

Observe that we must compute $\mathbf{v} = C^{-1}\mathbf{y}$ for $\mathbf{y} \in \{A\mathbf{x}^{\intercal}, A\mathbf{x}^{\intercal} + B\mathbf{p}_1^{\intercal}\}$. This is done be leveraging that C is lower diagonal by solving the system $\mathbf{y} = C\mathbf{v}$. For each row starting at row $i = 1$, observe that $\mathbf{y}_i = \sum_{j \in I_i} \mathbf{v}_j$ where I_i indexes the non-zero columns of the row C_i. Observe that $i \in I_i$ since C is diagonal. Therefore we can rewrite this as $\mathbf{v}_i = \mathbf{y}_i - \sum_{j \in I_i \setminus \{i\}} \mathbf{v}_j$ where these $\mathbf{v}_j$ have all been previously defined, i.e. $j \leq i$ for $j \in I_i$.

The full algorithm is presented in Figure 10 and manipulates the state of the algorithm by multiplying it with a square matrix at each step. This will assist in the explanation of the transposition principle. Note that no operations are requires when something is multiplied by its identity. The algorithm requires $O(n + g^{2.3})$ or $O(n + g^3)$ time to preprocess $-E^{-1}$ and then the main circuit requires $O(n + g^2)$ time. In particular, every step of the protocol requires $O(n)$ time except for step 4 when computing $(-E^{-1})\mathbf{p}$ which requires $O(g^2)$ since $E^{-1} \in \mathbb{F}^{g \times g}$ and $\mathbf{p} \in \mathbb{F}^g$ are dense. We note that when $g = 0$ the encoding algorithm can be simplified as shown in Figure 11.

Linear time encoding for transposed code We now discuss how to compute $f'(\mathbf{e}) = \mathbf{e} \cdot G^{\intercal}$ with the same complexity as computing $\mathbf{x} \cdot G$. The full algorithm is presented in Figure 12 and is a direct application of the circuit transposition principle [Bor57] on Figure 10. We leave the application of the transposition principle on the simplified encoder (Figure 11) to the reader.

The transposition principle states that if you transpose each of the matrix multiplications and apply them in reverse order, then what you get is the overall circuit computation transposed. Therefore, if we have a circuit which compute $f(\mathbf{x}) = \mathbf{x} \cdot G$ and we compile this into our matrix-multiplication representation (as done is Figure 10), transpose all of the multiplications, apply them in reverse order, then we will get a circuit for $f'(\mathbf{e}) = \mathbf{e} \cdot G^{\intercal}$ which has the same complexity as computing $\mathbf{x} \cdot G$.

E Linear-Time LPN Encoding with Gap

We now present an optimization which allows reducing the computational overhead of $O(n + g^2)$ for LDPC encoding to $O(n + g)$ at the expense of increasing the communication overhead of the protocol by $O(g)$. Moreover, this optimization reduces the constant term in front of n by a factor two. In the primal setting, we are computing $\mathbf{u} = H^{\intercal}\mathbf{x} + \mathbf{e}$, where H is g-approximate lower triangular. Let

$$H' := \left[\begin{array}{c|c} H & \begin{matrix} 0 \\ I_g \end{matrix} \end{array}\right]$$

be the extended matrix where the bottom g rows are the identity matrix. In addition, let $\mathbf{e}' = [\mathbf{e}|\mathbf{e}^*]$ where $\mathbf{e}^* \leftarrow \mathbb{F}^g$ is sampled uniformly. We can then define

$$\mathbf{u}' = H'^{\intercal}\mathbf{x} + \mathbf{e}'.$$

Observe that $\mathbf{u}' = [\mathbf{u}|\mathbf{u}^*]$ where $\mathbf{u}^* = (x_1, ..., x_g) + \mathbf{e}^*$. Since $\mathbf{e}^*$ is uniform then $\mathbf{u}^*$ is uniform that the security of this LPN instance reduces that the original instances.

LDPC encode circuit computing $f(\mathbf{x}) = \cdot G$ given H.

