# Encrypted Matrix-Vector Products from Secret Dual Codes*

Fabrice Benhamouda† Caicai Chen‡ Shai Halevi§ Yuval Ishai¶

Hugo Krawczyk‖ Tamer Mour† Tal Rabin** Alon Rosen†

November 13, 2025

**Abstract**

Motivated by applications to efficient secure computation, we consider the following problem of *encrypted matrix-vector product* (EMVP). Let $\mathbb{F}$ be a finite field. In an offline phase, a client uploads an encryption of a matrix $\mathbf{M} \in \mathbb{F}^{m \times \ell}$ to a server, keeping only a short secret key. The server stores the encrypted matrix $\widehat{\mathbf{M}}$. In the online phase, the client may repeatedly send encryptions $\widehat{\mathbf{q}}_i$ of query vectors $\mathbf{q}_i \in \mathbb{F}^\ell$, which enables the client and the server to locally compute compact shares of the matrix-vector product $\mathbf{M}\mathbf{q}_i$. The server learns nothing about $\mathbf{M}$ or $\mathbf{q}_i$. The shared output can either be revealed to the client or processed by another protocol.

We present efficient EMVP protocols based on variants of the *learning parity with noise* (LPN) assumption and the related *learning subspace with noise* (LSN) assumption. Our EMVP protocols are *field-agnostic* in the sense that the parties only perform arithmetic operations over $\mathbb{F}$, and are close to optimal with respect to both communication and computation. In fact, for sufficiently large $\ell$ (typically a few hundreds), the online computation and communication costs of our LSN-based EMVP can be *less than twice* the costs of computing $\mathbf{M}\mathbf{q}_i$ in the clear.

Combined with suitable secure post-processing protocols on the secret-shared output, our EMVP protocols are useful for a variety of secure computation tasks, including encrypted fuzzy search and secure ML.

Our technical approach builds on recent techniques for private information retrieval in the secret-key setting. The core idea is to encode the matrix $\mathbf{M}$ and the queries $\mathbf{q}_i$ using a pair of secret dual linear codes, while defeating algebraic attacks by adding noise.

*This is the full version of [BCH+25].

†Amazon Web Services, fabrice.benhamouda@gmail.com.

‡Bocconi University, BIDSA, {caicai.chen, tamer.mour, alon.rosen}@unibocconi.it. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

§Amazon Web Services, shai.halevi@gmail.com.

¶Technion and Amazon Web Services, yuval.ishai@gmail.com. Work done while at AWS.

‖Amazon Web Services, hugokraw@gmail.com.

**Amazon Web Services, rabintal@amazon.com.

# Contents

# 1 Introduction

Secure multiplication of a secret matrix by a secret vector is a useful primitive in many cryptographic applications. For example, recent systems for identifying uniqueness of humans use this primitive to find an approximate match between a new applicant and a database of existing users [ELS+24, BKSW24]. Here each row of the matrix is a feature vector of an existing user and the matrix-vector product computes the similarity between the applicant and each existing user. Since the feature vectors may include highly sensitive information such as biometric data, both the matrix and the vector should remain hidden. The shared matrix-vector product is then used by a secure post-processing protocol to extract some short digest, such as whether the new applicant is similar to an existing user. However, since the matrix-vector product is much smaller than the matrix, the cost of this post-processing step is often dominated by computing the matrix-vector product.

To practically support large-scale instances of the problem, with matrices containing thousands of columns and millions of rows, a recent solution adopted by World ID [BKSW24] opted for a 3-server architecture where both the matrix and the query vectors are distributed using a 2-out-of-3 secret-sharing scheme. While offering impressive performance, it relies on a weaker trust model that assumes no two servers can collude. Increasing the resilience of this solution (which is desirable given the sensitive nature of biometric information) incurs a high cost in storage and performance.

Another approach is to use homomorphic encryption (HE), as done for example in the Janus system [ELS+24]. An HE-based solution enjoys a better trust model, as the large encrypted matrix can be stored and processed by a single untrusted server. One may still want to distribute the homomorphic decryption function in order to enable queries by multiple parties or simply to avoid a single point of failure. Since the decryption function is relatively simple, the latter distributed computation can be made quite efficient and offer flexibility in the decryption threshold. However, the analysis in [BKSW24] indicates that the concrete costs of standard HE-based solutions are impractical for use in these large instances.

Can we provide a solution to matrix-vector multiplication that achieves the *best of both worlds*: the performance of the 3-server solution and the trust model and flexibility of HE-based solutions?

**Encrypted matrix-vector products.** To answer the above question, we initiate a systematic study of Encrypted Matrix-Vector Product (EMVP) protocols in a client-server model, where the client can preprocess and upload an encrypted matrix and then securely query the server to obtain the product of that matrix by any number of vectors. Our goal is to provide lightweight solutions that perform as close as possible to the insecure baseline.

To frame the EMVP problem more formally, let $\mathbb{F}$ be a finite field. A client holds a matrix $\mathbf{M} \in \mathbb{F}^{m \times \ell}$ and wants to outsource its storage and computation to a server. In an offline phase, the client uploads an encryption $\widehat{\mathbf{M}}$ of $\mathbf{M}$ to the server and retains only a short secret key. Later, in the online phase, the client repeatedly sends encrypted queries $\widehat{\mathbf{q}}_i$ corresponding to vectors $\mathbf{q}_i \in \mathbb{F}^{\ell}$, and both parties *locally* compute compact shares of the product $\mathbf{M}\mathbf{q}_i$. The server's share can either be sent to the client to recover the answer to the query, or post-processed using HE or another secure computation protocol. Crucially, the server learns nothing about the matrix $\mathbf{M}$ or the queries $\mathbf{q}_i$. How efficient can such an EMVP protocol be?

## 1.1 Our Results

We propose EMVP protocols that rely on the better trust model of the HE-based solutions while matching or even outperforming the concrete costs of the 3-server solution from [BKSW24]. Our protocols build on recent techniques for private information retrieval (PIR) in the secret-key setting

from [BIPW17, CHR17, CIMR25], and are based on "LPN-style" cryptographic assumptions that are not known to imply public-key encryption or even collision-resistant hashing.

A qualitative advantage of our protocols over existing HE-based solutions is that they are *field-agnostic* in the sense of only making a black-box use of the underlying field $\mathbb{F}$ [NP99, IPS09, AAB17]. This feature strongly correlates with concrete efficiency, and makes it easier to distribute both the client's query generation[1] and the post-processing of the output. We will further discuss this below.

Our efficient EMVP protocols can serve as a lightweight alternative to homomorphic encryption in the context of matrix-vector products, making them a useful building block in a wide array of cryptographic applications. In particular, the special case where the $\mathbf{q}_i$'s are standard basis vectors can be used to implement PIR [CGKS95, KO97]. More broadly, EMVP protocols enable secure delegation of encrypted fuzzy search, machine learning inference, and other linear algebra tasks over encrypted data. See Section 1.3.

More concretely, our EMVP protocols have the following features:

- Support matrix-vector products over arbitrary fields $\mathbb{F}$ (or even other useful rings), while being *field-agnostic* in the above sense.
- Require only a short client-side secret key and no additional state beyond a static encoding of $\mathbf{M}$ stored by the server. The client's key does not depend on $\mathbf{M}$ and can be reused.
- The communication and client computation are close to $\ell + m$, the cost of an insecure solution.
- The total computational cost is close to $\ell m$, the cost of an insecure solution.

This near-optimality holds concretely for some of our EMVP protocols, and asymptotically for all of them. In concrete terms, our EMVP solutions have much better costs than HE-based approaches. To give just a couple of data points, based on our security analysis: with the moderate row length $\ell = 1024$, we can compress the communication by a factor of 14 (vs. sending the entire matrix) while increasing the server's storage and computational work by a mere factor of 1.25 over the insecure solution where everything is done in the clear. With longer rows, setting $\ell = 10000$ as in [BKSW24], we get a compression ratio of 177× with the same overhead factor.

This level of efficiency is based on a new variant of previous assumptions, discussed below, which we introduce and analyze in this work. The communication can be further compressed by composing our EMVP protocols with high-rate HE schemes, improving the communication cost without a significant impact on computation. See discussion at the end of Section 4.1.

**Beyond a designated client.** Our basic EMVP protocols operate in a setting where we have a single data owner (client), who stores the encrypted matrix in the cloud and is the only one querying it. In this setting, which is the standard setting for searchable symmetric encryption [SWP00], the client has a secret key which is used to encrypt the matrix and the queries and to decrypt the responses. However, one would like to also consider more general settings in which the matrix is not owned by a single client, and multiple parties may want to query or update the encrypted matrix. One option is to distribute the client, and this is where field-agnostic protocols really shine. This property of our EMVP protocols makes the query generation and output decoding very easy

[1] The only operational advantage of HE with distributed decryption over our schemes is the ability of parties to encrypt queries without going through the distributed client. In some cases, however, distributed encryption may be a desirable feature, since sending encrypted shares of a query to one of the parties could be more efficient than encrypting the query with HE.

to distribute using standard secure computation techniques that only natively support arithmetic operations over $\mathbb{F}$. We present lightweight protocols for this distributed setting in Section 6.

A different approach for accommodating multiple clients, without making non-collusion assumptions or increasing the server storage, is by allowing the server to learn the matrix $\mathbf{M}$ (but not the queries $\mathbf{q}_i$) and allowing each client to store a small amount of information, or a "hint," about $\mathbf{M}$. This is similar to prior hint-based protocols for PIR and private semantic search [HHC+23, DPC23, HDCZ23], with the difference that the latter protocols can use a single global hint, whereas our approach requires each client to generate its own hint and keep it private. We elaborate on this flavor of EMVP in Appendix D.

### 1.2 The Underlying Cryptographic Assumptions

Our protocols rely on different flavors of "learning subspace with noise" (LSN) assumptions that were first introduced by Dodis et al. [DKL09] in the context of leakage-resilient cryptography and revisited by Chen et al. [CIMR25] in the context of PIR with preprocessing. These can be seen as less structured variants of the ad-hoc assumption underlying previous constructions of doubly efficient secret-key PIR from permuted Reed-Muller codes [BIPW17, CHR17]. In LSN assumptions, we have a *secret* linear code $C \subseteq \mathbb{F}^n$, and the assumptions assert that *polynomially many* "noisy" random codewords are pseudorandom, where the noise can either replace a codeword by a different vector or plant it in a low-dimensional subspace. This can be compared to the classic Learning Parity with Noise (LPN) assumption [BFKL94], asserting that a *single* noisy codeword in a random *public* code is pseudorandom. The different flavors of LSN vary mainly in the type of noise applied to codewords. As shown in [DKL09, CDV21, CIMR25], there are relations between LSN and LPN, where some LSN variants are implied by LPN, and most variants imply LPN.

Unlike standard homomorphic encryption or PIR in the plain model [DMO00, IKO05], LSN-style assumptions are not known to imply public-key encryption or even collision-resistant hashing. We explore both previous and new variants of LSN, describing cryptanalytical results and various trade-offs between communication, computation, and assumptions.

The most efficient instances of our EMVP protocol rely on a new variant of LSN, dubbed 1-dimensional split-LSN (1D-SLSN), in which noise is added by splitting each sampled codeword into a small number of blocks and multiplying each block by a secret nonzero scalar. Unlike the original flavor of LSN from [DKL09], which can be broken in quasi-polynomial time (when the code rate is any constant $\rho < 1$), we conjecture our new variant to be secure against sub-exponential time attacks. This and other variants of LSN certainly require further study, and in Section 7 we have made an initial cryptanalysis effort that guides our proposed parameters. While making new assumptions is always risky, in this case there seems to be substantial evidence supporting the hardness assumptions that we use:

First, we can use the same blueprint to get an efficient EMVP protocol that relies on just the standard LPN assumption (over general fields) in a well-studied parameter regime. That protocol has the same qualitative features as our most efficient protocols: it is field-agnostic while achieving near-optimal *asymptotic* (but not concrete) communication, storage and computation. A similar result was recently obtained in the context of secret-key PIR with preprocessing [CIMR25]. A key difference is that the PIR server's computational cost (in the RAM model) can be dramatically improved using a lattice-based approach [LMW23]. In contrast, our LPN-based solution for EMVP seems[2] close to optimal even under this metric.

[2]Strong unconditional lower bounds for data structure problems are notoriously hard. However, it is known that, for sufficiently large $\mathbb{F}$, when preprocessing $\mathbf{M} \in \mathbb{F}^{\ell \times \ell}$ into a polynomial-size $\widehat{\mathbf{M}} \in \mathbb{F}^L$, the server needs to read $\tilde{\Omega}(\ell^2)$ entries from $\widehat{\mathbf{M}}$ to answer a general query $\mathbf{Mq}$ even without any security requirements [CGL15].

Second, the basic LSN problem, where the noise consists of replacing a sampled codeword by a random vector with probability $\mu < 1$, was already studied in [DKL09]. It is an instance of the well-studied problem of learning mixtures of simple distributions, which in many cases is conjectured to be intractable. The search version of the basic variant of LSN was studied in [CDV21] and further progress on this problem may be of interest to the learning theory community.

Finally, the LSN-style assumptions we employ for our concretely efficient protocols, including 1D-SLSN, can be viewed as less structured variants of the assumptions underlying previous constructions of PIR from secretly permuted Reed-Muller codes [BIPW17, CHR17]. The latter have already withstood some analysis [BHW19, BW21, BHMW21]. We hope that the combination of compelling practical motivation and clean theory-oriented questions will encourage future analysis of LSN-style assumptions.

**Security fallbacks.** There are two ways to make our security assumptions even more conservative without significantly hurting efficiency. First, when a client can batch multiple queries $\mathbf{q}_i$, shuffling different sub-queries together is expected to enhance security [IKOS06, GIK+24]. Second, adding a small amount of Gaussian noise to the query vectors can serve as an extra security measure, especially in fuzzy search applications where noise is inherent. This is reminiscent of modern approaches for approximate homomorphic encryption [CKKS17].

### 1.3 Related Work

**Secret-key PIR with preprocessing.** Our work heavily builds on the technical approach of prior works on PIR with preprocessing in the secret-key setting [BIPW17, CHR17, CIMR25], and in particular the split-LSN assumption recently put forward in [CIMR25]. Beyond adapting their technique to the EMVP setting, our work contains many new optimizations. These include new variants of the split LSN assumption tailored to the case of EMVP over big fields, new techniques for minimizing the complexity of EMVP clients and servers, efficient protocols for distributing EMVP clients, and a concrete analysis of the underlying assumptions with respect to relevant classes of algebraic attacks. In fact, by using a standard reduction of PIR to matrix-vector product [CGKS95], our optimized EMVP protocols and concrete analysis of LSN-type assumptions are directly relevant also to this use case of EMVP.

In particular, in Appendix C we apply EMVP over $\mathbb{F}_4$ to improve the Boolean circuit complexity of secret-key PIR. Concretely, the most efficient construction from [CIMR25], on an $N$-bit database, requires a server circuit of size $(4 + o(1))N$, with $(1 + o(1))N$ bits of server storage, under an LSN-style assumption over $\mathbb{F}_2$. Here we use our new 1D-SLSN assumption over $\mathbb{F}_4$ to improve the Boolean circuit size to $(3.5 + o(1))N$, or alternatively to $(3 + o(1))N$ with 50% extra server storage.

**Trapdoored matrices.** Two recent independent works [BN25, VZ25] propose the idea of generating pseudorandom matrices $\mathbf{R}$ such that the linear map $\mathbf{v} \mapsto \mathbf{Rv}$ can be computed in near-linear time given a suitable trapdoor.[3] The motivation in [BN25] is similar to ours: allowing a weak client to delegate a linear algebra computation (such as matrix product) to a semi-honest server without revealing the input. However, the solution from [BN25] inherently requires the client to store the input matrices, whereas the EMVP client only needs to store a short secret key. Low-communication EMVP is nontrivial even without imposing any computational restrictions on the client, whereas the delegation problem considered in [BN25] is only meaningful with such restrictions. Still, trapdoored matrices are a useful building block for EMVP. In fact, by masking $\mathbf{M}$ with a trapdoored

[3] A similar idea appears in an earlier work by Sotiraki [Sot16], where a weaker security notion is considered.

matrix, one can convert any EMVP protocol that only protects the privacy of the queries $\mathbf{q}_i$ but not the matrix $\mathbf{M}$, such as one based on additively homomorphic encryption, to a fully secure EMVP with an efficient client. In Section 5 we propose new candidate constructions of trapdoored matrices that offer simplicity or efficiency advantages over the ones from [BN25, VZ25] and that are easier to implement in a distributed client setting. In particular, we give the first *linear-time* TDM candidates.

**Pseudorandom correlation generators.** A recent line of work on pseudorandom correlation generators (PCGs) [BCGI18, BCG$^+$19b] uses LPN-style assumptions to compress simple but useful multi-party correlations. Similarly to our work, such PCG constructions are typically field-agnostic. However, existing PCG-based secure computation protocols do not meet our efficiency constraints. In particular, practical PCGs only support *additive* correlations, such as additively shared multiplication triples [Bea91]. Protocols that consume such correlations require parties to play a symmetric role. In the EMVP context, this means that both parties need to hold a masked copy of $M$ and multiply it by a masked $q$. In contrast, our protocols are highly asymmetric, allowing the client to delegate the bulk of the work to the server. This is particularly useful when distributing the client via secure computation. Using PCGs in our distributed client protocols, we can maintain the field-agnostic feature of our centralized-client solutions even when the client is distributed.

**Searchable symmetric encryption.** The basic scenario we address in this paper, namely, data that is encrypted with a symmetric key and outsourced to the cloud for later querying, can be seen as analogous to the setting of searchable symmetric encryption (SSE) [SWP00, CGKO06]. However, while SSE has mostly focused on lexical search (such as keyword, Boolean, and range queries), EMVP can support semantic queries that often provide better insights. In particular, fuzzy search on vector embeddings can be used to query unstructured data such as images and biometrics. Another key difference is that SSE trades efficiency for leakage, for example achieving sublinear server time while leaking the *access patterns* of queries and results. Our EMVP-based solutions require linear work at the server (which is sometimes also the case for the insecure baseline) but avoid the above leakage and apply to a much wider set of applications. As examples, we can support private remote biometrics (as in [ELS$^+$24, BKSW24]) or provide efficient semantic search functionality (as in [HDCZ23]) on encrypted documents while dispensing with homomorphic encryption.

**Homomorphic encryption.** As discussed above, our techniques can be easily extended to support the delegation and decentralization of query issuing and data storage via efficient distributed clients (Section 6). In this case, our EMVP protocols can realize much of the benefits of homomorphic encryption, particularly when noting that many of the (F)HE applications require a distributed client for the decryption of query results (except if only the owner of the data and the queries receives the results). In use cases of HE in which the EMVP functionality suffices, our solutions may offer orders of magnitude speedup compared to HE techniques. Even in the easier case of multiplying an unencrypted matrix by an encrypted vector, recent HE-based solutions (see [HYT$^+$24] and references therein) are significantly slower than our EMVP protocol and require much wider matrices to attain their maximal level of efficiency.