1. Let $H = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}$ where $E \in \mathbb{F}^{g \times g}, C \in \mathbb{F}^{s \times s}$. Abort if C is not lower triangular with ones on the diagonal or if $E' := -FC^{-1}B + E$ is not invertible. Precompute $-E'^{-1}$.
2. Initialize $\mathbf{w} \in \mathbb{F}^n$ s.t. $\mathbf{w} = [\mathbf{x}|\mathbf{p}|\mathbf{p}'|\mathbf{p}'']$, $\mathbf{p} \in \{0\}^g, \mathbf{p}', \mathbf{p}'' \in \{0\}^s$ and compute
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ A & 0 & 0 & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix} \mathbf{w}^\intercal$$
i.e. $\mathbf{p}' := A\mathbf{x}^\intercal$.
3. For in order $i \in [s]$,
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & 0 & I_s & 0 \\ 0 & 0 & U & V \end{bmatrix} \mathbf{w}^\intercal$$
where $U = \{0\}^{s \times s}$ except for $U_{i,i} = 1$ and $V = C_i$ except for $V_{i,j} = -C_{i,j}$ for $j \in [i-1]$ and $V_{i,i} = 0$. That is, $\mathbf{p}''_i := \mathbf{p}'_i - \sum_{j \in [i-1]} C_{i,j}\mathbf{p}''_j = (C^{-1}A\mathbf{x}^\intercal)_i$.
4. Compute
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ D & 0 & 0 & -F \\ 0 & 0 & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix} \mathbf{w}^\intercal, \mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & -E'^{-1} & 0 & 0 \\ 0 & 0 & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix} \mathbf{w}^\intercal, \mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & B & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix} \mathbf{w}^\intercal$$
i.e. $\mathbf{p} := -F\mathbf{p}'' + D\mathbf{x}^\intercal = (-FC^{-1}A + D)\mathbf{x}^\intercal$, then $\mathbf{p} := (-E'^{-1})\mathbf{p} = -E'^{-1}(-FC^{-1}A + D)\mathbf{x}^\intercal$, and finally $\mathbf{p}' := \mathbf{p}' + B\mathbf{p}^\intercal = A\mathbf{x}^\intercal + B\mathbf{p}^\intercal$.
5. For in order $i \in [s]$,
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & 0 & V & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix} \mathbf{w}^\intercal$$
where $V = C_i$ except for $V_{i,j} = -C_{i,j}$ for $j \in [i-1]$ and $V_{i,i} = C_{i,i}$. That is, $\mathbf{p}'_i := \mathbf{p}'_i - \sum_{j \in [i-1]} C_{i,j}\mathbf{p}'_j = C^{-1}(A\mathbf{x}^\intercal + B\mathbf{p}^\intercal)_i$.
6. Output the codeword $\mathbf{c} := [\mathbf{m}|\mathbf{p}|\mathbf{p}']$.

Fig. 10: LDPC encoder with running time $O(n + g^2)$ where $H \in \mathbb{F}^{n \times (n-k)}$ is that code parity check matrix and approximate lower triangular with gap g.

Simplified LDPC encode circuit computing $f(\mathbf{x}) = \mathbf{x} \cdot G$ given H with $g = 0$.

1. Let $H = \begin{bmatrix} A & C \end{bmatrix}$ where $C \in \mathbb{F}^{m \times m}$. Abort if C is not lower triangular with ones on the diagonal.
2. Initialize $\mathbf{w} \in \mathbb{F}^n$ s.t. $\mathbf{w} = [\mathbf{x}|\mathbf{p}], \mathbf{p} \in \{0\}^s$ and compute
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 \\ A & 0 \end{bmatrix} \mathbf{w}^\intercal$$
i.e. $\mathbf{p}^\intercal = A\mathbf{x}^\intercal$.
3. For in order $i \in [s]$,
$$\mathbf{w}^\intercal := \begin{bmatrix} I_k & 0 \\ 0 & V \end{bmatrix} \mathbf{w}^\intercal$$
where $V = I_s$ except for $V_{i,j} = -C_{i,j}$ for $j \in [i]$. That is, $\mathbf{p}_i := -\sum_{j \in [i]} C_{i,j}\mathbf{p}_j = (C^{-1}A\mathbf{x}^\intercal)_i$.
4. Output the codeword $\mathbf{c} := [\mathbf{x}|\mathbf{p}]$.

Fig. 11: Simplified LDPC encoder with running time $O(n)$ where $H \in \mathbb{F}^{n \times (n-k)}$ is that code parity check matrix and strictly lower triangular.