**Privacy-preserving machine learning.** Computing linear transformations on vector embeddings (that is, matrix vector products) is at the heart of machine learning (ML) techniques. Our solutions allow for processing such transformations where both the matrix and vector are encrypted. In particular, this is the case for ML inference where both the model and the query are encrypted,

an example of a setting where FHE would have been previously needed. Furthermore, our schemes lead to more efficient solutions even when compared to private inference on plaintext models (and encrypted queries) as in [JVC18, PZM+24] and many other works. Note that applying a multilayer neural network in our case requires interaction between the server and the querier for each iteration of the network (for decryption and re-encryption of the previous layer result). This per-iteration interaction has also been part of previous work, where it was needed to efficiently handle nonlinear elements in the neural layers (even in solutions where the model is in the clear). Here we can combine the decryption/re-encryption with computing these nonlinear functions.

### 1.4 Brief Technical Overview

At the heart of our construction is a simple but powerful technique that was previously used for PIR in a secret-key setting [BIPW17, CHR17, CIMR25]: represent matrix rows as codewords in a secret linear code, and construct encrypted queries using "noisy" shifted dual codewords. Unlike standard noisy linear algebra techniques in cryptography, here the noise is typically in the form of planting the shifted dual codeword in a low-dimensional space. The server's response is a highly compressed matrix, which together with decoding information generated by the client can be used to recover $\mathbf{Mq}_i$ either by the client or by a secure post-processing protocol. Our careful noise design is intended to defeat known algebraic attacks while preserving correctness and decoding efficiency. The security of this approach reduces to different flavors of the LSN assumption discussed above.

We analyze several variants of the LSN assumption, including ones previously put forward in [DKL09, CIMR25] and new variants that are tailored to the case of large fields or rings. We provide evidence for their plausibility by analyzing security against natural classes of algebraic attacks, and demonstrate how to instantiate our protocols under each. The main algebraic attack that we analyze is one that tries to find a nonzero low-degree polynomial that vanishes on all of the noisy LSN samples. This attack, attributed in [DKL09] to Raz, breaks in quasi-polynomial time the basic variant of LSN (with constant code rate $k/n < 1$), where each sampled codeword is replaced by a random vector with some probability $\mu < 1$ such that $1 - \mu$ is noticeable. Our analysis shows that by planting each codeword in a low-dimensional linear space, we can get security against algebraic attacks of this kind that run in sub-exponential time. The latter flavor of LSN serves as the basis for our most efficient constructions. To maximize efficiency, we plant each codeword in a *product space*. Concretely, we split each codeword into a small number of blocks and multiply each block by a secret nonzero scalar. The above algebraic attack is the best attack we are aware of in most parameter regimes, assuming there are no restrictions on the number of samples available to the attacker.

We also show that, using only the standard LPN assumption (over general fields), EMVP can be realized without requiring additively homomorphic encryption while retaining the qualitative advantage of being field-agnostic. This conservative EMVP protocol has near-optimal asymptotic efficiency, but is much worse in terms of concrete efficiency than our LSN-based solutions. Our LPN-based EMVP protocol follows the technical approach of the recent LPN-based secret-key PIR construction from [CIMR25], using LSN with a special noise distribution. We further attempt to optimize the concrete efficiency of this construction, and provide some data points in Table 2.

In the sections that follow, we formalize the EMVP model, define and analyze the relevant assumptions, and present our constructions and optimizations in detail.

## 2 Preliminaries

**Linear algebra.** We will use $\mathbb{F}$ to denote a finite field, typically $\mathbb{F} = \mathbb{F}_p$ for a large prime $p$. We will also discuss an extension of our protocols to more general rings $R$, including the rings $R = \mathbb{Z}_{2^k}$ that are natively supported by standard computer architectures. We use boldface letters for matrices and vectors and capital letters for linear spaces. For instance, $\mathbf{C} \in \mathbb{F}^{k \times n}$ will denote a generating matrix of a linear code $C \subseteq \mathbb{F}^n$. We use $(a \,|\, b)$ and $[A \,|\, B]$ for horizontal concatenation of vectors and matrices and $(a \,//\, b)$ and $[A \,//\, B]$ for vertical concatenation. For matrices $A \in \mathbb{F}^{k \times \ell}$ and $B \in \mathbb{F}^{m \times n}$, we denote by $A \otimes B \in \mathbb{F}^{km \times \ell n}$ the standard Kronecker product of $A$ and $B$, which is defined by $(A \otimes B)_{i,j} = A_{\lceil i/m \rceil, \lceil j/n \rceil} \cdot B_{i\%m, j\%n}$.

**Probability.** We use $x \leftarrow X$ to denote a uniformly random choice of $x$ from a set $X$. We denote by $\mathsf{Ber}(\mu)$ a Bernoulli random variable which takes the value 1 with probability $\mu$ and 0 with probability $1 - \mu$. More generally, for a finite field $\mathbb{F}$, the Bernoulli variable $\mathsf{Ber}_{\mathbb{F}}(\mu)$ takes the value 0 with probability $1 - \mu$ and is otherwise uniformly random in $\mathbb{F} \setminus \{0\}$.

**Cryptography.** We use the standard notions of negligible functions, computational indistinguishability (denoted $X \approx_c Y$), and pseudorandom functions (PRFs) [Gol01].

### 2.1 Encrypted Matrix-Vector Product

An EMVP protocol starts with an offline preprocessing phase, in which a client encrypts a matrix $\mathbf{M} \in \mathbb{F}^{m \times \ell}$ using a secret key $\mathsf{sk}$, and sends the encrypted matrix $\widehat{M}$ to a server. In the online phase, which can be invoked many times, the client can securely compute matrix-vector products $\mathbf{Mq}$ for any chosen $\mathbf{q}$, where the outputs are split between the client and the server. This is done by having the client map $(\mathsf{sk}, \mathbf{q})$ to a pair $(\widehat{q}, q')$ and send the encrypted query $\widehat{q}$ to the server. The server, on inputs $(\widehat{M}, \widehat{q})$ computes a compact encoded matrix $M'$. The output $a = \mathbf{Mq}$ can be decoded from $(M', q')$ using a low-complexity decoding algorithm that can either be performed locally by the client or distributed by a higher-level application.

We say that the EMVP protocol is *field-agnostic* if it only makes a black-box use of the field, in the sense that field elements are labeled arbitrarily and all algorithms have oracle access to the field operations [IPS09, AAB17]. In fact, most of our protocols can be cast in the stricter black-box model from [AAB17] that does not even allow the protocol to know an upper bound on the field size. In particular, the number of field operations required by our protocols is independent of the field size. Our default set of field operations include addition, subtraction, multiplication, unit, inverse, zero-test, and sampling of random field elements.

In the following definition, adapted from earlier definitions of secret-key PIR [BIPW17, CHR17, CIMR25], we consider both a general version and a field-agnostic version. In the latter we view the field $\mathbb{F}$ as an implicit parameter and assume all algorithms have oracle access to basic field operations. In fact, some instances of our protocols do not require zero-test and inverse, and can apply over rings such as $\mathbb{Z}_{2^k}$.

**Definition 2.1** (Encrypted Matrix-Vector Product)**.** *An* encrypted matrix-vector product (EMVP) *protocol is a tuple of PPT algorithms* $\mathsf{EMVP} = (\mathsf{G}, \mathsf{E}, \mathsf{Q}, \mathsf{A}, \mathsf{D})$ *with the following syntax:*

- $\mathsf{G}(\mathbb{F}, 1^\lambda, 1^m, 1^\ell)$*: The* key generation algorithm *takes a description of a field* $\mathbb{F}$*, security parameter* $\lambda$ *and matrix dimensions* $m, \ell$*, and returns a* secret key $\mathsf{sk}$*. We assume that* $\mathsf{sk}$ *contains* $\mathbb{F}, \lambda, m, \ell$*. In the field-agnostic case,* $\mathbb{F}$ *is not given as input and is not part of* $\mathsf{sk}$*.*

- $\mathsf{E}(\mathsf{sk}, \mathbf{M})$*: The* matrix encryption algorithm *takes a secret key* $\mathsf{sk}$ *and a matrix* $\mathbf{M} \in \mathbb{F}^{m \times \ell}$ *and returns an encrypted matrix* $\widehat{M}$*. In the field-agnostic case,* $\widehat{M} \in \mathbb{F}^{\widehat{m} \times \widehat{\ell}}$*. In this case we will use boldface font for* $\widehat{\mathbf{M}}$*.*
- $\mathsf{Q}(\mathsf{sk}, \mathbf{q})$*: The* query algorithm *takes a secret key* $\mathsf{sk}$ *and a query vector* $\mathbf{q} \in \mathbb{F}^{\ell}$*, and returns a pair* $(\widehat{q}, q')$*, where* $\widehat{q}$ *is the* encrypted query *and* $q'$ *is a (short)* decoding key*. In the field-agnostic case we have* $\widehat{\mathbf{q}} \in \mathbb{F}^{\widehat{\ell}}$ *and* $\mathbf{q}' \in \mathbb{F}^{\ell'}$ *(where typically* $\ell' \ll \ell$*).*
- $\mathsf{A}(\widehat{M}, \widehat{q})$*: The* answer algorithm *takes an encrypted matrix* $\widehat{M}$ *and an encrypted query* $\widehat{q}$ *and returns a (compact)* encoded output $M'$*. In the field-agnostic case we have* $\mathbf{M}' \in \mathbb{F}^{m' \times \ell'}$*.*
- $\mathsf{D}(\mathsf{sk}, M', q')$*: The* decoding algorithm *takes secret key* $\mathsf{sk}$*, encoded output* $M'$ *and decoding vector* $q'$*, and returns an output vector* $\mathbf{a} \in \mathbb{F}^{m}$*.*

***Correctness.*** *For any field* $\mathbb{F}$*,* $\lambda, m, \ell \in \mathbb{N}$*,* $\mathsf{sk} \in \mathsf{G}(\mathbb{F}, 1^{\lambda}, 1^{m}, 1^{\ell})$*,* $\mathbf{M} \in \mathbb{F}^{m \times \ell}$*,* $\widehat{M} \in \mathsf{E}(\mathsf{sk}, M)$*,* $\mathbf{q} \in \mathbb{F}^{\ell}$*,* $(\widehat{q}, q') \leftarrow \mathsf{Q}(\mathsf{sk}, \mathbf{q})$*, and* $M' \in \mathsf{A}(\widehat{M}, \widehat{q})$*, we have* $\mathsf{D}(\mathsf{sk}, M', q') = \mathbf{Mq}$*.*

***Security.*** *For any non-uniform polynomial-time algorithm* $\mathcal{A}$ *that makes queries to its oracle, and any polynomials* $m := m(\lambda)$ *and* $\ell := \ell(\lambda)$ *and field* $\mathbb{F} := \mathbb{F}(\lambda)$*, there is a negligible function* $\delta$ *such that for any* $\lambda \in \mathbb{N}$ *and* $\mathbf{M} \in \mathbb{F}^{m \times \ell}$*, it holds that*

$$\left| \Pr[\mathcal{A}^{\mathsf{Q}(\mathsf{sk}, \cdot)}(1^{\lambda}, \widehat{M}) = 1] - \Pr[\mathcal{A}^{\mathsf{Q}_0(\mathsf{sk}, \cdot)}(1^{\lambda}, \widehat{M}_0) = 1] \right| \leq \delta(\lambda),$$

*where* $\mathsf{sk} \leftarrow \mathsf{G}(\mathbb{F}, 1^{\lambda}, 1^{m}, 1^{\ell})$*,* $\widehat{M} \leftarrow \mathsf{E}(\mathsf{sk}, \mathbf{M})$*,* $\widehat{M}_0 \leftarrow \mathsf{E}(\mathsf{sk}, \mathbf{0})$*, and* $\mathsf{Q}_0(\mathsf{sk}, \cdot)$ *is an oracle that ignores its input and outputs* $\mathsf{Q}(\mathsf{sk}, 0^{\ell})$ *(with fresh randomness).*

***Field-agnostic EMVP.*** *We say that* EMVP *is* field-agnostic *if all of the above algorithms only require oracle access to* $\mathbb{F}$*, where field elements have arbitrary labels and the field oracle is used to perform field operations.*

Given the security requirement of EMVP, we can view the intermediate values $(M', q')$ generated by the protocol as a 2-out-of-2 (nonlinear and compact) secret-sharing of the output $\mathbf{Mq}$ between the server and the client. In our main instantiation, we will have $\mathbf{M}' \in \mathbb{F}^{m \times s}$ for $s \ll \ell$ and $\mathbf{q}' = (\mathbf{p}', \mathbf{r}')$ with $\mathbf{p}' \in \mathbb{F}^{s}$ and $\mathbf{r}' \in \mathbb{F}^{m}$, where $\mathbf{Mq} = \mathbf{M}'\mathbf{p}' - \mathbf{r}'$. Decoding the output from the shares can either be done locally by the client, or performed by a post-processing protocol that has better efficiency or different functionality.

## 2.2 Learning Parity with Noise

We recall the *learning parity with noise* (LPN) problem over a general field $\mathbb{F}$ [BFKL94, IPS09]. LPN is parameterized by a *dimension* parameter $k := k(\lambda)$, a *noise rate* $\varepsilon := \varepsilon(\lambda) \in (0, 1)$ and a *number of samples* given to the adversary which we denote by $m := m(\lambda)$.

**Definition 2.2** (LPN)**.** *The* (decisional) learning parity with noise *assumption* $(k, \varepsilon)$-$\mathsf{LPN}$ *over* $\mathbb{F} := \mathbb{F}(\lambda)$ *asserts that for any polynomial* $m := m(\lambda)$ *we have:*

$$(\mathbf{r}_i, \mathbf{r}_i^{\intercal}\mathbf{s} + e_i)_{i \in [m]} \quad \approx_c \quad (\mathbf{r}_i, u_i)_{i \in [m]},$$

*where* $\mathbf{s} \leftarrow \mathbb{F}^{k}$ *and, for any* $i$*,* $\mathbf{r}_i \leftarrow \mathbb{F}^{k}$*,* $e_i \leftarrow \mathsf{Ber}_{\mathbb{F}}(\varepsilon)$*, and* $u_i \leftarrow \mathbb{F}$*.*

*We denote by* $(k, \varepsilon, m)$-$\mathsf{LPN}$ *a restricted variant of LPN where indistinguishability holds only with* $m := m(\lambda)$ *samples. In this case, we will also consider a variant where the public* $k \times m$ *matrix of* $\mathbf{r}_i$*, viewed as the generating matrix of a linear code, is picked from a general distribution.*

For any $\gamma \geq 1/2$, LPN with noise rate $\varepsilon = 1/k^{\gamma}$ implies public-key encryption [Ale03]. Obtaining a similar result with $\gamma < 1/2$ is a major open problem.

## 2.3 Trapdoored Matrices

To minimize the computational overhead of the EMVP client, we will rely on variants of the recent notion of *trapdoored matrices* from [BN25, VZ25]. A trapdoored matrix (TDM) is a pseudorandom matrix $\mathbf{R}$ which is generated together with a trapdoor that enables fast computation of the linear map $\mathbf{v} \mapsto \mathbf{R}\mathbf{v}$.

**TDM and EMVP.** As discussed in Section 1.3, a TDM can be used to upgrade a relaxed EMVP protocol $P'$, which only hides the client's vectors $\mathbf{q}_i$ but not the matrix $\mathbf{M}$, into a full-fledged EMVP protocol $P$ that also hides $\mathbf{M}$. The protocol $P$ proceeds in the following natural way: The client applies the offline phase of $P'$ to the masked matrix $\mathbf{M}' = \mathbf{M} + \mathbf{R}$. In the online phase, whenever the client wants to query $\mathbf{M}\mathbf{q}_i$, it uses $P'$ to query $\mathbf{M}'\mathbf{q}_i$, from which $\mathbf{M}\mathbf{q}_i$ can be computed by subtracting $\mathbf{R}\mathbf{q}_i$. The latter can be computed efficiently by the client given the trapdoor. In particular, the above transformation implies an EMVP protocol from any linearly homomorphic encryption scheme and a TDM.

**Centralized vs. distributed TDM.** In the centralized client setting, we can interpret a "fast" trapdoored computation of the linear map $\mathbf{v} \mapsto \mathbf{R}\mathbf{v}$ as having a near-linear size evaluation circuit $\mathsf{EC}$ that may depend on the trapdoor. However, for the distributed-client variant of our EMVP protocols considered in Section 6, it is helpful to have a near-linear size *public* $\mathsf{EC}$ that computes $\mathbf{R}\mathbf{v}$ given $\mathbf{v}$ together with a random trapdoor $\mathsf{td}$ that was used to generate $\mathbf{R}$. We refer to the latter variant as a *universal trapdoored matrix*. To capture both variants, we consider the trapdoor to be a random vector $\mathsf{td} \in \mathbb{F}^{\kappa_1} \times \{0,1\}^{\kappa_2}$ given as an input to the trapdoored matrix generator. The separation between field elements and bits enables us to support field-agnostic constructions. In our universal TDM candidates, we will have $\kappa_2 = 0$.

We formalize both the basic and universal variants of TDM below.

**Definition 2.3** (Trapdoored Matrix)**.** *A* trapdoored matrix *with trapdoor size $(\kappa_1, \kappa_2)$ is a polynomial time algorithm $\mathsf{TDM}$ with the following syntax: $\mathsf{TDM}(1^\lambda, 1^m, 1^n, \mathsf{td})$ takes $\lambda, m, n$ and a trapdoor $\mathsf{td} \in \mathbb{F}^{\kappa_1} \times \{0,1\}^{\kappa_2}$ for $\kappa_b = \kappa_b(\lambda, m, n)$, and outputs $(\mathbf{R}, \mathsf{EC})$, where $\mathbf{R} \in \mathbb{F}^{m \times n}$ and $\mathsf{EC} : \mathbb{F}^n \to \mathbb{F}^m$ is an arithmetic* evaluation circuit *over $\mathbb{F}$. The algorithm $\mathsf{TDM}$ should satisfy the following requirements.*

- **Correctness***: For every $\lambda, m, n$, $\mathsf{td} \in \mathbb{F}^{\kappa_1} \times \{0,1\}^{\kappa_2}$ and $(\mathbf{R}, \mathsf{EC}) \in \mathsf{TDM}(1^\lambda, 1^m, 1^n, \mathsf{td})$, we have $\mathsf{EC}(\mathbf{v}) = \mathbf{R}\mathbf{v}$ for every $\mathbf{v} \in \mathbb{F}^n$.*
- **Pseudorandomness***: For every polynomials $m := m(\lambda)$ and $n := n(\lambda)$, we have $\mathbf{R}_\lambda \approx_c \mathbf{U}_\lambda$, where $\mathbf{R}_\lambda$ is the first output of $\mathsf{TDM}(1^\lambda, 1^m, 1^n, \mathsf{td})$ for $\mathsf{td} \leftarrow \mathbb{F}^{\kappa_1} \times \{0,1\}^{\kappa_2}$, and $\mathbf{U}_\lambda \leftarrow \mathbb{F}^{m \times n}$.*

*A* universal trapdoored matrix *is defined similarly to the above, except that $\mathsf{EC}$ is given $\mathsf{td}$ as an additional input and pseudorandomness is strengthened to require that $(\mathbf{R}, \mathsf{EC})_\lambda \approx_c (\mathbf{U}, \mathsf{EC})_\lambda$.*

While this is not explicitly part of the definition, we would like the circuit $\mathsf{EC}$ to be of nearly optimal size, namely close to linear in $m + n$.

The recent works of Braverman and Newman [BN25] and Vaikuntanathan and Zamir [VZ25] present recursive constructions of TDM in which $\mathsf{EC}$ is computable in time $(m+n)^{1+\varepsilon}$ or even $\tilde{O}(m+n)$ under the standard LPN assumption or variants of the McEliece assumption. In Section 5 we propose direct (non-recursive) candidate TDM constructions that can achieve better asymptotic and concrete efficiency and satisfy the universal notion of TDM, at the expense of relying on less standard or even new assumptions.

# 3 Learning Subspace with Noise Assumptions

In this section we review the *learning subspace with noise* (LSN) class of assumptions put forward in [DKL09, CIMR25] and introduce a new variant that will be used to minimize the overhead of our EMVP protocols. We also extend the previous assumptions by considering variants that use structured codes (in rings or otherwise) to support faster encoding.

All flavors of LSN have the following form: We pick a secret $k$-dimensional linear code $C \subseteq \mathbb{F}^n$, and then sample polynomially many random codewords $\mathbf{c}_i$ from $C$, where each codeword is subject to some noise or perturbation. In all cases, we require that the perturbed vectors from $C$ should be indistinguishable from the distribution obtained by applying a similar perturbation to truly random vectors. The different flavors of LSN vary in the type of noise/perturbation applied to sampled codewords. In most parameter regimes, these assumptions are not known to imply public-key encryption, or even collision-resistant hashing.

**Definition 3.1** (Basic LSN [DKL09, CIMR25])**.** *The (basic)* learning subspace with noise *assumption* $(k, n, \mu)$-$\mathsf{LSN}$ *asserts that for a uniformly random secret rank-$k$ matrix* $\mathbf{C} \in \mathbb{F}^{k\times n}$ *and any polynomial number of samples* $m := m(\lambda)$*, it holds that:*

$$(\mathbf{c}_1 + \mathbf{e}_1, \ldots, \mathbf{c}_m + \mathbf{e}_m) \approx_c (\mathbf{u}_1, \ldots, \mathbf{u}_m),$$

*where* $\mathbf{c}_i = \mathbf{a}_i^\intercal \mathbf{C}$ *for* $\mathbf{a}_i \leftarrow \mathbb{F}^k$*,* $\mathbf{e}_i$ *is uniformly random in* $\mathbb{F}^n$ *with probability* $\mu$ *and* $\mathbf{e}_i = \mathbf{0} \in \mathbb{F}^n$ *otherwise, and* $\mathbf{u}_i \leftarrow \mathbb{F}^n$*.*

In the above definition and in the following, the parameters $k, n, \mu$ are all functions of the security parameter $\lambda$.

For codes of constant asymptotic rate $k/n < 1$, the above flavor of LSN requires a high level of noise ($\mu = 1-o(1)$) to be plausibly secure against polynomial-time attacks. Indeed, if $\mu < 1-(k/n)^d$, there is an $n^{O(d)}$-time LSN distinguisher [DKL09, CDV21, CIMR25]. In particular, if $1 - \mu$ is inverse-polynomial, the above basic variant of LSN is broken in quasi-polynomial time. On the other extreme, if $k = n$ the vectors are truly random, and when $k$ is very close to $n$ basic LSN is equivalent to LPN. (Concretely, when $n = k + 1$, the basic LSN assumption over $\mathbb{F}_2$ with noise rate $\mu$ is equivalent to the LPN assumption with noise rate $\mu/2$ [DKL09, CDV21, CIMR25].) If we change the noise pattern so that every sample from $\mathbf{c}_i$ is subject to noise in all but $k$ coordinates, then LSN is implied by LPN with similar parameters [CIMR25]. This observation underlies the LPN-based implementation of our protocols.

To avoid the quasi-polynomial time attack and improve the efficiency of our EMVP protocols, we consider other natural variants of LSN in which each sampled codeword $\mathbf{c}_i$ is planted in a random low-dimensional linear subspace $V_i \subseteq \mathbb{F}^n$ containing it. That is, each sample includes a random basis for such a space. We refer to this variant as *regular LSN*. Alternatively, we may plant $\mathbf{c}_i$ in an affine subspace $\mathbf{c}_i + V_i$. Over large $\mathbb{F}$, the two variants or regular LSN are equivalent up to a difference of 1 in the dimension (see Section 7.1).

Regular LSN with a subspace of dimension $r$ is at least as secure as (but seemingly more secure than) a "regular" variant of basic LSN with noise rate $\mu = 1 - 1/r$ in which each chunk of $r$ samples has exactly codeword and $r-1$ noise vectors. A similar regular variant of LPN has been extensively studied in the literature; see [AG11, BØ23, LWYY24, KPRR25] and references therein. To resist polynomial-time algebraic attacks, regular LSN would require $r$ to be super-constant. However, unlike basic LSN, here we can obtain security against sub-exponential time algebraic attacks even with constant code rate $k/n < 1$ by letting $r = k^\delta$ for any constant $\delta > 0$. See Section 7.1.

In the most efficient variants of our EMVP protocols, the space $V_i$ is a product of $s$ linear subspaces $V_{i,j}$ of $\mathbb{F}^{n/s}$, which can be viewed as splitting the coordinates of $\mathbf{c}_i$ into $s$ blocks, and hiding each block $\mathbf{c}_{i,j}$ in the affine space $\mathbf{c}_{i,j} + V_{i,j}$. We refer to this variant as "split LSN."

**Definition 3.2** (Split-LSN [CIMR25])**.** *The $(k, n, r, s)$-SLSN assumption asserts that for a uniformly random secret rank-$k$ matrix $\mathbf{C} \in \mathbb{F}^{k\times n}$ and any polynomial number of samples $m := m(\lambda)$, it holds that:*

$$((\mathbf{V}_{i,1}, \mathbf{c}_{i,1} + \mathbf{e}_{i,1}), \ldots, (\mathbf{V}_{i,s}, \mathbf{c}_{i,s} + \mathbf{e}_{i,s}))_{i\in[m]} \approx_c ((\mathbf{V}_{i,1}, \mathbf{u}_{i,1}), \ldots, (\mathbf{V}_{i,s}, \mathbf{u}_{i,s}))_{i\in[m]},$$

*where $\mathbf{c}_i = \mathbf{a}_i^\intercal \mathbf{C}$ for $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $(\mathbf{c}_{i,1}, \ldots, \mathbf{c}_{i,s})$ is a partitioning of $\mathbf{c}_i$ into blocks of length $n/s$, and for any $i, j$, $\mathbf{u}_{i,j} \leftarrow \mathbb{F}^{n/s}$ and $\mathbf{e}_{i,j} = \mathbf{d}_{i,j}^\intercal \mathbf{V}_{i,j}$ for $\mathbf{V}_{i,j} \leftarrow \mathbb{F}^{(r-1)\times(n/s)}$ and $\mathbf{d}_{i,j} \leftarrow \mathbb{F}^{r-1}$.*

This conjecture becomes weaker as $r$ increases, but even the case $r = 2$ already seems hard, as was conjectured by Chen et al. [CIMR25]:

**Conjecture 3.1** (Split-LSN conjecture [CIMR25])**.** *For every $\delta > 0$ and $0 < \rho < 1$, the $(k, n, 2, s)$-SLSN assumption holds when $k \geq \rho n$ and $s \geq (n/k) \cdot n^\delta$.*

The Split-LSN assumption with $s \gg n/k$ is a less structured variant of the assumption underlying the sk-PIR protocol proposed in [BIPW17, CHR17] and further analyzed in [BHW19, BW21, BHMW21]. See [CIMR25] for discussion. Splitting allows us to increase the dimension of the planting ambient space with essentially no increase in the computation cost of the EMVP protocol.

**1-Dimensional Split LSN.** Recall that the split-LSN experiment splits a sampled codeword into blocks, and then plants each block in a random *affine* space. It seems simpler to use a (homogeneous) *linear* space instead. Indeed, in the analysis section (Section 7) we study this simpler variant. To maximize efficiency, we want to use a 1-dimensional linear space for hiding each block. The main problem with this variant is that the security proof takes advantage of the affine structure of the original split-LSN assumption. To get around this difficulty, we will need to make the assumption slightly more complex by allowing the adversary to shift each sampled codeword before it is planted in a linear space. The shifts correspond to queries $q_i$ made by the EMVP client, and are similar to the shifts in a related assumption from [BIPW17].[4] We formalize the 1-dimensional variant below.

**Definition 3.3** (1D-Split-LSN)**.** *The* 1-dimensional split-LSN *assumption $(k, n, s)$-1D-SLSN asserts that for a uniformly random secret rank-$k$ matrix $\mathbf{C} \in \mathbb{F}^{k\times n}$ and any polynomial number of samples $m := m(\lambda)$ it holds that:*

$$(\alpha_{i,1} \cdot (\mathbf{c}_{i,1} + \Delta_{i,1}), \ldots, \alpha_{i,s} \cdot (\mathbf{c}_{i,s} + \Delta_{i,s}))_{i\in[m]} \approx_c (\mathbf{u}_i)_{i\in[m]},$$

*where $\mathbf{u}_i \in \mathbb{F}^n$, $\mathbf{c}_i = \mathbf{a}_i^\intercal \mathbf{C}$ for $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $(\mathbf{c}_{i,1}, \ldots, \mathbf{c}_{i,s})$ is a partitioning of $\mathbf{c}_i$ into blocks of length $n/s$, for any $i, j$, $\alpha_{i,j} \leftarrow \mathbb{F} \setminus \{0\}$, and where the shifts $\Delta_{i,j} \in \mathbb{F}^{n/s}$ in the $i$-th sample can be chosen adversarially based on previous samples (this choice does not affect the random experiment).*

Note that 1D-SLSN is not meaningful over $\mathbb{F}_2$, since in this case all coefficients $\alpha_{i,j}$ need to be 1. For fields $\mathbb{F}_q$ with $q \geq 3$, we conjecture it to have similar security to the original SLSN assumption with a 1-dimensional affine space, namely with $r = 2$. In particular:

[4] A different variant of this assumption from [CHR17] avoids these shifts by relying on the fact that the query domain is polynomial, which is not the case here. One can avoid these shifts in our context by relying on the earlier Split-LSN assumption (Definition 3.2) with $r = 2$, paying roughly 2x more in communication and server computation.

**Conjecture 3.2** (1D-SLSN conjecture)**.** *For every $\delta > 0$ and $0 < \rho < 1$ and $\mathbb{F}$ such that $|\mathbb{F}| \geq 3$, the $(k, n, s)$-*1D-SLSN *assumption over $\mathbb{F}$ holds when $k \geq \rho n$ and $s \geq (n/k) \cdot n^{\delta}$.*

In Section 7 we will analyze a simplified version of 1D-SLSN where all offsets $\Delta_{i,j}$ are 0. The extra shifting power of the adversary is not helpful for any attack we are aware of. The hardness of the simplified variant of the conjecture implies indistinguishability between the case where the client's queries $\mathbf{q}_i$ are random and the case where $\mathbf{q}_1 = \mathbf{q}_2 = \ldots = \mathbf{q}_m = \mathbf{0}$. While this intuitively seems like the easiest distinguishing case, we cannot back this by a security reduction.

Finally, motivated by the concrete analysis in Section 7, we will also consider a more conservative variant of 1D-SLSN where each sample uses an independently random partition into $s$ blocks.

### 3.1 Ring-LSN and Other Structured Codes

All of the assumptions discussed up to this point involve a *random* code $C$, making the computational cost of both the matrix encoding and query encoding scale quadratically with the row length parameter $\ell$. To make these costs scale quasilinearly or even linearly with $\ell$, we can choose $C$ from suitable families of *structured* codes. Indeed, the secret-key PIR constructions from [BIPW17, CHR17] implicitly rely on such structured variants of LSN.

One common choice of a structured code corresponds to a natural ring variant of LSN, where both $C$ and its dual $D$ admit quasilinear-time encoding using polynomial multiplication. This takes a particularly simple form when $n = 2k$. In this case, the encoding matrix of $C$ can be written as $\mathbf{C} = (I \,|\, \mathbf{C}')$ where $\mathbf{C}' \in \mathbb{F}^{k \times k}$ is a random circulant matrix, and the dual code $D$ can be generated by $\mathbf{D} = (-\mathbf{C}'^{\mathsf{T}} \,|\, I)$, hence can also support quasilinear-time encoding. Concretely, we work over the polynomial ring $R = \mathbb{F}[X]/P(X)$, where $P$ is a fixed polynomial of degree $k$, typically an irreducible polynomial or even just $P = X^k - 1$. We view both the code $C$ and a message $v$ as elements of $R$, represented by coefficient vectors $\mathbf{C}$ and $\mathbf{v}$ in $\mathbb{F}^k$. The encoding of $v$ under $C$ is $(v, C \cdot v) \in R^2 \equiv \mathbb{F}^n$ and the dual encoding under $D$ is $(-C^{\mathsf{T}} \cdot v, v) \in \mathbb{F}^n$. When $P(X) = X^k - 1$, the polynomial $C^{\mathsf{T}}$ is obtained from $C$ by reversing its coefficient vector. In general, if the mapping $\mathbf{v} \mapsto C \cdot \mathbf{v}$ can be implemented with $s$ additions and scalar multiplications, then the transpose mapping $\mathbf{v} \mapsto C^{\mathsf{T}} \cdot \mathbf{v}$ can be implemented with $O(s)$ such operations [Bor56].

Ring variants of this kind serve as common substitutes for random linear codes in the context of cryptography from noisy linear algebra [LPR10, HKL+12, MBD+18, BCG+20]. This makes ring variants of our LSN assumptions plausible. Some care must be taken when using codes over algebraic rings in the context of 1D-SLSN to avoid bad interactions between the algebraic ring structure and the 1D-SLSN block structure. For example a particularly bad choice would be to use $P = X^k - 1$ and blocks of size $n/s$ that divides $k$. When relying on 1D-SLSN it may therefore be better to use an irreducible $P$. (Alternatively, one can take $P(X) = X^k - 1$ and apply a public random permutation to the codeword (in both $C$ and $D$) to avoid this bad interaction.)

Finally, we note that one could potentially use other families of dual linear codes that admit fast encoding, since our EMVP protocols do not need the ring structure per se. When $n = 2k$, this can be done by letting $\mathbf{C} = (I \,|\, \mathbf{C}')$ and $\mathbf{D} = (-\mathbf{C}'^{\mathsf{T}} \,|\, I)$ as above, where $\mathbf{C}' \in \mathbb{F}^{k \times k}$ is a distribution over $k \times k$ matrices such that each instance of $\mathbf{C}'$ is fast, in the sense that $\mathbf{v} \mapsto \mathbf{C}' \cdot \mathbf{v}$ can be computed by a circuit with $O(k)$ additions and scalar multiplications. Good heuristic choices of such $\mathbf{C}'$ can be obtained from known families of linear-time encodable codes, e.g., ones from [BDMP98, DJM98, GM08, DI14, BCG+22]. Alternatively, one can use a recent construction of fast dual codes from [BR25], based on "repeat multiple accumulate" codes. In this construction both the code and its dual provably meet the Gilbert-Varshamov bound over $\mathbb{F}_2$, and can be conjectured to meet this bound over general $\mathbb{F}$.

**LSN modulo a composite.** Our EMVP protocols are not inherently limited to work over finite fields. For instance, it may be desirable to design EMVP protocols that work natively over rings of the form $R = \mathbb{Z}_{2^k}$, where elements may not have an inverse. Here we note that most of our protocols do not require inversion, since one just needs to generate a random systematic code and its dual. For instance, when $n = 2k$ as before we can let $\mathbf{C} = [\, I \,|\, \mathbf{C}_1 \,]$ and $\mathbf{D} = [\, -\mathbf{C}_1^{\mathsf{T}} \,|\, I \,]$ for a random $\mathbf{C}_1$ over $R$, where security would follow from the suitable variant of LSN over $R$. For the most efficient variant based on 1D-SLSN, we must of course choose random *invertible* scalars $\alpha_i \in R$ (and in this case we do need to invert them). In the context of LPN-based cryptography over $\mathbb{Z}_{2^k}$, such constructions have been analyzed and have so far resisted all known attacks [LWYY24].

# 4 EMVP Protocols

In this section we describe two EMVP protocols. In Section 4.1, we present our main protocol, which maximizes efficiency under the 1D-SLSN assumption. In Section 4.2 (and Appendix A) we describe our LPN-based protocol, which yields near-optimal *asymptotic* (but not concrete) efficiency under a standard assumption.

## 4.1 EMVP from 1D-SLSN

Recall that in the 1D-SLSN assumption, we hide each sampled codeword from a secret code $C$ by splitting it into $s$ blocks, and multiplying each block by a random invertible scalar $\alpha_i$. An EMVP protocol based on the conjectured hardness of distinguishing such (shifted) samples from random is formally described in Fig. 1.

In the setup phase of the protocol, the rows of the matrix $\mathbf{M}$ are encoded into $\tilde{\mathbf{M}}$ using a random systematic encoding matrix $\mathbf{D}$, and then $\tilde{\mathbf{M}}$ is masked with a pseudorandom matrix $\mathbf{R}$. The resulting matrix $\widehat{\mathbf{M}} = \tilde{\mathbf{M}} + \mathbf{R}$ is uploaded to the server. To make a query, the client samples a random $\mathbf{c} \leftarrow C$, where $C$ is the dual code of the code generated by $\mathbf{D}$. Setting $\tilde{\mathbf{q}} = \mathbf{c} + (\mathbf{q} \,|\, 0^k)$, the clients wants to obtain the product $\tilde{\mathbf{M}}\tilde{\mathbf{q}}$ (which is equal to $\mathbf{Mq}$ since $\mathbf{D} = (\mathbf{I}|\mathbf{D}')$ is systematic).