Transposed LDPC encode circuit computing $f(\mathbf{e}) = \mathbf{e} \cdot G^\intercal$ given H.

1. Let $H = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}$ where $E \in \mathbb{F}^{g \times g}, C \in \mathbb{F}^{s \times s}$. Abort if C is not lower triangular with ones on the diagonal or if $E' := -FC^{-1}B + E$ is not invertible. Precompute $-E'^{-1}$.
2. Initialize $w := [\mathbf{e}||\mathbf{p}'']$ and decompose it as $\mathbf{w} = [\mathbf{x}|\mathbf{p}|\mathbf{p}'|\mathbf{p}'']$ s.t. $\mathbf{p} \in \mathbb{F}^g, \mathbf{p}' \in \mathbb{F}^s, \mathbf{p}'' \in \{0\}^s$.
3. For *reverse* order $i \in [s]$,
$$\mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & 0 & V & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix}$$
where $V = I_s$ except for $V_{i,j} = -C_{i,j}$ for $j \in [i-1]$ and $V_{i,i} = C_{i,i}$. That is, for $j \in \{j \in [i-1] \mid C_{i,j} \neq 0\}$ compute $\mathbf{p}'_j := \mathbf{p}'_j - C_{i,j}\mathbf{p}'_i$.
4. Compute
$$\mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & B & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix}, \mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & -E'^{-1} & 0 & 0 \\ 0 & 0 & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix}, \mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ D & 0 & 0 & -F \\ 0 & 0 & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix}$$
i.e. $\mathbf{p} := \mathbf{p} + \mathbf{p}'B$, then $\mathbf{p} = \mathbf{p}(-E'^{-1\intercal})$, and finally $\mathbf{x} := \mathbf{x} + \mathbf{p}D, p'' := \mathbf{p}'' - \mathbf{p}F$.
5. For *reverse* order $i \in [s]$,
$$\mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ 0 & 0 & I_s & 0 \\ 0 & 0 & U & V \end{bmatrix}$$
where $U = \{0\}^{s \times s}$ except for $U_{i,i} = 1$ and $V = C_i$ except for $V_{i,j} = -C_{i,j}$ for $j \in [i-1]$ and $V_{i,i} = 0$. That is, $\mathbf{p}'_i := \mathbf{p}'_i + \mathbf{p}''_i$ and $\mathbf{p}''_j := -C_{i,j}\mathbf{p}''_i$ for $j \in [i-1]$.
6. compute
$$\mathbf{w} := \mathbf{w} \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & I_g & 0 & 0 \\ A & 0 & I_s & 0 \\ 0 & 0 & 0 & I_s \end{bmatrix}$$
i.e. $\mathbf{x} := \mathbf{x} + \mathbf{p}'A$.
7. Output $\mathbf{x}$.

Fig. 12: LDPC encoder with running time $O(n + g^2)$ where $H \in \mathbb{F}^{n \times (n-k)}$ is that code parity check matrix and approximate lower triangular with gap g.

Note that while $H^\intercal$ might have good minimum distance, there is a chance that $H'^\intercal$ does not. For example, consider the last $s < (n-k)$ columns of H, with some probability on the first k sparse columns has non-zero elements only in the last s rows. As such, this column can be express as a linear combination of the last s columns. Depending on the weight of the columns and how large s is, this can happen with good probability. This leads us to the idea that a bad code (i.e. $H'^\intercal$) can be "patched" by setting certain positions of $\mathbf{e}'$ to be uniform. Each one of these positions effectively allows us to nullify a linear dependency in the rows of $H'^\intercal$.

In the dual setting, recall that we are computing $\mathbf{e}'G'^\intercal = \mathbf{u}'$, where G' is the generator for the parity check matrix H'. Again, we simply need to ensure that the last g positions of $\mathbf{e}'$ are uniformly distributed.

This then gives us a linear time algorithm for computing $\mathbf{e}'G'^\intercal$ which leveraging the fact that H' is strictly lower triangular. However, this comes at the expense of making the vector-OLE and silent-OT key setup slightly more expensive, i.e. $g \in \{24, 32\}$ more base OTs.