Of course, the client needs to hide $\tilde{\mathbf{q}}$ from the server, which it does (under 1D-SLSN) by breaking it into $s$ blocks and multiplying the $i$'th block by the scalar $\alpha_i$ as above. Concatenating all these blocks, the client sends to the server the encrypted vector $\widehat{\mathbf{q}}$. The server multiplies each block of $\widehat{\mathbf{M}}$ with the corresponding block of $\widehat{\mathbf{q}}$, sending the result (which is an $m$-by-$s$ matrix) back to the client. The client can undo the effect of the $\alpha_i$ by taking a linear combination of the $s$ columns vectors with coefficients $\alpha_i^{-1}$. Note, however, that since $\tilde{\mathbf{M}}$ is masked by $\mathbf{R}$, the client must also cancel the effect of the server's evaluation over the mask. This can be done efficiently if $\mathbf{R}$ is a trapdoored matrix, which allows the client to perform fast multiplication with $\tilde{\mathbf{q}}$.

**Lemma 4.1** (EMVP from 1D-SLSN: Correctness)**.** *The EMVP protocol from Fig. 1 satisfies the correctness requirement.*

*Proof.* In decoding, the client computes the value $\mathbf{a} = \mathbf{M}'\mathbf{p}' - \mathbf{r}'$. By construction, $\mathbf{r}' = \mathsf{EC}(\tilde{\mathbf{q}})$, which is equal to $\mathbf{R}\tilde{\mathbf{q}}$ by the correctness of the underlying trapdoored matrix. Additionally, we can rewrite

$$\mathbf{M}'\mathbf{p}' = [\widehat{\mathbf{M}}_1\widehat{\mathbf{q}}_1 \mid \ldots \mid \widehat{\mathbf{M}}_s\widehat{\mathbf{q}}_s]\mathbf{p}' = \sum_{i=1}^{s} \alpha_i^{-1}\widehat{\mathbf{M}}_i\widehat{\mathbf{q}}_i = \sum_{i=1}^{s} \widehat{\mathbf{M}}_i\tilde{\mathbf{q}}_i = \widehat{\mathbf{M}}\tilde{\mathbf{q}}.$$

Correctness then holds since

$$\widehat{\mathbf{M}}\tilde{\mathbf{q}} = \mathbf{MD}\tilde{\mathbf{q}} + \mathbf{R}\tilde{\mathbf{q}} = \mathbf{M}\big(\mathbf{D}(\mathbf{q} \,|\, 0^k) + \mathbf{Dc}\big) + \mathbf{r}' = \mathbf{Mq} + \mathbf{r}',$$

**Parameters:**

- 1D-SLSN parameters $k(\lambda), n(\lambda), s(\lambda)$ such that $s|n$ and $n - k \geq \lambda^{\Omega(1)}$.
- Field oracle $\mathbb{F}$, where $|\mathbb{F}| > 2$.
- Trapdoored matrix TDM with trapdoor size $\kappa(\lambda, m, n)$.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

**The Protocol:**

- $\mathsf{G}(1^\lambda, 1^m, 1^\ell)$:
  1. Let $\lambda_0 = \min\{\lambda' : \lambda' \geq \lambda,\ \ell \leq n(\lambda') - k(\lambda')\}$ and let $\ell_0 = n(\lambda_0) - k(\lambda_0)$.
     // In the following, we assume w.l.o.g. that $\ell = \ell_0$. Otherwise, the rows in the input matrix are padded with $\ell_0 - \ell$ zeros (it always holds that $\ell_0 \geq \ell$).
  2. Let $n = n(\lambda_0), k = k(\lambda_0), s = s(\lambda_0), \kappa = \kappa(\lambda, m, n)$
  3. Return $\mathsf{sk} = (\mathbf{k}, m, \ell, n, k, s, \kappa)$ where $\mathbf{k} \leftarrow \{0,1\}^\lambda$ is a PRF key
- $\mathsf{E}(\mathsf{sk}, \mathbf{M})$: Use $\mathsf{PRF}(\mathbf{k}, \cdot)$ to determine the randomness in the following.
  1. Let $\mathbf{C} \leftarrow \mathbb{F}^{k \times n}$ define a secret 1D-SLSN code $C$
  2. Let $D = C^\perp$ and $\mathbf{D} \in \mathbb{F}^{\ell \times n}$ be a *systematic* generator matrix of $D$, where $\mathbf{D} = [I_\ell \,|\, \mathbf{D}']$
  3. Let $\mathsf{td} \leftarrow \mathbb{F}^\kappa$ and $(\mathbf{R}, \mathsf{EC}) \leftarrow \mathsf{TDM}(1^\lambda, 1^m, 1^n, \mathsf{td})$ // $\mathbf{R}$ is a trapdoored matrix
  4. Return $\widehat{\mathbf{M}} = \mathbf{MD} + \mathbf{R} \in \mathbb{F}^{m \times n}$
- $\mathsf{Q}(\mathsf{sk}, \mathbf{q})$:
  1. Let $\mathbf{r} \leftarrow \mathbb{F}^k$ and $\mathbf{c} \leftarrow \mathbf{r}^\mathsf{T}\mathbf{C} \in \mathbb{F}^n$ // $\mathbf{c}$ is a random codeword
  2. Let $\tilde{\mathbf{q}} = (\mathbf{q} \,|\, 0^k) + \mathbf{c}$ and $\mathbf{r}' = \mathsf{EC}(\tilde{\mathbf{q}}) \in \mathbb{F}^m$ // use the trapdoor to compute $\mathbf{r}' = \mathbf{R}\tilde{\mathbf{q}}$
  3. Let $\alpha_1, \ldots, \alpha_s \leftarrow \mathbb{F} \setminus \{0\}$ and let $\mathbf{p}' = (\alpha_1^{-1}, \ldots, \alpha_s^{-1}) \in \mathbb{F}^s$
  4. Let $\widehat{\mathbf{q}} = (\alpha_1 \cdot \tilde{\mathbf{q}}_1 \,|\, \ldots \,|\, \alpha_s \cdot \tilde{\mathbf{q}}_s) \in \mathbb{F}^n$, where $(\tilde{\mathbf{q}}_1 \,|\, \ldots \,|\, \tilde{\mathbf{q}}_s)$ is a partition of $\tilde{\mathbf{q}}$ into blocks of length $n/s$
  5. Let $\mathbf{q}' = (\mathbf{p}', \mathbf{r}')$
  6. Return $(\widehat{\mathbf{q}}, \mathbf{q}')$
- $\mathsf{A}(\widehat{\mathbf{M}}, \widehat{\mathbf{q}})$:
  1. Parse $\widehat{\mathbf{M}}$ as $[\widehat{\mathbf{M}}_1 \,|\, \ldots \,|\, \widehat{\mathbf{M}}_s]$ and $\widehat{\mathbf{q}}$ as $(\widehat{\mathbf{q}}_1 \,|\, \ldots \,|\, \widehat{\mathbf{q}}_s)$.
  2. Return $\mathbf{M}' = [\widehat{\mathbf{M}}_1\widehat{\mathbf{q}}_1 \,|\, \ldots \,|\, \widehat{\mathbf{M}}_s\widehat{\mathbf{q}}_s] \in \mathbb{F}^{m \times s}$
- $\mathsf{D}(\mathsf{sk}, \mathbf{M}', \mathbf{q}')$: Return $\mathbf{M}'\mathbf{p}' - \mathbf{r}' \in \mathbb{F}^m$

Figure 1: Main EMVP Protocol from 1D-SLSN.

where the last equality follows by the fact that $\mathbf{D}$ is of the form $[I_\ell \,|\, \mathbf{D}']$ and $\mathbf{c}$ is orthogonal to the code generated by $\mathbf{D}$. □

**Lemma 4.2** (EMVP from 1D-SLSN: Security)**.** *The EMVP protocol from Fig. 1 is secure under* $(k, n, s)$-1D-SLSN *(Definition 3.3).*

*Proof.* Consider a hybrid protocol where the outputs of the PRF are replaced by uniformly random outputs and, consequently, $C$ is a uniformly random code and $\mathbf{R}$ that is generated by TDM using a uniformly random trapdoor. Furthermore, replace $\mathbf{R}$ by a uniformly random matrix. Under the security of the PRF and TDM, this is as secure as the original protocol. In the hybrid protocol, since the database encoding is masked by a uniformly random $\mathbf{R}$ which is independent in the queries generated by the client, the adversary's view can be simulated only given the queries, which consist of polynomially many samples of the form $\widehat{q} = (\alpha_1 \cdot \tilde{\mathbf{q}}_1 \,|\, \ldots \,|\, \alpha_s \cdot \tilde{\mathbf{q}}_s)$, where $\tilde{\mathbf{q}} = (\tilde{\mathbf{q}}_1 \,|\, \ldots \,|\, \tilde{\mathbf{q}}_s)$ is a partition of $(\mathbf{q} \,|\, 0^k) + \mathbf{c}$ where $\mathbf{c}$ is a uniformly random codeword in $C$ and $\alpha_1, \ldots, \alpha_s \leftarrow \mathbb{F} \setminus \{0\}$. Such queries correspond precisely to $(k, n, s)$-1D-SLSN samples with hidden code $C$ and shifts $(\Delta_1 \,|\, \ldots \,|\, \Delta_s) = (\mathbf{q} \,|\, 0^k)$ (see Definition 3.3). These queries are thus pseudorandom under $(k, n, s)$-1D-SLSN and the security of the protocol follows. □

**Optimizing download rate.** Recall that 1D-SLSN considers a pseudorandom experiment where each sampled codeword is split into $s$ blocks, and each block is hidden in a 1-dimensional linear space. The parameter $s$ has little impact on the client and server computation, but it does impact the download rate, blowing up the size of the server's answer by a factor of $s$. To achieve download rate 1 (namely, server-to-client message that approaches the output length), we can compose the EMVP protocol with any rate-1 additively homomorphic encryption (AHE) in the following way. In parallel to sending the query $\widehat{q}$ to the server, the client also sends the AHE-encrypted $\mathbf{q}' = (\mathbf{p}', \mathbf{r}')$. Since $\mathbf{q}'$ is short, this does not add much to the upload cost. Now instead of sending $\mathbf{M}'$ to the client, the server uses the AHE to homomorphically evaluate the output $\mathbf{M}'\mathbf{p}' - \mathbf{r}'$, which the client can decrypt. Note that this does not require additional rounds of interaction. Rate-1 AHE can be implemented from a variety of cryptographic assumptions [DGI+19, GH19, BDGM19], though AHE with lower constant rate may be better in practice; see, e.g., [BMW24] and references therein. With this extension, the protocol is no longer field agnostic. However, the non-agnostic part is asymptotically dominated by the other costs.

**Alternative LSN-style assumptions.** The protocol from Fig. 1 can be easily adapted to use the $(k, n, r, s)$-SLSN assumption (Definition 3.2) instead of 1D-SLSN. This may allow for a more flexible choice of parameters (see Section 7), increasing both the upload cost and the download cost by an extra factor of $r$. The increase in the download cost can be mitigated via composition with HE as discussed above.

**Application to PIR.** Private information retrieval (PIR) with secret-key preprocessing can be reduced to EMVP by viewing the database as a matrix, and multiplying it by an indicator vector corresponding to the client's query. Applying this directly for a database of $N$ bits, viewed as an $N^{1/2} \times N^{1/2}$ matrix over $\mathbb{F}_2$, leads to PIR with $O(N^{1/2})$ communication, which can be further improved via a tensoring-based approach [CIMR25]. Using $(k, n, 2, s)$-SLSN to implement EMVP over $\mathbb{F}_2$, the construction from [CIMR25] allows the server to be implemented by a Boolean circuit of size $(4 + o(1))N$ with $(1 + o(1))N$ bits of server storage. In Appendix C we show how to use EMVP over $\mathbb{F}_4$ based on 1D-SLSN to improve the server circuit complexity to $(3.5 + o(1))N$, or alternatively to $(3 + o(1))N$ with 50% extra server storage.

**Post-processing.** Our EMVP protocol generates output shares of the matrix-vector product $\mathbf{Mq}$, where the server holds a matrix $\mathbf{M}'$ and the client holds vectors $\mathbf{p}', \mathbf{r}'$ such that $\mathbf{Mq} = \mathbf{M}'\mathbf{p}' - \mathbf{r}'$. These output shares are typically much smaller than the input: the matrix $\mathbf{M}'$ has the same number of rows as $\mathbf{M}$ but only $s$ columns compared to the $\ell$ columns of $\mathbf{M}$. (We will typically have $s \ll \ell$; see Table 1 for some concrete choices of parameters.) While simply communicating $\mathbf{M}'$ to the client is good enough for some use cases, this may be undesirable for two possible reasons: either because it involves too much communication, or because it reveals to the client too much information about $\mathbf{M}$. An example of the latter is the functionality from [BKSW24], where the client should only learn whether there is an entry of $\mathbf{Mq}$ which is bigger than some public threshold.

A conceptually simple and general approach for tackling both of the above issues simultaneously is to implement the post-processing via the use of FHE. When $\ell \gg s$, the resulting computational overhead can be much smaller than performing the entire computation under the hood of FHE. Our universal TDM construction can be used to compute the output mask $\mathbf{r}'$ from a short encrypted query and an encrypted TDM trapdoor.

As a leaner alternative to FHE, one can first securely convert the nonlinear output shares $(\mathbf{M}', (\mathbf{p}', \mathbf{r}'))$ to additive output shares, and then apply some combination of traditional secure computation techniques and linear sketching. The details of such an approach depend on the specific use case and are beyond the scope of this work. To securely convert the output shares to additive shares, one can either use AHE (with communication cost comparable to the matrix-vector product size) or use a lighter-weight approach based on PCGs for VOLE [BCGI18] (with communication cost comparable to the size of the output shares of the matrix-vector product).

Finally, in settings where the EMVP client is distributed using an honest-majority MPC protocol (see Section 6), the above post-processing approaches can become much more efficient. In particular, the above use of AHE can be replaced by lightweight techniques for information-theoretic MPC. Further details and optimizations are left for future work.

### 4.2 EMVP from LPN

A more conservative EMVP protocol based on the standard LPN assumption over $\mathbb{F}$ can be obtained by adapting an LPN-based secret-key PIR protocol from [CIMR25] to the EMVP functionality.

**Theorem 4.1** (EMVP from LPN)**.** *Suppose* $(k, k^{-\gamma})$-$\mathsf{LPN}$ *holds for some* $0 < \gamma < 1$*. Then, for any* $0 < \gamma' < \gamma$*, there is a field-agnostic EMVP protocol for* $m \times \ell$ *matrices with server storage and computation* $O(\ell^{1/\gamma'} \cdot m)$*, query length* $O(\ell^{1/\gamma'})$ *and answer length* $O(m)$*.*

Letting $\gamma = 1 - \delta$ and $\gamma' = 1 - 2\delta$ for $\delta$ approaching 0, we get an EMVP protocol in which all complexity measures are close to optimal under a standard variant of LPN. In fact, assuming LPN in the (still standard) low-noise regime $\varepsilon = \mathrm{polylog}(k)/k$, all complexity measures are optimal up to $\mathrm{polylog}(k)$ factors.

We hereby give a high-level overview of the protocol. A detailed description of the protocol, formal analysis of its correctness and security and concrete performance data are given in Appendix A.

The protocol follows the same blueprint underlying the $\mathsf{1D\text{-}SLSN}$-based protocol from Fig. 1, except for the different type of noise and the way it is handled. Specifically, the client adds noise to its query by applying Bernoulli noise with probability $\varepsilon$ in each of the first $\ell = n - k$ coordinates of the query codeword, i.e. in all the coordinates except for the systematic part. In such a case, the distribution of the queries follows the distribution of LPN samples with dimension $k$ and noise $\varepsilon$, where the hidden code from which the query codewords come defines a matrix of $k$ LPN secrets. Note that the client now is able to decode the result of the evaluation only if the query it sends is noiseless. This occurs with probability only $(1 - \varepsilon)^{n-k} < 1$.

By tuning the noise parameter, we can get a protocol that is secure under LPN but has a non-negligible failure probability. We apply an error-correcting code to each column of $\mathbf{M}$ to make the failure probability negligible. Note that erasure decoding suffices since the client is able to anticipate the noisy queries. In particular, we can make the failure probability negligible at a low amortized cost, which vanishes when $m$ grows.

Unfortunately, in spite of the near-optimal asymptotic complexity, obtaining significant savings in communication has a high concrete price in computation and storage. See Table 2 and the preceding discussion in Appendix A.

# 5 New Trapdoored Matrix Constructions

In this section we present new TDM constructions (Definition 2.3) that are tailored to our goals of concrete efficiency and lightweight protocols for distributing EMVP clients.

All of the following constructions generate the trapdoored matrix $\mathbf{R}$ by multiplying secret and public matrices $\mathbf{M}_i$, where (1) all matrices are "fast" in the sense that $\mathbf{v} \mapsto \mathbf{M}_i\mathbf{v}$ can be computed by a near-linear size arithmetic circuit, and (2) the matrices are "incompatible" in the sense that they do not share the same structure. As in [BN25, VZ25], we consider here without loss of generality the case of generating a square $k \times k$ trapdoor matrix $\mathbf{R}$. The general case reduces to this case by covering a rectangular matrix with square matrices, padding with 0's if needed. However, as discussed below, some variants of our TDM constructions have a lower overhead for a rectangular TDM $\mathbf{R}$ that has many more rows than columns or vice versa.

## 5.1 TDM from Dual LPN

We start with a simple generic construction of TDM from the dual version of the LPN assumption, applied to suitable families of structured codes. In this construction, the pseudorandom matrix $\mathbf{R}$ is defined by $\mathbf{R} = \mathbf{HE}$, where $\mathbf{H} \in \mathbb{F}^{k\times n}$ is a public parity-check matrix for which dual LPN holds, namely $(\mathbf{H}, \mathbf{He}) \approx_c (\mathbf{H}, \mathbf{u})$ for a sparse $\mathbf{e}$ and random $\mathbf{u}$, and $\mathbf{E} \in \mathbb{F}^{n\times k}$ is a secret *sparse* matrix defined by $\mathsf{td}$. (Here the $\kappa_1$ field elements of $\mathsf{td}$ determine the values of the nonzero entries of $\mathbf{E}$ and the $\kappa_2$ bits determine their locations.)

As we show below, $\mathbf{R}$ is pseudorandom based on the dual-LPN assumption defined by $\mathbf{H}$. To ensure that $\mathbf{R}$ is also *fast* given the trapdoor $\mathbf{E}$, we pick $\mathbf{H}$ from a family of fast matrices, namely where the mapping $\mathbf{e} \mapsto \mathbf{He}$ can be computed by a (near-)linear size circuit. For example, we can let $\mathbf{H}$ be a random quasi-cyclic matrix, corresponding to the Ring-LPN assumption over $\mathbb{F}$ [HKL$^+$12, BCG$^+$20], or pick it from any family of linear-time encodable codes that are conjectured to be (dual-)LPN-friendly [DI14, BCG$^+$22, BCF$^+$24].

To formalize the above, we start by recalling the definition of dual-LPN.

**Definition 5.1** (Dual-LPN)**.** *Let $\mathcal{H}$ denote a PPT sampling algorithm where, for any $k, n \in \mathbb{N}$, $\mathcal{H}(1^k, 1^n)$ samples a matrix $\mathbf{H} \in \mathbb{F}^{k\times n}$. The* (decisional) dual-LPN *assumption $(k, n, \mathcal{H}, \varepsilon)$-*dual-LPN *over $\mathbb{F} := \mathbb{F}(\lambda)$, for polynomials $k := k(\lambda)$, $n := n(\lambda)$ and $\varepsilon := \varepsilon(\lambda) \in (0, 1)$, asserts that:*

$$(\mathbf{H}, \mathbf{He}) \quad \approx_c \quad (\mathbf{H}, \mathbf{u}),$$

*where $\mathbf{H} \leftarrow \mathcal{H}(1^k, 1^n)$, $\mathbf{e} \leftarrow \mathsf{Ber}^n_{\mathbb{F}}(\varepsilon)$ and $\mathbf{u} \leftarrow \mathbb{F}^k$.*

The $(k, n, \mathcal{H}, \varepsilon)$-dual-LPN assumption is equivalent to LPN (Definition 2.2) with dimension $k' = n - k$, noise rate $\varepsilon$ and $n$ samples, where the LPN generating matrix is (a random basis of) the code whose parity-check matrix is $\mathbf{H} \leftarrow \mathcal{H}$ (see, e.g., [BCG$^+$19b, Section 3.3.1]).

**Parameters:**

- Field oracle $\mathbb{F}$
- dual-LPN parameters $k(\lambda), n(\lambda), \varepsilon(\lambda)$ and a sampling algorithm $\mathcal{H}(1^k, 1^n)$ with output in $\mathbb{F}^{k \times n}$

$\mathsf{TDM}(1^\lambda, \mathsf{td})$ // **Generate $k \times k$ trapdoored matrix for $k = k(\lambda)$**

1. Use $\mathsf{td}$ to sample $\mathbf{H} \leftarrow \mathcal{H}(1^k, 1^n)$ and $\mathbf{E} \leftarrow \mathsf{Ber}_{\mathbb{F}}^{n \times k}(\varepsilon)$
2. Let $\mathsf{EC} : \mathbb{F}^k \to \mathbb{F}^k$ be an arithmetic circuit that, on input $\mathbf{v} \in \mathbb{F}^k$, computes $\mathbf{v}' = \mathbf{E}\mathbf{v}$ and outputs $\mathbf{H}\mathbf{v}'$
3. Output $(\mathbf{R} = \mathbf{H}\mathbf{E}, \mathsf{EC})$

Figure 2: TDM from dual-LPN.

A standard hybrid argument (see Appendix B) implies that if $(\mathbf{H}, \mathbf{He})$ is pseudorandom for a Bernoulli noise vector $\mathbf{e}$, then so is $(\mathbf{H}, \mathbf{HE})$ for a Bernoulli matrix $\mathbf{E}$.

**Lemma 5.1** (Dual-LPN implies matrix dual-LPN)**.** *The $(k, n, \mathcal{H}, \varepsilon)$-dual-LPN assumption over $\mathbb{F}$ implies the following for any polynomial $m := m(\lambda)$:*

$$(\mathbf{H}, \mathbf{HE}) \quad \approx_c \quad (\mathbf{H}, \mathbf{U}),$$

*where $\mathbf{H} \leftarrow \mathcal{H}(1^k, 1^n)$, $\mathbf{E} \leftarrow \mathsf{Ber}_{\mathbb{F}}^{n \times m}(\varepsilon)$ and $\mathbf{U} \leftarrow \mathbb{F}^{k \times m}$.*

Lemma 5.1 directly induces a trapdoored matrix construction, presented in Fig. 2, which given a security parameter $\lambda$ generates a $k \times k$ trapdoored matrix.

**Proposition 5.1** (TDM from dual-LPN)**.** *The TDM from Fig. 2 is secure under $(k, n, \mathcal{H}, \varepsilon)$-dual-LPN. Moreover, the expected size of the evaluation circuit $\mathsf{EC}$ is $\varepsilon kn + |\mathsf{EC}_{\mathbf{H}}|$, where $|\mathsf{EC}_{\mathbf{H}}|$ is the size of an arithmetic circuit $\mathsf{EC}_{\mathbf{H}} : \mathbb{F}^n \to \mathbb{F}^k$ evaluating the linear map $\mathbf{e} \mapsto \mathbf{He}$.*

To make the dual-LPN based TDM construction efficient, we would like to minimize the size of $\mathsf{EC}_{\mathbf{H}}$. We discuss two different instantiations based on two different types of fast codes: quasi-cyclic codes and specific families of linear-time encodable codes.

### 5.1.1 TDM from Ring-LPN

Ring-LPN [HKL+12, MBD+18, BCG+20], an LPN analogue of ring-LWE [LPR10], broadly refers to a class of structured LPN assumptions where both the encoding and its dual have a ring-based structure that gives rise to quasilinear-time computation using FFT. For example, one can take $\mathbf{H}$ to be a random Toeplitz matrix, or alternatively a random quasi-cyclic matrix of the form $\mathbf{H} = [\, I \,|\, \mathbf{C} \,]$, where $I$ is the $k \times k$ identity matrix and $\mathbf{C}$ is a random circulant matrix whose rows are all cyclic shifts of a random vector in $\mathbb{F}^k$. Alternatively (and more conservatively), letting $\widehat{\mathbb{F}}$ be a degree-$k$ field extension of $\mathbb{F}$, viewed as a $k$-dimensional vector space over $\mathbb{F}$, $\mathbf{C}$ can be chosen to implement multiplication by a random scalar in $\widehat{\mathbb{F}}$. In all of these cases, dual-LPN is conjectured to hold and the map $\mathbf{e} \mapsto \mathbf{He}$ can be computed by a circuit of size $O(k \log k)$.

Compared to the standard LPN assumption, (dual) ring-LPN is more structured and hence less conservative. However, the above instantiations of ring-LPN are comparable to standard LPN in terms of their concrete security against known attacks. Assuming known attacks are optimal up to a poly$(k)$ factor, we get a TDM for $k \times k$ matrices with security against poly$(k)$-time distinguishers in which $\mathsf{EC}$ can be a circuit of size $k \cdot t$ for any $t = \omega(\log k)$. This can be compared to the recursive LPN-based constructions from [BN25, VZ25], where $\mathsf{EC}$ has size (at least) $k \cdot t^2$ and concrete savings over the naive solution are only achieved for larger $k$.

### 5.1.2 TDM from Linear-Size Dual-LPN

In the above ring-LPN based TDM, the mapping $\mathbf{e} \mapsto \mathbf{He}$ can be computed by circuits of quasi-linear size. Since the ring structure is not essential for dual-LPN to hold, one may try to instantiate Proposition 5.1 with *linear-size* computable maps $\mathbf{e} \mapsto \mathbf{He}$ for which dual-LPN holds. Candidates for such $\mathbf{H}$ were proposed in the line of work on pseudorandom correlation generators [BCGI18, BCG$^+$19b] and can be based on different families of linear-time encodable codes from [BDMP98, DJM98, GM08, DI14, BCG$^+$22]. Note that even if the code is efficiently decodable, its dual may support LPN. These candidates give rise to TDM constructions in which the second (non-sparse) linear map $\mathbf{H}$ is computable by a linear-size circuit. However, the total asymptotic size of $\mathsf{EC}$ is dominated by the sparse linear map $\mathbf{E}$, and hence will be the same as in the previous ring-LPN based constructions.

The general template for such constructions is as follows. Suppose that, for $\mathbf{C} \in \mathbb{F}^{k\times n}$, $\mathbf{x} \mapsto \mathbf{x}^\mathsf{T}\mathbf{C}$ is an encoding function of a linear code $C$ such that (primal) LPN holds in $C^\perp$. Then the transpose map $\mathbf{e} \mapsto \mathbf{Ce}$ satisfies dual-LPN. Moreover, by the transposition principle [Bor56, Ber], if the encoding $\mathbf{x} \mapsto \mathbf{x}^\mathsf{T}\mathbf{C}$ can be implemented with $s$ additions and scalar multiplications, then the transpose map $\mathbf{e} \mapsto \mathbf{Ce}$ requires only $O(s)$ such operations. Hence, linear codes with fast encoders whose *duals* are not known to admit efficient decoding algorithms can serve as a basis for fast TDM.

**Linear-size evaluation for rectangular TDM.** When $\mathbf{R} = \mathbf{HE}$ is a square $k \times k$ matrix, $\mathbf{E}$ should have $\omega(k \log k)$ nonzero entries. Indeed, the pseudorandomness of $\mathbf{R}$ requires each of the $k$ columns of $\mathbf{E}$ to have $t = \omega(\log \lambda)$ nonzero entries. This makes $\mathsf{EC}$ super-linear. However, if $\mathbf{R}$ is a rectangular $Tk \times k$ matrix, where $T \geq t$, we can make $\mathsf{EC}$ linear in the output size by using a sparse $\mathbf{E} \in \mathbb{F}^{2Tk\times k}$ with $t$ nonzero entries per column and fast, dual-LPN friendly $\mathbf{H} \in \mathbb{F}^{Tk\times 2Tk}$. Using the transposition principle, the same holds for a rectangular TDM of dimensions $k \times Tk$.

## 5.2 Faster and Universal TDM?

Recall that in the above dual-LPN based TDM constructions, the trapdoored matrix is of the form $\mathbf{R} = \mathbf{HE}$, where $\mathbf{E}$ is a sparse LPN noise matrix. This has two disadvantages. First, the asymptotic and concrete level of sparsity of $\mathbf{E}$ is limited (typically around 100 nonzero entries per column or more), leading to a corresponding slowdown. (As discussed above, this slowdown can be mitigated when $\mathbf{R}$ is rectangular.) Second, this TDM is not *universal* because the composed map $\mathbf{v} \mapsto \mathbf{HEv}$ is only fast when the sparse mapping $\mathbf{E}$ is wired into $\mathsf{EC}$, ruling out a public universal $\mathsf{EC}$.

Instead, we propose a heuristic approach for a universal TDM that can also support $\mathsf{EC}$ with an asymptotically optimal size. The general blueprint is to let the trapdoored matrix be a product of a constant number of secret "fast" matrices, where to support mixing we interleave the products with public random permutations. Concretely, let $c \geq 1$ be an expansion factor, and let

$$\mathbf{R} = \mathbf{S}_L \cdot \Pi_L \cdot \mathbf{S} \cdot \Pi_R \cdot \mathbf{S}_R, \tag{1}$$

where

- $\mathbf{S}_L \in \mathbb{F}^{ck\times k}, \mathbf{S} \in \mathbb{F}^{ck\times ck}, \mathbf{S}_R \in \mathbb{F}^{k\times ck}$ are secret structured matrices,
- $\Pi_L, \Pi_R$ are public random $ck \times ck$ permutation matrices.

An expansion factor $c > 1$ makes the final map $\mathbf{S}_L$ act as a randomness extractor. A similar idea was used in the construction of linear-size pairwise-independent hash functions from [IKOS08], which follows a similar blueprint. Note that if $c = 1$ and $\mathbb{F}$ is small, then the probability of $\mathbf{R}$ being

singular is noticeably bigger than a random matrix even when the secret matrices are random. This explains why we need $c > 1$.

Before specifying how to choose the secret matrices, we note the following common feature of our instantiations: each secret matrix is an affine (degree-1) function of $\mathsf{td}$, where $\mathsf{td}$ contains $\Omega(k)$ field elements. Thus, the mapping from $\mathsf{td}$ to $\mathbf{R}$ is a degree-3 polynomial map determined by $\Pi_L$ and $\Pi_R$. Since it is commonly conjectured that an overwhelming fraction of degree-3 polynomial maps $\mathbb{F}^{\Omega(k)} \mapsto \mathbb{F}^{k^2}$ define a pseudorandom generator (as a natural generalization of the standard MQ assumption [MI88]), the security of the above TDM would follow from the additional leap of faith that the same holds for the degree-3 map induced by a random choice of $\Pi_L, \Pi_R$.

It remains to specify the choices of the expansion factor $c$ and the secret structured matrices. We propose two instantiations of Eq. (1):

1. **Quasi-cyclic.** Here we take $c = 2$, let $\mathbf{S}_L, \mathbf{S}_R$ be quasi-cyclic matrices of the form $[\, I \,|\, \mathbf{C} \,]$, each defined by a random vector in $\mathbb{F}^k$, and $\mathbf{S}$ a circulant matrix defined by a vector in $\mathbb{F}^{2k}$. Note that once we fix $\Pi_1$ and $\Pi_2$, the evaluation circuit $\mathsf{EC}(\mathsf{td})$ has quasilinear size, since it computes $O(1)$ convolutions of vectors of length $O(k)$ interleaved with fixed permutations.

2. **Linear size.** Alternatively, we can pick the 3 secret matrices so that the corresponding linear maps can be implemented by linear-size circuits. A conservative choice would be to let $c \geq 2$ and pick each of the secret linear maps from a "randomizing" family that has a uniform output on every nonzero input. Concretely, consider a bilinear function $B : \mathbb{F}^{k_1} \times \mathbb{F}^{k_2} \to \mathbb{F}^{k_3}$ (here $k_i = \Theta(k)$) with the following properties: (1) for every fixed nonzero $\mathbf{v} \in \mathbb{F}^{k_2}$, the output of $B(\mathbf{s}, \mathbf{v})$ induced by a uniform choice of $\mathbf{s} \in \mathbb{F}^{k_1}$ is uniform in $\mathbb{F}^{k_3}$; (2) $B$ can be implemented by an arithmetic circuit of size $O(k_1 + k_2 + k_3)$. This can be used to instantiate Eq. (1) in the following way. Each of the 3 secret linear maps is defined by applying $B$ with the key $\mathbf{s}$ taken from $\mathsf{td}$ (an independent $\mathbf{s}$ for each map) and $\mathbf{v}$ as the input to this map. The output of $B$ is fed into the next step. An explicit $B$ as above is known to exist, and can be constructed from asymptotically good codes with linear-size encoders [IKOS08, DI14]. We conjecture that for most choices of $B$ as above (and in particular the explicit ones from the literature), the TDM defined by Eq. (1) is secure.

**An RAA-style TDM candidate.** Finally, we propose a leaner, but also more ad hoc, variant of Eq. (1) based on the design principle of Repeat-Accumulate-Accumulate (RAA) codes [BDMP98, DJM98, GM08]. Binary RAA codes were recently applied and concretely analyzed in the context of efficient cryptographic proof systems [BCF+24]. The RAA-style TDM realizes a linear map that starts by expanding the input, repeating each input entry $c$ times, then applies several iterations of a public permutation and a secret weighted accumulation (see below), and finally shrinks by outputting the sum of each block of length $c$. Standard RAA codes require $c \geq 3$, which is a natural choice here as well. More formally, the RAA-style TDM candidate is defined by

$$\mathbf{R} = \mathbf{D}^{\mathsf{T}} \cdot \Pi_4 \cdot \mathbf{S}_3 \cdot \Pi_3 \cdot \mathbf{S}_2 \cdot \Pi_2 \cdot \mathbf{S}_1 \cdot \Pi_1 \cdot \mathbf{D}, \tag{2}$$

where

- $\mathbf{D} \in \mathbb{F}^{ck \times k}$ is a fixed matrix that repeats each input $c$ times and its transpose $\mathbf{D}^{\mathsf{T}}$ shrinks $\mathbf{v} \in \mathbb{F}^{ck}$ to $\mathbf{v}' \in \mathbb{F}^k$ by adding the entries in each block of length $c$, namely $v'_i = \sum_{j=(c-1)i+1}^{ci} v_i$;
- $\Pi_i \in \mathbb{F}^{ck \times ck}$ are public, random and independent permutation matrices;

- $\mathbf{S}_i \in \mathbb{F}^{ck \times ck}$ are secret, random and independent weighted accumulation matrices, where for $n = ck$ and secret weight vector $\mathbf{w} \in \mathbb{F}^n$, such a matrix realizes the linear map $\mathbf{v} \in \mathbb{F}^n \mapsto \mathbf{v}' \in \mathbb{F}^n$ defined by $v'_i = \sum_{j=1}^{i} w_i v_i$.

Unlike all previous constructions, this TDM candidate assumes that $\mathbb{F}$ is large (concretely, that $|\mathbb{F}|^{-c}$ is considered negligible), to avoid a bad event where all $c$ copies of an input are accumulated with weight 0 in the first accumulation step. Alternatively, the weight vectors can be restricted to have nonzero entries.

The candidate TDM constructions in this section have the advantages of being universal, and can potentially beat the asymptotic and concrete efficiency of the previous constructions. In fact, the linear-size instantiations (if secure) have asymptotically optimal circuit size. On the down side, these are new and speculative designs, that require further analysis and tuning of parameters. We leave a more thorough study of the concrete security and efficiency of these and other TDM candidates to future work.

# 6 Distributing the EMVP Client

The fact that the EMVP client in our 1D-SLSN-based protocol is both lightweight and field-agnostic makes it easy to distribute between two or more "worker" parties by using standard secure multiparty computation (MPC) techniques. Concretely, we can use any MPC based on a linear secret-sharing scheme (LSSS) over $\mathbb{F}$ that supports multiplications and additions over $\mathbb{F}$, as captured by the "arithmetic black-box" model from [DN03]. This includes protocols in many different settings: both honest and dishonest majority, with security against both passive and active adversaries. We use the standard notation of $[\mathbf{v}]$ to denote LSSS shares of the vector $\mathbf{v}$, and assume that shared values can be added without interaction. This implies, for example, that a convolution of two vectors (corresponding to polynomial multiplication) can be performed with linear communication and quasilinear computation using FFT.

Although we can build on essentially any MPC protocol for arithmetic circuits from the literature, it is particularly attractive to combine our EMVP protocol with field-agnostic MPC protocols, which results in a field-agnostic distributed client protocol. Field-agnostic MPC includes most protocols in the honest-majority setting, such as the classical protocols of [BGW88, CCD88, RB89] and their more efficient variants, as well as dishonest-majority protocols in the OT-hybrid model [IPS09]. In the honest-majority setting, when the number of parties is not too large, one can further enhance the efficiency of such protocols using pseudorandom secret sharing (PRSS) [CDI05, BBC+19, BBG+21]. In the dishonest-majority setting, efficiency can be greatly enhanced using LPN-based pseudorandom correlation generators (PCGs) [BCGI18, BCG+19a, BCG+20, RRT23, LXYY25, KPRR25], which are also field agnostic.

We sketch the high-level details of a distributed-client variant of our main protocol (Fig. 1) here, and leave further optimization and implementation efforts to future work.