For silent OT, recall that we use t instances of a distributed point function to construct $\mathbf{e}\Delta$, with weight t. Here, $\mathbf{e} \in \{0,1\}^n$ is a binary vector with weight t while $\Delta \in \mathbb{F}$ is a field element, i.e. scaler. The parties will receive secret shares $\mathbf{u}, \mathbf{v}$ of $\mathbf{e}\Delta$ while the receiver receives $\mathbf{e}$ and the sender receives Δ. From this, they can construct the OT correlations by locally computing $\mathbf{u}G^\intercal, \mathbf{e}G^\intercal$ and $\mathbf{v}^\intercal$. Silent VOLE is effectively the same except that e has its non-zero elements uniform in $\mathbb{F}$.

Therefore, the added cost of this approach is creating this correlation for $\mathbf{e}^*\Delta \in \mathbb{F}^g$ and the we do the same computation except with $\mathbf{e}' = [\mathbf{e}|\mathbf{e}^*]$ instead of $\mathbf{e}$. In the Silent OT case, $\mathbf{e}^*$ can be computed with g OTs. Computing shares of $\mathbf{e}$ along requires approximately $t \log_2 n \approx 1200$ OTs. Here, t is the weight of e and depends on the parameters used. Therefore if we set $g \leq t \log_2 n$, the overall communication overhead is at most $2\times$.

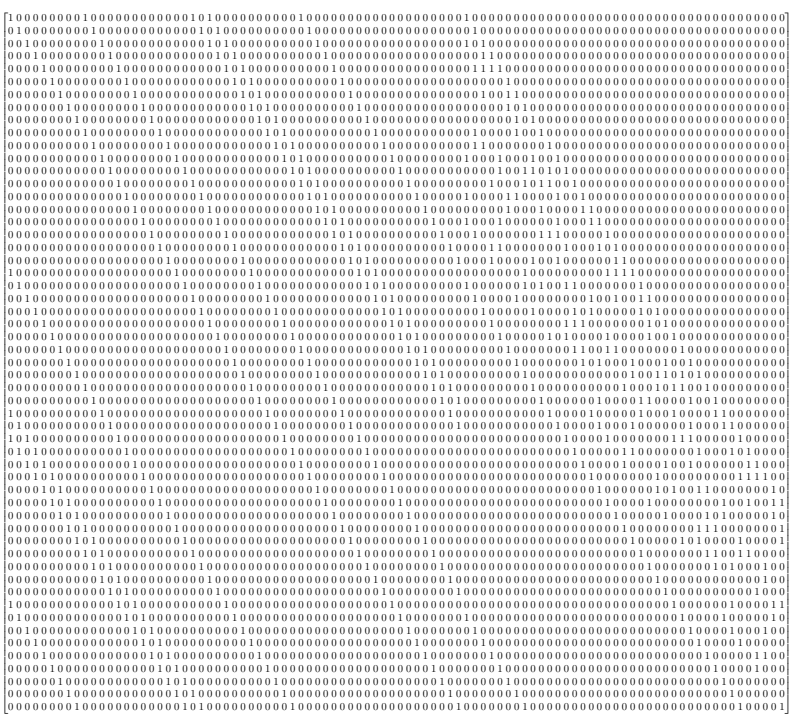
F Silver Code Details

We node give a concrete instantiating of our Slv5 code with column weight $t \in \{5, 11\}$. The left $m \times m$ submatrix L consists of weight t columns where each is a cyclic shift by 1 of the previous. For $t = 5$ the first column has non-zero locations rows $\{0, 0.372071, 0.576568, 0.608917, 0.854475\} \cdot m$ while for $t = 11$ the first column is $\{0, 0.00278835, 0.0883852, 0.238023, 0.240532, 0.274624, 0.390639, 0.531551, 0.637619, 0.945265, 0.965874\} \cdot m$. For small m, if two non-zero locations collide we insert into the next non-zero position.