We start by assuming a trusted setup, where the server already holds the encrypted matrix $\widehat{\mathbf{M}}$ and the parties emulating the distributed client are given shares of the secret code $\mathbf{C}$. We assume here ring-LSN with $n = 2k$, in which case the encoding by $\mathbf{C}$ or its dual $\mathbf{D}$ are implemented by a polynomial multiplication over the ring $R = \mathbb{F}[X]/P(X)$ for a (typically irreducible) degree-$k$ polynomial $P$. We will still use matrix notation $\mathbf{C}$ for generality, with the understanding that in our default implementation $\mathbf{C}$ is defined by a vector in $\mathbb{F}^k$ that represents a polynomial in $R$.

**Distributing the decoding algorithm** $\mathsf{D}$**.** Given shares $[\mathbf{p}']$, $[\mathbf{r}']$ and a public $\mathbf{M}'$ obtained from the server, one can non-interactively compute fresh output shares $[\mathbf{a}] = [\mathbf{M}'\mathbf{p}' - \mathbf{r}'] + [\mathbf{0}]$ (where $[\mathbf{0}]$

are fresh shares of an all-0 vector). The shares $[\mathbf{a}]$ can either be reconstructed as the answer to the query, or used as input for a secure post-processing protocol.

**Distributing the query generation algorithm $\mathsf{Q}$.** In addition to the secret code $[\mathbf{C}]$ (viewed as a polynomial in $R$), the query algorithm is given a shared trapdoor $[\mathsf{td}]$ for a universal TDM; see below for an alternative implementation using a distributed variant of TDM.

Given the above shared inputs $[\mathbf{C}], [\mathsf{td}]$, the distributed query generation computes a public query $\widehat{\mathbf{q}}$ and shared decoding information $[\mathbf{q}']$ in the following way:

- Sample a random message $[\mathbf{r}]$ ($\mathbf{r} \in R$) and let $[\mathbf{c}] \leftarrow ([\mathbf{r}], [\mathbf{Cr}])$, where $\mathbf{Cr}$ is computed using a secure ring multiplication in $R$, which can be implemented efficiently using FFT over $\mathbb{F}$.
- Let $[\tilde{\mathbf{q}}] \leftarrow ([\mathbf{q}] \,|\, [0^k]) + [\mathbf{c}]$ and $[\mathbf{r}'] \leftarrow \mathsf{EC}([\tilde{\mathbf{q}}], [\mathsf{td}])$; evaluating $\mathsf{EC}$ (a small universal TDM circuit) requires a small number of atomic secure field operations. We describe optimized variants below.
- Generate random nonzero shared field elements $[\alpha_1], \ldots, [\alpha_s]$ and let $[\mathbf{p}'] \leftarrow ([\alpha_1^{-1}], \ldots, [\alpha_s^{-1}]) \in \mathbb{F}^s$; sampling random nonzero shared field elements and inverting them can be done using the standard technique from [BB89].
- Let $[\widehat{\mathbf{q}}] \leftarrow ([\alpha_1 \cdot \tilde{\mathbf{q}}_1] \,|\, \ldots \,|\, [\alpha_s \cdot \tilde{\mathbf{q}}_s]) \in \mathbb{F}^n$, where $(\tilde{\mathbf{q}}_1 \,|\, \ldots \,|\, \tilde{\mathbf{q}}_s)$ is a partition of $\tilde{\mathbf{q}}$ into blocks of length $n/s$; here we just need standard secure scalar-vector multiplication.
- Let $[\mathbf{q}'] = ([\mathbf{p}'], [\mathbf{r}'])$
- Return $(\widehat{\mathbf{q}}, [\mathbf{q}'])$, where $\widehat{\mathbf{q}}$ is reconstructed from $[\widehat{\mathbf{q}}]$.

**Distributing the matrix encryption $\mathsf{E}$ and the TDM** The matrix encryption algorithm is given as inputs $[\mathbf{D}]$, the dual code of $[\mathbf{C}]$, and a random shared trapdoor $\mathsf{td}$ for the TDM. Recall that in our ring-LSN implementation with $n = 2k$, the dual codes $[\mathbf{C}]$ and $[\mathbf{D}]$ are both defined by the same random polynomial in $R$. The encoded matrix $[\mathbf{MD}]$ is computed row-by-row using polynomial multiplication, as in the distributed query generation. The remaining challenge in computing $\widehat{\mathbf{M}} = \mathbf{MD} + \mathbf{R}$ is computing $[\mathbf{R}]$ from $[\mathsf{td}]$. This is done by using the trapdoor to evaluate $\mathbf{Rv}$ for each unit vector $\mathbf{v} \in \mathbb{F}^n$. Note that our TDM constructions mainly focus on the case of square matrices. This is without loss of generality, since any rectangular matrix can be covered by square matrices with at most 2x waste. In fact, when a distributed client needs to add new rows to the matrix, we can accommodate this by adding another square TDM to the cover.

We propose two concretely efficient methods for distributing the TDM evaluation as required above. The first is to use a *universal* TDM. A particularly attractive choice is the cyclic variant of our new universal TDM candidate from Section 5.2 (see Eq. (1)), which requires 3 multiplications of secret polynomials interleaved with public permutations. As before, polynomial multiplications in $R$ can be easily preprocessed using a suitable PCG. Each public permutation just requires the parties to shuffle their secret-shared entries as specified by the permutation.

Alternatively, we can rely on a non-universal LPN construction, such as the ring-LPN based construction from Section 5.1.1, in the following way. Consider a *distributed* TDM, defined as the sum of (standard) TDMs locally generated by each party, each with its own trapdoor $\mathsf{td}_i$. For instance, using the ring-LPN based TDM, each party $i$ picks a sparse matrix $\mathbf{E}_i$ and lets $\mathbf{R}_i = \mathbf{HE}_i$ be its additive share of $\mathbf{R}$, where $\mathbf{H}$ is a public quasi-cyclic matrix. Given $[\mathbf{R}]$ and $[\mathbf{DM}]$, the parties can locally compute $[\widehat{\mathbf{M}}] = [\mathbf{DM}] + [\mathbf{R}]$. To multiply $\mathbf{R}$ by a secret-shared vector $[\tilde{\mathbf{q}}]$, the parties apply their local TDMs to compute $\mathbf{HE}_i \cdot [\tilde{\mathbf{q}}]$ which form additive shares $[\mathbf{R}\tilde{\mathbf{q}}]$. Note that, unlike the

universal TDM approach, this distributed TDM solution does not natively support security against active adversaries.

# 7 Cryptanalysis

In this section we analyze the LSN-type assumptions we use with respect to some relevant types of algebraic attacks, which would allow us to propose concrete parameters and discuss the concrete efficiency in Section 7.4 below. (We note that in this section we use row vectors by default, vs. column vectors in the rest of the document.)

## 7.1 Algebraic Attacks

Here we devise some algebraic attacks on regular-LSN and split-LSN (the cases $s = 1$ and $s > 1$ from Definition 3.2). The setting that we analyze in both cases is having a random secret rank-$k$ code $C \subset \mathbb{F}^n$, generated by a matrix $\mathbf{C} \in \mathbb{F}^{k \times n}$ (which we almost always assume is systematic, $\mathbf{C} = [\mathbf{I}_k | \mathbf{X}]$ with $\mathbf{I}_k$ the $k \times k$ identity matrix). The adversary is given many samples (as many as it wants), where for each sample a random codeword $\mathbf{c} = \mathbf{rC}$ is chosen and the adversary is given either $d$ random vectors in $\mathbb{F}^n$ or $d$ random vectors whose span includes $\mathbf{c}$. (The parameter $d$ corresponds to $r \cdot s$ from Definition 3.2.)

The difference between regular-LSN and split-LSN is that in regular-LSN these $d$ vectors are uniform in $\mathbb{F}^n$, whereas in split-LSN each vector has only one non-zero block. That is, in split-LSN the index set $[n]$ is partitioned into $s$ blocks of size $b = n/s$ each, and each vector is uniform in one block and zero in all other blocks.

This way of choosing the samples is convenient for our analysis, but is slightly different than the way it is described in Definition 3.2. There we used dimension-$(d-1)$ affine space, where $d-1$ vectors $\mathbf{v}_i$ are chosen at random and we set $\mathbf{v}_d = \mathbf{c} + \sum_i \gamma_{i<d} \mathbf{v}_i$ for random scalars $\gamma_i \in \mathbb{F}$. Here, instead, we use dimension-$d$ linear space where the $d$ random vectors are chosen subject to $\mathbf{c} \in \mathsf{Span}(\mathbf{v}_1, \ldots, \mathbf{v}_d)$.

For regular-LSN ($s = 1$), the statistical difference between the two choices of samples is at most $1/|\mathbb{F}|$. Indeed, the only difference is that in the affine case the adversary knows that the linear combination yielding $\mathbf{c}$ has $\gamma_d = 1$, whereas in the linear case the coefficient $\gamma_d$ is also uniform in $\mathbb{F}^d$. But since $\mathbf{c}$ is itself a random codeword, then the only difference between these distributions is that in the linear case we allow $\gamma_d = 0$ whereas the affine case from Definition 3.2 we do not.

For Split-LSN ($s > 1$), the two distributions are different because for each block, the coefficient multiplying the last vector of the block may be different. However, the attacks that we describe below against the linear-space distribution imply an attack against the affine-space distribution from Definition 3.2. Indeed, a sample from the affine-space distribution can be transformed to a sample from the linear-space distribution by replacing the set of vectors for each block by a random generating set of vectors spanning the same subspace.

Finally, for the 1D-SLSN case from Definition 3.3, all our attacks use shifts equal to zero. It is unclear how to use shifts to improve the attacks.

Below we describe an algebraic attack in a framework that applies to both $s = 1$ and $s > 1$ cases, and then analyze separately the complexity of this attack for regular-LSN ($s = 1$) and for split-LSN ($s > 1$). This attack follows the common pattern of tensor-based rank attacks, which succeed in finding a nonzero degree-$d$ polynomial that annihilates all samples, if one exists, in time roughly $m^d$, where $m$ is the length of each sample. (As we note in Section 7.1.1 below, this attack can be thought of as an adaptation of the Arora-Ge type of attacks [AG11] to our secret-code setting.) The attack uses the following simple lemma:

**Lemma 7.1.** *Fix a field $\mathbb{F}$ and parameters $d, k, n$ such that $d \leq n - k$. For every rank-$k$ code $C \subset \mathbb{F}^n$ there exists a non-zero degree-$d$ multivariate polynomial $P_C(\cdot)$, such that $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = 0$ holds for every set of vectors that non-trivially span vectors from $C$ (in other words: if there exist coefficients $\gamma_i$ that are not all zero such that $\sum_i \gamma_i \mathbf{v}_i \in C$, then $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = 0$).*

*Proof.* Let $\pi : \mathbb{F}^n \to \mathbb{F}^d$ be any linear map such that $C \subseteq \mathsf{Kernel}(\pi)$ and $\mathsf{Image}(\pi) = \mathbb{F}^d$ (i.e., $\pi$ is of full rank). Such a linear map exists since $\dim(C) + \dim(\mathbb{F}^d) = k + d \leq n$. For example, let $\mathbf{c}_1, \ldots, \mathbf{c}_k$ be a basis of $C$, and let $\mathbf{c}_{k+1}, \ldots, \mathbf{c}_n$ be an extension of it to a basis of $\mathbb{F}^n$. Then define $\pi(\mathbf{c}_i) = \mathbf{0}$ for $i \leq n-d$ and $\pi(\mathbf{c}_i) = \mathbf{e}_{i-n+d}$ for $i \geq n-d+1$ (where $\mathbf{e}_j$ is the $j$'th unit vector in $\mathbb{F}^d$).

For any set of $d$ vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d \in \mathbb{F}^n$, let $\mathbf{V}_\pi$ be the $d$-by-$d$ matrix over $\mathbb{F}$ with the column vectors $\pi(\mathbf{v}_i)^\intercal$ as columns: $\mathbf{V}_\pi = [\pi(\mathbf{v}_1)^\intercal | \cdots | \pi(\mathbf{v}_d)^\intercal]$. We define the polynomial $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = \det(\mathbf{V}_\pi)$, which is indeed a non-zero degree-$d$ multivariate polynomial. ($P_C$ is non-zero since $\pi$ is full rank.)

It remains to show that $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = 0$ whenever $\sum_{i=1}^d \gamma_i \mathbf{v}_i \in C$ for coefficients $\gamma_i$ that are not all zero. As $C$ is in the kernel of $\pi$, we have

$$\sum_{i=1}^{d} \gamma_i \, \pi(\mathbf{v}_i) = \pi\big(\sum_{i=1}^{d} \gamma_i \mathbf{v}_i\big) = 0,$$

and since the $\gamma_i$ are not all zero then it means that the $\pi(\mathbf{v}_i)$'s are linearly dependent. Therefore, the determinant of $\mathbf{V}_\pi$ must be zero. Thus $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = \det([\pi(\mathbf{v}_1)^\intercal | \cdots | \pi(\mathbf{v}_d)^\intercal]) = 0$. □

**The basic attack.** We note that regardless of $C$ or $\pi$, the monomials in $\det([\pi(\mathbf{v}_1)^\intercal | \cdots | \pi(\mathbf{v}_d)^\intercal])$ are always a subset of the entries of the tensor $\widehat{\mathbf{v}} = \mathbf{v}_1 \otimes \mathbf{v}_2 \otimes \cdots \otimes \mathbf{v}_d$ (that has dimension $n^d$). Hence for every code $C$ there is a vector $\mathbf{w}_C$ (of the coefficients of $P_C$) such that $\langle \mathbf{w}_C, \widehat{\mathbf{v}} \rangle = 0$ for every sample $\widehat{\mathbf{v}} = \mathbf{v}_1 \otimes \mathbf{v}_2 \otimes \cdots \otimes \mathbf{v}_d$ that the adversary sees. This means that the $\widehat{\mathbf{v}}$'s from all the queries live in a proper subspace of $\mathbb{F}^{n^d}$. After seeing at most $n^d$ samples, the adversary can see that their rank is strictly less than $n^d$, thus distinguishing them from random. (If the $\mathbf{v}_i$'s were random then whp the $\widehat{\mathbf{v}}$ from different samples are not contained in any proper subspace of $\mathbb{F}^{n^d}$.)

To mount this attack, the adversary needs to check linear dependence between $n^d$ vectors, but these are highly structured vectors that can each be described by at most $nd$ scalars. Hence we consider the complexity of this attack to be "essentially $n^d$". (Here and elsewhere, we use the conservative convention of assuming the attacker can solve a system of equations in linear time.)

Note that $n^d$ is only an upper bound, and below we show that essentially the same attack can be carried out with lower complexity. To that end, we exhibit classes of linear maps $\pi$ satisfying the conditions in the proof of Lemma 7.1, where the set of possible monomials in $\det(\mathbf{V}_\pi)$ is smaller than $n^d$.

### 7.1.1 Relations to Algebraic Attacks from the Literature

As mentioned above, we can view our algebraic attacks as variants of the Arora-Ge attacks [AG11] on LPN, adjusted to handle our setting where the code is secret. In the attacks from [AG11], the LPN secret $\mathbf{s}$ plays the same role as the code $C$ in our setting. They consider multiple LPN samples of the form $\mathbf{y}_i = \mathbf{A}_i\mathbf{s} + \mathbf{e}_i$ where the noise $\mathbf{e}_i$ is "structured" enough to have a non-zero degree-$d$ annihilating polynomial $\widehat{P}$. Since the matrix $\mathbf{A}_i$ is known, then they can transform this noise-annihilating polynomial into secret-dependent polynomial $P_\mathbf{s}(\mathbf{A}, \mathbf{y}) = \widehat{P}(\mathbf{y} - \mathbf{A}\mathbf{s})$ (of the same degree as $\widehat{P}$) that a annihilates all the samples, and use tensor-based rank attacks to find $P_\mathbf{s}$ given enough samples. In our case, it is the code $C$ which is secret. We similarly show that there

exists a code-dependent non-zero degree-$d$ polynomial $P_C$ that annihilates all the samples, and use tensor-based rank attacks to find $P_C$ given enough samples.

We also remark that more sophisticated attacks based on Gröbner bases/XL [CKPS00] do not seem to help in our case. These techniques are useful in situations where the number of samples available to the attacker is limited, allowing to better utilize the limited samples at the price of increasing the degree of the annihilating polynomials (and thus the complexity of the attack). This is useful in the context of PCGs (such as in [BØ23, KPRR25]) where the number of samples is fixed by the construction itself. But in our case we allow the adversary to draw as many samples as it wants, so we get attacks with better complexity by using more samples than by using Gröbner/XL.

### 7.1.2 Complexity of the Split-LSN Attack

In this subsection we use the notations from the definitions of split-LSN in Section 3. Specifically, the index set $[n]$ is split into $s$ blocks of dimension $b = n/s$, and each block is hidden in a linear space of dimension $r$. Thus the total number of vectors is $d = s \cdot r = nr/b$.

To be able to use the attack from above in this setting, we therefore need the condition $n \geq k+d$, or equivalently $n \geq kb/(b-r)$. In fact if we let $n' \leq n$ be the smallest multiple of $b$ such that $n' \geq kb/(b-r)$, then we can truncate all the vectors to dimension $n'$ and the attack will still work. Below we therefore assume that the length $n$ is exactly that, $n = b \cdot \lceil k/(b-r) \rceil$, so that the number of blocks is $s = \lceil k/(b-r) \rceil$ and the total number of vectors is $d = rs = r \lceil k/(b-r) \rceil$.

We can now use a straightforward application of the attack, and notice that since each of the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ is supported only on $b$ coordinates then the tensor $\otimes_i \mathbf{v}_i$ has only $b^d = b^{r\lceil k/(b-r) \rceil}$ non-zero terms. The attack complexity in this case is therefore essentially $b^{r\lceil k/(b-r) \rceil}$. In particular, for the most aggressive setting of $r = 1$, we get complexity of $b^{\lceil k/(b-1) \rceil}$. If $k$ is divisible by $b$ and $k < b^2$, then $\lceil k/(b-1) \rceil = 1 + k/b$ so we get complexity of $b^{1+k/b}$.

We note that the reduction in the base of the exponent (from $n$ to $b$) is only possible because the partition into blocks is fixed and does not change from one sample to the next. If instead for each sample we choose a different partition of the index set $[n]$ into $s$ blocks of size $b$ each, then we seem to have to consider all the possible tensors, none of them is always zero. In that case, the best bound that we know is that for a generic regular-LSN, which as we show below is $(k+1)^d$.[5]

### 7.1.3 Complexity of the Regular-LSN Attack

We show that even for the general regular-LSN problem where the $\mathbf{v}_i$'s are chosen at random (subject to $\mathsf{Span}(\mathbf{v}_1, \ldots, \mathbf{v}_d) \cap C \neq \{\mathbf{0}\}$), the complexity of the attack from above can be lowered from $n^d$ to $(k+1)^d$. We assume that the code is generated by a systematic matrix $\mathbf{C} = [\mathbf{I}_k | \mathbf{X}]$, and extend $\mathbf{C}$ with $n-k$ additional rows to get a basis $M$ for $\mathbb{F}^n$.

Below it is convenient to view a vector $\mathbf{w} \in \mathbb{F}^n$ as a pair $(\mathbf{u}|\mathbf{v}) \in \mathbb{F}^k \times \mathbb{F}^{n-k}$. Consider now the linear map $\pi(\mathbf{u}|\mathbf{v}) = \mathbf{v} - \mathbf{u}\mathbf{X} \in \mathbb{F}^d$. A codeword $(\mathbf{u}|\mathbf{v}) \in C$ is generated as $(\mathbf{u}|\mathbf{v}) = \mathbf{r}\mathbf{C} = (\mathbf{r}|\mathbf{r}\mathbf{X})$ and therefore $\pi(\mathbf{u}|\mathbf{v}) = \mathbf{v} - \mathbf{u}\mathbf{X} = \mathbf{r}\mathbf{X} - \mathbf{r}\mathbf{X} = \mathbf{0}$, so $C$ is in the kernel of $\pi$. At the same time, the image of $\pi$ contains all of $\mathbb{F}^d$ since any vector $\mathbf{w} \in \mathbb{F}^d$ can be obtained as $\mathbf{w} = \pi(\mathbf{0}|\mathbf{w})$. This $\pi$ therefore satisfies the conditions of Lemma 7.1, and we can use it in the attack from above. With this $\pi$, the polynomial $P_C$ from Lemma 7.1 is the determinant

$$P_C((\mathbf{u}_1|\mathbf{v}_1), \ldots, (\mathbf{u}_d|\mathbf{v}_d)) = \det\left([(\mathbf{v}_1 - \mathbf{u}_1\mathbf{X})^\intercal | \cdots | (\mathbf{v}_d - \mathbf{u}_d\mathbf{X})^\intercal]\right),$$

[5] Even for blocks that are rather structured but not quite fixed, we do not know how to do better than $(k+1)^d$. For example, this is the case when the index set $[n]$ is partitioned into $2s$ fixed half-blocks of size $b/2$ each, and for each sample we assemble $s$ blocks, each a random pair of those half-blocks.

which is a sum of all the diagonal products of that matrix with coefficients $\pm 1$. Each one of these diagonal products has the form $\prod_{i=1}^{d}(v_{ji} - \langle \mathbf{u}_j, \mathbf{x}_i \rangle)$ where the $j$'s are some permutation of the $i$'s (and the $\mathbf{x}_i$'s are columns of the matrix $\mathbf{X}$).

For every subset $S \subseteq \{1, \ldots, d\}$, the determinant therefore has a term corresponding the subset tensor $\mathbf{x}_S = \otimes_{i \in S} \mathbf{x}_i$ (in order), and each such term is multiplied by a sum of tensors of the $\mathbf{u}_i$'s and $\mathbf{v}_i$'s (of dimension $k^{|S|}$). For example, when $d = 2$ then the matrix $A$ is

$$A = \begin{bmatrix} v_{1,1} - \langle \mathbf{u}_1, \mathbf{x}_1 \rangle & v_{2,1} - \langle \mathbf{u}_2, \mathbf{x}_1 \rangle \\ v_{1,2} - \langle \mathbf{u}_1, \mathbf{x}_2 \rangle & v_{2,2} - \langle \mathbf{u}_2, \mathbf{x}_2 \rangle \end{bmatrix}$$

and so its determinant is

$$\begin{aligned} \det(a) &= v_{1,1}v_{2,2} - v_{2,1}v_{1,2} + \langle v_{1,2}\mathbf{u}_2 - v_{2,2}\mathbf{u}_1,\ \mathbf{x}_1 \rangle - \langle v_{1,1}\mathbf{u}_2 - v_{2,1}\mathbf{u}_1,\ \mathbf{x}_2 \rangle \\ &\quad + \langle \mathbf{u}_1 \otimes \mathbf{u}_2 - \mathbf{u}_2 \otimes \mathbf{u_1},\ \mathbf{x}_1 \otimes \mathbf{x}_2 \rangle \\ &= \big\langle ((v_{1,1}v_{2,2} - v_{2,1}v_{1,2}) \mid (v_{1,2}\mathbf{u}_2 - v_{221}\mathbf{u}_1) \mid (v_{2,1}\mathbf{u}_1 - v_{1,1}\mathbf{u}_2) \mid (\mathbf{u}_1 \otimes \mathbf{u}_2 - \mathbf{u}_2 \otimes \mathbf{u_1})), \\ &\qquad (\quad 1 \quad \mid \quad \mathbf{x}_1 \quad \mid \quad \mathbf{x}_2 \quad \mid \quad \mathbf{x}_1 \otimes \mathbf{x}_2 \quad ) \big\rangle. \end{aligned}$$

More generally, the last big inner product has one term of dimension 1 (the sum of all the terms $\prod_i v_{ij}$), $d$ terms of dimension $k$ (for terms that have $d-1$ factors $v_{ij}$'s and one $\langle \mathbf{u}_j, \mathbf{x}_i \rangle$ for each $i = 1, \ldots, d$), $\binom{d}{2}$ terms of dimension $k^2$, etc. The total dimension is therefore

$$\sum_{i=0}^{d} \binom{d}{i} k^i = (k+1)^d.$$

The attack in this case consists of collecting enough of these vectors of dimension $(k+1)^d$ until is finds linear dependence. This would happen after at most $(k+1)^d$ regular-LSN samples, since these samples live in a proper subspace of a space of dimension $(k+1)^d$ (but whp only after $(k+1)^d + 1$ random samples). Hence the attack complexity is essentially $(k+1)^d$.

### 7.2 Can We Have Better Attacks?

Below we bring some evidence that this type of algebraic attacks cannot be made much cheaper. Specifically, we show that there are no annihilating polynomials as per Lemma 7.1 with degree smaller than $d$.

**Lemma 7.2.** *For any rank-$k$ code $C \subset \mathbb{F}^n$ and positive integer $d$, the only multivariate polynomial $P$ of degree $\leq d-1$ such that $P(\mathbf{w}_1, \ldots, \mathbf{w}_d) = 0$ whenever $\mathbf{w}_1, \ldots, \mathbf{w}_d$ non-trivially span a codeword in $C$, is the zero polynomial $P \equiv 0$.*

*Proof.* We first prove this lemma for the specific rank-$k$ code $C'$ generated by the matrix $\mathbf{C}' = [\mathbf{I}_k|0]$, i.e., the subspace of all the vectors whose last $n-k$ coordinates are zero. We denote each $n$-vector $\mathbf{w} \in \mathbb{F}^n$ as $\mathbf{w} = (\mathbf{u}|\mathbf{v})$ where $\mathbf{u} \in \mathbb{F}^k$ and $\mathbf{v} \in \mathbb{F}^{n-k}$. Recall that $(\mathbf{u}_1|\mathbf{v}_1), \ldots, (\mathbf{u}_d|\mathbf{v}_d)$ non-trivially span a codeword in $C'$ if for some coefficients $\gamma_i$, not all zero, it holds that $\sum_i \gamma_i(\mathbf{u}_i|\mathbf{v}_i) \in C'$, which happens if and only if the $\mathbf{v}_i$'s are linearly dependent. In particular, they non-trivially span a codeword in $C'$ whenever one of these vectors is of the form $(\mathbf{u}_i|\mathbf{v}_i = \mathbf{0})$.

Fix a multivariate polynomial $P(\mathbf{w}_1, \ldots, \mathbf{w}_d)$ of degree less than $d$ that vanishes whenever the vectors non-trivially span a codeword in $C'$, and consider its monomials. Since it has degree less than $d$ then for each monomial $M \in P$ there is at least one vector $\mathbf{w}_i = (\mathbf{u}_i|\mathbf{v}_i)$ such that $M$ is

disjoint from the entries in $\mathbf{v}_i$. For each $i \in [d]$ denote by $P_i$ the polynomial consisting of all the monomials that are disjoint of $\mathbf{v}_i$, so we can write $P_i(\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \mathbf{u}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_d)$. Consider now any arbitrary assignment to the variables of $P_i$, and also set $\mathbf{v}_i = \mathbf{0}$. We note the following about this assignment:

- Since $\mathbf{v}_i = \mathbf{0}$ and $P_i$ contains all the monomials in $P$ that are disjoint from $\mathbf{v}_i$, then

$$P(\mathbf{w}_1, \ldots, \mathbf{w}_d) \;=\; P_i(\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \mathbf{u}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_d)$$

- Since $\mathbf{v}_i = \mathbf{0}$ then $\mathbf{w}_1, \ldots, \mathbf{w}_d$ non-trivially span a codeword in $C'$, hence $P(\mathbf{w}_1, \ldots, \mathbf{w}_d) = 0$.

It follows that $P_i(\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \mathbf{u}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_d) = 0$ for any assignment to $(\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \mathbf{u}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_d)$, so $P_i$ is the zero polynomial, $P_i \equiv 0$. Since this holds for all $i \in [d]$, and every monomial in $P$ belongs to at least one of the $P_i$'s, then $P$ is also the zero polynomial.

Having proven the lemma for $C'$ (which is spanned by the rank-$k$ matrix $\mathbf{C}'$), consider now some other code $C$ which is spanned by some other rank-$k$ matrix $\mathbf{C} \in \mathbb{F}^{k \times n}$. Since $\mathbf{C}, \mathbf{C}'$ have the same rank $k$, then there exists an invertible $n$-by-$n$ matrix $\mathbf{A}$ such that $\mathbf{C}' = \mathbf{C}\mathbf{A}$ (this follows from the fact that $\mathbf{C}'$ can be obtained from $\mathbf{C}$ by doing a series of elementary column operations, each of these operations corresponding to a multiplication by an invertible matrix on the right). Assume now that there is a non-zero multivariate polynomial $P$ of degree less than $d$ that vanishes on any set of vectors $\mathbf{w}_1, \ldots, \mathbf{w}_d$ that non-trivially span a codeword in $C$. We define the polynomial $P'(\mathbf{w}_1, \ldots, \mathbf{w}_d) = P(\mathbf{w}_1\mathbf{A}, \ldots, \mathbf{w}_d\mathbf{A})$, and note that $P'$ too is a multivariate polynomial of degree less than $d$. Moreover, fix any set of vectors $\mathbf{w}_1, \ldots, \mathbf{w}_d$ that non-trivially span a codeword in $C'$, namely $\sum_i \gamma_i \mathbf{w}_i \in C'$ when not all the $\gamma_i$'s are zero. Then there is some $k$-vector $\mathbf{r}$ such that $\sum_i \gamma_i \mathbf{w}_i = \mathbf{r}\mathbf{C}'$, and therefore

$$\sum_i \gamma_i(\mathbf{w}_i\mathbf{A}) = \mathbf{r}\mathbf{C}'\mathbf{A} = \mathbf{r}\mathbf{C} \in C.$$

It follows that the vectors $\mathbf{w}_i\mathbf{A}$ non-trivially span a vector in $C$, which implies that

$$P'(\mathbf{w}_1, \ldots, \mathbf{w}_d) = P(\mathbf{w}_1\mathbf{A}, \ldots, \mathbf{w}_d\mathbf{A}) = 0.$$

By the proof for the special case of $C'$, this implies that $P' \equiv 0$, and since $\mathbf{A}$ is invertible then for all $\mathbf{w}_1, \ldots, \mathbf{w}_d$ we have $P(\mathbf{w}_1, \ldots, \mathbf{w}_d) = P'(\mathbf{w}_1\mathbf{A}^{-1}, \ldots, \mathbf{w}_d\mathbf{A}^{-1}) = 0$, so $P$ is also the zero polynomial. □

### 7.3 Other Attacks

We next sketch some other attacks that are not special cases of the general attack from Section 7.1. While these attacks are not very relevant for our range of parameters, there may be other setting where one need to take them into account. For the sake of simplicity, we present those attacks in the regular-LSN setting ($s = 1$ and $d = r$). The same attacks also work in the split-LSN setting ($s > 1$ and $d = rs$). Recall indeed that the only difference between regular-LSN and split-LSN is that split-LSN vectors are more structured and are zeros everywhere except on the matching block. This structure does not hinder the attacks below (and to our knowledge, does not allow significantly more efficient attacks either).

### 7.3.1 Exhaustive search

One obvious class of attacks is just brute-force search: The attacker tries to guess the secret code $C$ and check the samples against each guess. Clearly, once the correct code $C$ was guessed then the attacker can verify that the vectors given in each sample indeed non-trivially intersect $C$. It is enough to guess only the first $k+1$ columns of the code, so the complexity of this attack is $|\mathbb{F}|^{k(k+1)}$. Another option is to guess a few vectors of the code (say $t$ of them) and running the attack from above for the remaining $(k-t)$-dimensional code. But the gain from using a lower-rank code does not seem to justify the expense of having to make $|\mathbb{F}|^{t(k+1)}$ guesses.

### 7.3.2 Inclusion/Exclusion

Another set of attacks, that seem to only apply when $k \ll n$, involve taking the intersections and unions of the linear spaces from multiple samples.

**The basic inclusion/exclusion attack.** The basic form of such attacks applies only when $k \cdot d < n$. (Recall that $d$ is the number of vectors in each sample, namely the dimension of the linear space in which we hide the codeword. In the notations from Definitions 3.2 and 3.3 we have $d = rs$.) Since the linear space spanned by the $d$ vectors in each sample intersects the code $C$ non-trivially, then taking the union of the vectors from a few samples (say $t$ of them) yields with high probability a linear space $V$ of dimension $\min(n, td)$ such that $V \cap C$ has dimension $\min(k, t)$. In particular, if $kd < n$ then taking the union of $k$ samples yields a proper subspace $V \subset \mathbb{F}^n$ that contains the entire code $C$ with high probability. Repeating this union $n$ times and taking the intersection of these $n$ unions, we recover the code $C$ with high probability.

**Extended form.** When $kd$ is just slightly larger than $n$, the inclusion/exclusion attack can sometimes still be used in conjunction with the algebraic attacks from above. Specifically, for parameters $n, k, d$ such that $k \geq 3$, $(k-1)d < n$ and $(k+1)d = n + \delta$, we can transform a given dimension-$d$ regular-LSN into a different regular-LSN instance with $n' = 2d$, $k' = 2$ and $d' = \delta$. We then can apply the algebraic attacks form above to the new problem, obtaining an attack with complexity essentially $(k'+1)^{d'} = 3^{\delta}$, which may be better than directly attacking the original instances with (complexity $(k+1)^d$). The transformation works as follows:

1. Draw two samples of the original regular-LSN, denote them $\mathbf{v}^*_1, \ldots, \mathbf{v}^*_{2d}$. Define the subspace $V^* = \mathsf{Span}(\mathbf{v}^*_1, \ldots, \mathbf{v}^*_{2d})$. We also denote $C^* = V^* \cap C$. With high probability, $\dim(V^*) = 2d < n$ and $\dim(C^*) = 2$.

2. Repeat the following many times, for $i = 1, 2, 3, \ldots$:

   (a) Draw $k-1$ samples of the original regular-LSN, denote them $\mathbf{v}^i_1, \ldots, \mathbf{v}^i_{(k-1)d}$, and let $V^i = \mathsf{Span}(\mathbf{v}^i_1, \ldots, \mathbf{v}^i_{(k-1)d})$ and $C^i = V^i \cap C$. We have $\dim(V^i) \leq (k-1)d < n$, and with high probability $\dim(C^i) = k-1$.

   (b) Compute the intersection $W^i = V^i \cap V^*$. Then $\dim(W^i) = \dim(V^*) + \dim(V^i) - \dim(V^* + V^i)$ which with high probability is at most $2d + (k-1)d - n = (k+1)d - n = \delta$. Let $\mathbf{w}^i_1, \ldots, \mathbf{w}^i_\delta$ be a basis for $W^i$.
   At the same time, we have $W^i \cap C = (V^i \cap C) \cap (V^* \cap C)$, and so

   $$\dim(W^i \cap C) = \dim(V^* \cap C) + \dim(V^i \cap C) - \dim((V^* + V^i) \cap C) = 2 + (k-1) - k = 1$$

   (where the penultimate equality above holds with high probability).

Viewing all the $W^i$'s as sub-spaces of the dimension-$2d$ space $V^*$, they all have dimension $\delta$ and they all intersect non-trivially the rank-2 code $C^* \subset V^*$. Hence we have a new regular-LSN problem with dimensions $n' = 2d, k' = 2$ and $d' = \delta$, as promised. (We can also randomly project everything down into $\mathbb{F}^{n'}$, with high probability this will maintain all these dimensions while bringing everything back to standard representation where the "universe" is just $\mathbb{F}^{n'}$.)

More generally, let $1 \le \alpha \le k$. Assume that $(k-\alpha)d < n$ and $(k+1)d = n+\delta$. (The above attack corresponds to the case $\alpha = 1$. Note the constraint $(k-\alpha)d < n$ is weaker than $(k-1)d < n$ in the above attack.) Then we can extend the above attack to transform a given dimension-$d$ regular-LSN into a different regular-LSN instance with $n' = \min(n, (\alpha+1)d)$, $k' = \alpha+1$, and $d' = \delta$. Concretely, instead of drawing two samples in step 1, we draw $\alpha + 1$ samples, and instead of drawing $k - 1$ samples in step 2a, we draw $k - \alpha$ samples. Then we can apply the previous algebraic attack to obtain an attack with complexity essentially $(k'+1)^{d'} = (\alpha+2)^{\delta}$.

**Limitations.** While very efficient, these inclusion/exclusion attacks seem to only apply when $kd$ is not much larger than $n$. Hence, as far as we see, these attacks are not relevant for our parameter regime of interest.

### 7.4 Parameters

We recall the parameters of our "most aggressive" (and most efficient) construction based on 1D-Split-LSN:

- The matrix dimensions are $m \times \ell$: we have $m$ records, each a dimension-$\ell$ vector over the field $\mathbb{F}$;
- Records are encoded using an $(\ell, n)$-linear code $D$ (so the server's overhead is a factor of $n/\ell$).
- The "query code" that we want to hide using 1D-Split-LSN is the dual of $D$, which we denote by $C$. Letting $k = n - \ell$, this a $(k, n)$-linear code over $\mathbb{F}$.
- We split the length-$n$ codewords in $C$ into $s$ blocks, each of length $b = n/s$.

We denote by $\lambda$ the security parameter, and choose parameters so that the attacks from this section all have complexity more than $2^\lambda$. For the 1D-Split-LSN constructions we need $b^{\lceil k/(b-1) \rceil} \ge 2^\lambda$ due to the algebraic attack from Section 7.1.2. We also need $(n/b + 1) \cdot k \ge n + \lambda$ to ensure that we cannot reduce the complexity using the inclusion/exclusion attacks from Section 7.3, but this will usually be a weaker constraint.