We define the non-extended version of the right $m \times m - g$ matrix R by the following square submatrices. For $t = 5$ (left) and $t = 11$ (right) we have

$$\begin{bmatrix}
0 & 4 & 11 & 15 \\
0 & 8 & 9 & 10 \\
1 & 2 & 10 & 14 \\
0 & 5 & 8 & 15 \\
3 & 13 & 14 & 15 \\
2 & 4 & 7 & 8 \\
0 & 9 & 12 & 15 \\
1 & 6 & 8 & 14 \\
4 & 5 & 6 & 14 \\
1 & 3 & 8 & 13 \\
3 & 4 & 7 & 8 \\
3 & 5 & 9 & 13 \\
8 & 11 & 12 & 14 \\
6 & 10 & 12 & 13 \\
2 & 7 & 8 & 13 \\
0 & 6 & 10 & 15
\end{bmatrix}
\qquad
\begin{bmatrix}
6 & 7 & 8 & 12 & 16 & 17 & 20 & 22 & 24 & 25 \\
0 & 1 & 6 & 10 & 12 & 13 & 17 & 19 & 30 & 31 \\
1 & 4 & 7 & 10 & 12 & 16 & 21 & 22 & 30 & 31 \\
3 & 5 & 9 & 13 & 15 & 21 & 23 & 25 & 26 & 27 \\
3 & 8 & 9 & 14 & 17 & 19 & 24 & 25 & 26 & 28 \\
3 & 11 & 12 & 13 & 14 & 16 & 17 & 21 & 22 & 30 \\
2 & 4 & 5 & 11 & 12 & 17 & 22 & 24 & 30 & 31 \\
5 & 8 & 11 & 12 & 13 & 17 & 18 & 20 & 27 & 29 \\
13 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 26 & 30 \\
3 & 8 & 13 & 15 & 16 & 17 & 19 & 20 & 21 & 27 \\
0 & 2 & 4 & 5 & 6 & 21 & 23 & 26 & 28 & 30 \\
2 & 4 & 6 & 8 & 10 & 11 & 22 & 26 & 28 & 30 \\
7 & 9 & 11 & 14 & 15 & 16 & 17 & 18 & 24 & 30 \\
0 & 3 & 7 & 12 & 13 & 18 & 20 & 24 & 25 & 28 \\
1 & 5 & 7 & 8 & 12 & 13 & 21 & 24 & 26 & 27 \\
0 & 16 & 17 & 19 & 22 & 24 & 25 & 27 & 28 & 31 \\
0 & 6 & 7 & 15 & 16 & 18 & 22 & 24 & 29 & 30 \\
2 & 3 & 4 & 7 & 15 & 17 & 18 & 20 & 22 & 26 \\
2 & 3 & 9 & 16 & 17 & 19 & 24 & 27 & 29 & 31 \\
1 & 3 & 5 & 7 & 13 & 14 & 20 & 23 & 24 & 27 \\
0 & 2 & 3 & 9 & 10 & 14 & 19 & 20 & 21 & 25 \\
4 & 13 & 16 & 20 & 21 & 23 & 25 & 27 & 28 & 31 \\
1 & 2 & 5 & 6 & 9 & 13 & 15 & 17 & 20 & 24 \\
0 & 4 & 7 & 8 & 12 & 13 & 20 & 23 & 28 & 30 \\
0 & 3 & 4 & 5 & 8 & 9 & 23 & 25 & 26 & 28 \\
0 & 3 & 4 & 7 & 8 & 10 & 11 & 15 & 21 & 26 \\
5 & 6 & 7 & 8 & 10 & 11 & 15 & 21 & 22 & 25 \\
0 & 1 & 2 & 3 & 8 & 9 & 22 & 24 & 27 & 28 \\
1 & 2 & 13 & 14 & 15 & 16 & 19 & 22 & 29 & 30 \\
2 & 14 & 15 & 16 & 19 & 20 & 25 & 26 & 28 & 29 \\
8 & 9 & 11 & 12 & 13 & 15 & 17 & 18 & 23 & 27 \\
0 & 2 & 4 & 5 & 6 & 7 & 10 & 12 & 14 & 19
\end{bmatrix}$$

Each row specifies the non-zero locations at a relative offset of g positions left of the main diagonal, where g is $16, 32$ respectively. In particular, this will result in each (non-edge case) row/column having exactly t non-zero positions. This is then augmented with the additional width 1 non-zero diagonal located a positions $5, 31$ below this main diagonal.For example, with $m = 55$ and $t = 5$ we obtain



Alternative instantiations based on our methodology can be obtained from the source code located at [Rin] along with our implementation of the noise impulse method.