Finally, due to efficiency considerations, we want to minimize server overhead factor $f = n/\ell$: This parameter controls the storage and computation overhead of the server (vs. storing the plaintext matrix and computing the plaintext matrix vector product), as well as the increase of query upload bandwidth. In addition, we also want to minimize the number of blocks $s$ (which means maximizing the block size $b$), as it controls the size of the server's answer and the client's decoding work. Concretely, the *compression ratio* for the download bandwidth compared to the naive solution of sending the full matrix is $\ell/s = b/f$, since the server returns $s$ field elements per row, instead of $\ell$ elements in the naive solution.

To get concrete parameters, we can begin by deciding the server overhead factor that we are willing to tolerate, which determines the ratio between $k$ and $\ell$. To wit, server overhead of $f\times$ implies $k = \ell(f-1)$ (up to rounding). For example, a $4\times$ server overhead means $k = 3\ell$, while overhead of $1.25\times$ means $k = \ell/4$, etc.

Rewriting the constraint $b^{\lceil k/(b-1)\rceil} \geq 2^\lambda$ as $\lceil k/(b-1)\rceil \log_2 b \geq \lambda$, we can ignore the ceiling and get the sufficient condition $k \geq \lambda \cdot (b-1)/\log_2 b$. This constraint, together with $k = \ell(f-1)$, implies the following:

- We recall that we need $b > f$ to get a more efficient download bandwidth than the trivial protocol that just sends the entire matrix to the client for every query, and since $b$ is an integer then $b \geq \lfloor f \rfloor + 1$ (and in particular $b \geq 2$). For $f < 2$ this means that $k \geq \lambda$ and therefore $\ell \geq \lambda/(f-1)$. In general we have $k \geq \lambda \cdot \lfloor f \rfloor / \log_2(\lfloor f \rfloor + 1)$ and $\ell \geq k/(f-1)$.

  For example, server overhead $f = 4$ implies $k \geq 4\lambda/\log_2(5) \approx 1.7227\lambda$ and $f \geq k/3 \approx 0.5742 \cdot \lambda$. For another example, server overhead $f = 1.25$ yields $k \geq \lambda$ and $\ell \geq 4\lambda$.

- For any value of $\ell$ above this minimal value, we can maximize the compression ratio $b/f$ by using the largest value of $b$ that satisfies $(b-1)/\log_2 b \leq k/\lambda$. For example, with $\lambda = 128$ and $f = 4$, the smallest value of $\ell$ is 73, but if we have a larger value of (say) $\ell = 1024 = 8\lambda$, then we can get block-size $b = 180$. Rounding up so that $n$ is a multiple of $b$, we have $k = 3116$ and $n = 4140$.

We list some parameter examples in Table 1, from the smallest possible record size $\ell$ up to $\ell = 10000$ (which is roughly the size used in the application from [BKSW24]).

### 7.4.1 Can We Get Better Parameters?

As we commented at the bottom of Section 7.1.2, the algebraic attacks seem to become more expensive if instead of fixed blocks, we choose the partition into blocks at random for each sample separately. In that case, the attack complexity becomes $(k+1)^{\lceil k/(b-1)\rceil}$ (instead of $b^{\lceil k/(b-1)\rceil}$), which enables significantly better parameters.

Rewriting the last constraint as $\lceil k/(b-1)\rceil \log_2(k+1) \geq \lambda$, we ignore the $+1$ and ceiling and get the sufficient condition $k \log_2 k \geq \lambda \cdot (b-1)$. Substituting $b = \lfloor f \rfloor + 1$ yields a lower bound on $k$, and then $k = \lceil \ell(f-1) \rceil$ yields a lower bound on $\ell$. For values of $\ell$ above that lower bound, we can set $k = \lceil \ell(f-1) \rceil$ and $b = 1 + \lfloor k \log_2 k/\lambda \rfloor$.

For example with $\lambda = 128$, for server overhead $f = 1.25$ we need at least $b = 2$, so $k \log_2 k > \lambda = 128$ or $k \geq 28$, and therefore $\ell = k/(f-1) = 108$ (vs. $\ell = 74$ with fixed blocks). With the same $f = 1.25$ and $\lambda = 128$, if we have $\ell = 512$ (so $k = 128$) then the block-size parameter $b$ can be as large as $b = 1 + \lfloor 128 \cdot \log_2(128)/128 \rfloor + 1 = 8$ (vs. $b = 2$ with fixed blocks). We include a few parameter-setting examples for this variant in Table 1.

As an alternative to using a random block partition in each query, the server can apply a random permutation to the columns of the encoded matrix once the number of queries exceeds the threshold allowed by the fixed-partition analysis. This has the advantage of better cache-friendliness (only requiring operations on contiguous blocks) but the disadvantage of requiring the client to know the current column permutation that needs to be applied to the queries.

**Acknowledgements.** We thank Benny Applebaum, Nico Döttling, Alexandra Henzinger, Aayush Jain, Mayank Rathee, Nicolas Resch, and David Wu for helpful discussions and pointers.

Caicai Chen, Tamer Mour and Alon Rosen are supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

Table 1: Parameter settings for the 1D-LSN construction (Fig. 1) with $\lambda = 128$. The encrypted query is $\widehat{q} \in \mathbb{F}^n$, the server's computation is comparable to multiplying $\widehat{\mathbf{M}} \in \mathbb{F}^{m \times n}$ by $\widehat{\mathbf{q}} \in \mathbb{F}^n$, the answer is $\mathbf{M}' \in \mathbb{F}^{m \times s}$ for $s = n/b$, and the client's local work (with ring-LSN variant) is $\tilde{O}(n) + 2sm$ field operations.

| record length $\ell$ | compression ratio $b/f$ | code params $(k, n)$ |
|---|---|---|
| Server overhead factor $f = 4$, min $\ell = 73$ | | |
| 73 | 5/4=1.25 | (222,295) |
| 128 | 11/4=2.75 | (389,517) |
| 512 | 75/4=18.75 | (1588,2100) |
| 1024 | 180/4=45 | (3116,4140) |
| 10000 | 2668/4=667 | (30020,40020) |
| Server overhead factor $f = 1.25$, min $\ell = 512$ | | |
| 512 | 2/1.25=1.6 | (128,640) |
| 1024 | 6/1.25=4.8 | (260,1284) |
| 10000 | 140/1.25=112 | (2600,12600) |
| Random block partition with $f = 1.25$, min $\ell = 108$ | | |
| 108 | 2/1.25=1.6 | (28,136) |
| 512 | 8/1.25=6.4 | (128,640) |
| 1024 | 17/1.25=13.6 | (268,1292) |
| 10000 | 221/1.25=176.8 | (2597,12597) |

# References


[AAB17] Benny Applebaum, Jonathan Avron, and Chris Brzuska. Arithmetic cryptography. *J. ACM*, 64(2):10:1–10:74, 2017. 5, 10

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011. 13, 26, 27

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

[BB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*, pages 201–209. ACM, 1989. 25

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 67–97, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 24

[BBG+21] Fabrice Benhamouda, Elette Boyle, Niv Gilboa, Shai Halevi, Yuval Ishai, and Ariel Nof. Generalized pseudorandom secret sharing and efficient straggler-resilient secure computation. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 129–161, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland. 24

[BCF+24] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast snarks from interleaved RAA codes. *IACR Cryptol. ePrint Arch.*, page 1609, 2024. 20, 23

[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. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 291–308. ACM, 2019. 24

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 8, 20, 22

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 387–416. Springer, 2020. 15, 20, 21, 24

[BCG+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633. Springer, 2022. 15, 20, 22

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press. 8, 19, 22, 24

[BCH+25] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. Encrypted matrix-vector products from secret dual codes. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025. 1

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437. Springer, 2019. 18

[BDMP98] Sergio Benedetto, Daniel Divsalar, Guido Montorsi, and Fabrizio Pollara. Analysis, design, and iterative decoding of double serially concatenated codes with interleavers. *IEEE Journal on Selected Areas in Communications*, 16(2):231–244, February 1998. 15, 22, 23

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991. 8

[Ber] Daniel J. Bernstein. The transposition principle. `https://cr.yp.to/transposition.html`. Accessed: 2025-05-06. 22

[BFKL94] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 278–291, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 6, 11

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988. 24

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient pir. *Cryptology ePrint Archive*, 2021. 7, 14

[BHW19] Elette Boyle, Justin Holmgren, and Mor Weiss. Permuted puzzles and cryptographic hardness. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 465–493, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland. 7, 14

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 662–693. Springer, 2017. 5, 6, 7, 9, 10, 14, 15

[BKSW24] Remco Bloemen, Daniel Kales, Philipp Sippl, and Roman Walch. Large-scale MPC: scaling private iris code uniqueness checks to millions of users. *IACR Cryptol. ePrint Arch.*, page 705, 2024. 4, 5, 8, 19, 33

[BMW24] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-rate PIR for databases with small records. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 1463–1477. ACM, 2024. 18

[BN25] Mark Braverman and Stephen Newman. Sublinear-overhead secure linear algebra on a dishonest server. *CoRR*, abs/2502.13060, 2025. 7, 8, 12, 20, 21

[BØ23] Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 391–422. Springer, 2023. 13, 28

[Bor56] F. Bordewijk. Interreciprocity applied to electrical networks. *Applied Scientific Research*, 6(1):1–14, 1956. 15, 22

[BR25] Martijn Brehm and Nicolas Resch. Linear time encodable binary code achieving GV bound with linear time encodable dual achieving GV bound. *arXiv preprint arXiv:2509.07639*, 2025. 15

[BW21] Keller Blackwell and Mary Wootters. A note on the permuted puzzles toy conjecture. *arXiv preprint arXiv:2108.07885*, 2021. 7, 14

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988. 24

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362, Cambridge, MA, USA, February 10–12, 2005. Springer Berlin Heidelberg, Germany. 24

[CDV21] Aidao Chen, Anindya De, and Aravindan Vijayaraghavan. Learning a mixture of two subspaces over finite fields. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, volume 132 of *Proceedings of Machine Learning Research*, pages 481–504. PMLR, 2021. 6, 7, 13

[CGKO06] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 79–88, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. 8

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50. IEEE Computer Society, 1995. 5, 7

[CGL15] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. New unconditional hardness results for dynamic and online problems. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1089–1107. IEEE Computer Society, 2015. 6

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 694–726. Springer, 2017. 5, 6, 7, 9, 10, 14, 15

[CIMR25] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. Cryptology ePrint Archive, Paper 2025/646, 2025. 5, 6, 7, 9, 10, 13, 14, 18, 19, 42, 44, 46

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland. 7

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000. 28

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2019. 18

[DI14] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 169–182. ACM, 2014. 15, 20, 22, 23

[DJM98] Daniel Divsalar, Hui Jin, and Robert J. McEliece. Coding theorems for "turbo-like" codes. In *Proc. 36th Annual Allerton Conference on Communication, Control, and Computing*, pages 201–210, Monticello, IL, USA, 1998. 15, 22, 23

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 621–630. ACM, 2009. 6, 7, 9, 13

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in*

*Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2000. 6

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2003. 24

[Döt14] Nico Döttling. *Cryptography based on the Hardness of Decoding*. PhD thesis, Karlsruhe Institute of Technology, 2014. 44

[DPC23] Alex Davidson, Gonçalo Pestana, and Sofía Celi. Frodopir: Simple, scalable, single-server private information retrieval. *Proc. Priv. Enhancing Technol.*, 2023(1):365–383, 2023. 6

[ELS+24] Kasra Edalatnejad, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, Massimo Marelli, and Carmela Troncoso. Janus: Safe biometric deduplication for humanitarian aid distribution. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 655–672. IEEE, 2024. 4, 8

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464. Springer, 2019. 18

[GIK+24] Adrià Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. Computationally secure aggregation and private information retrieval in the shuffle model. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 4122–4136. ACM, 2024. 7

[GM08] Venkatesan Guruswami and Widad Machmouchi. Explicit interleavers for a repeat accumulate accumulate (RAA) code construction. In Frank R. Kschischang and En-Hui Yang, editors, *2008 IEEE International Symposium on Information Theory, ISIT 2008, Toronto, ON, Canada, July 6-11, 2008*, pages 1968–1972. IEEE, 2008. 15, 22, 23

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, Cambridge, UK, 2001. 10

[HDCZ23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In *Proceedings of the 29th symposium on operating systems principles*, pages 396–416, 2023. 6, 8

[HHC+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 3889–3905. USENIX Association, 2023. 6

[HKL+12] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 346–365. Springer, 2012. 15, 20, 21

[HYT+24] Jiaxing He, Kang Yang, Guofeng Tang, Zhangjie Huang, Li Lin, Changzheng Wei, Ying Yan, and Wei Wang. Rhombus: Fast homomorphic matrix-vector multiplication for secure two-party inference. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 2490–2504. ACM, 2024. 8

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005. 6

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *47th Annual Symposium on Foundations of Computer Science*, pages 239–248, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press. 7

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 433–442, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 22, 23

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, 2009. 5, 10, 11, 24

[JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018: 27th USENIX Security Symposium*, pages 1651–1669, Baltimore, MD, USA, August 15–17, 2018. USENIX Association. 9

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 5

[KPRR25] Vladimir Kolesnikov, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. Stationary syndrome decoding for improved PCGs. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 284–317. Springer, 2025. 13, 24, 28

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic ram computation from ring lwe. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 595–608, New York, NY, USA, 2023. Association for Computing Machinery. 6

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. 15, 21

[LWYY24] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 149–179. Springer, 2024. 13, 16

[LXYY25] Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators for any finite field. *IACR Cryptol. ePrint Arch.*, page 169, 2025. To appear in Eurocrypt 2025. 24

[MBD+18] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Inf. Theory*, 64(5):3927–3943, 2018. 15, 21

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In Christof G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453, Davos, Switzerland, May 25–27 1988. Springer. 23

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 245–254. ACM, 1999. 5

[PZM+24] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy*, pages 4753–4771, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. 9

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC*, 1989. 24

[RRT23] Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 602–632, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. 24

[Sot16] Aikaterini Sotiraki. Authentication protocol using trapdoored matrices. Master's thesis, Massachusetts Institute of Technology, 2016. 7

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE, 2000. 5, 8

[VZ25] Vinod Vaikuntanathan and Or Zamir. Improving algorithmic efficiency using cryptography. *CoRR*, abs/2502.13065, 2025. 7, 8, 12, 20, 21

# A LPN-Based EMVP

We obtain an LPN-based EMVP by modifying the LPN-based sk-PIR from [CIMR25]. We present the protocol in detail in Fig. 3 and prove its security and correctness in what follows.

The protocol is parametrized by a failure probability $\eta > 0$ and a parameter $\tau > 0$, which roughly corresponds to the probability that a query is noiseless (before applying the error correction). In Fig. 3, we set $\tau = \sqrt{15\ln(1/\eta)/|\mathbb{F}|}$, which is large enough to allow for erasure decoding over $\mathbb{F}$ yet bounded away from 1. This choice is optimized for the concrete setting, where we aim to gain advantage in communication in general, and download cost in particular, compared to the trivial solution (where the client downloads the matrix) already with small values of $\ell$. (The requirement that $(1-\varepsilon)^\ell \geq \tau$ poses a trade-off between $\ell$ and the LPN noise rate $\varepsilon$ and therefore dimension $k$.) We provide concrete performance data of instantiations in this setting in Table 2.

To obtain Theorem 4.1, we must set $\tau = 1 - o(1)$, which allows using high-rate Reed-Solomon codes for erasure-decoding, thus achieving asymptotically near-optimal communication and computation for sufficiently large $\ell$. Specifically, given a large enough $\ell$, we tune the security parameter for the LPN parameters to be $k = \ell^{1/\gamma'}$ and $\varepsilon = k^{-\gamma} = o(1/\ell)$, obtaining $\tau = 1 - o(1)$. On the other hand, concrete LPN security under such a choice of parameters requires $\ell$ to be impractically large.

While the following lemma considers the choice of parameters specified by Fig. 3, the statement still holds for when $\tau = 1 - o(1)$ with a similar proof adjusted to the different choice of parameters.

**Lemma A.1** (EMVP from LPN: Correctness)**.** *The EMVP protocol from Fig. 3 satisfies correctness except with probability $\eta$ at any query, over the randomness of the client.*

*Proof.* By the correctness of the underlying $\mathsf{TDM}$, it holds for all $j$ that the value computed by the client in step 1.4. of the query algorithm is $\mathbf{r}'_j = \mathbf{R}\widehat{\mathbf{q}}_j$. Additionally, we may rewrite the value of $\mathbf{M}'$ computed by the server as

$$\begin{aligned}
\mathbf{M}' &= \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbb{1}_{m'_1}(j)) \cdot \widehat{\mathbf{M}}\widehat{\mathbf{q}}_j \\
&= \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbb{1}_{m'_1}(j)) \cdot \Big( (I_{m_2} \otimes \mathbf{G}) \cdot \mathbf{M} \cdot [I_\ell \,|\, -\mathbf{S}^\intercal] + \mathbf{R} \Big) \widehat{\mathbf{q}}_j \\
&= \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbf{G}_j) \cdot \mathbf{M} \cdot [I_\ell \,|\, -\mathbf{S}^\intercal] \widehat{\mathbf{q}}_j + \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbb{1}_{m'_1}(j)) \mathbf{R}\widehat{\mathbf{q}}_j \\
&= \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbf{G}_j) \cdot \mathbf{M} \cdot (\mathbf{q} + \mathbf{e}_j) + \sum_{j=1}^{m'_1} (I_{m_2} \otimes \mathbb{1}_{m'_1}(j)) \mathbf{r}'_j.
\end{aligned}$$

where $\mathbf{G}_j$ is the $m'_1 \times m_1$ matrix that is equal to $\mathbf{G}$ at the $j^{th}$ row and is zeros anywhere else. Hence,

**Parameters:**

- LPN parameters $k(\lambda)$ and $\varepsilon(\lambda) = o(1)$.
- Failure probability $\eta > 0$.
- Field oracle $\mathbb{F}$ such that $|\mathbb{F}| > 60\ln(1/\eta)$. We denote $\tau = \sqrt{15\ln(1/\eta)/|\mathbb{F}|} < 1/2$.
- A Reed-Solomon code $\mathsf{RS}(K, N)$ over $\mathbb{F}$ for any dimension $K$ and length $N$ such that $K < N \leq |\mathbb{F}|$, with an efficient erasure-decoding algorithm $\mathsf{dec}$.
- Trapdoored matrix $\mathsf{TDM}$ with trapdoor size $\kappa(\lambda, m, n)$.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

**The Protocol:**

- $\mathsf{G}(1^\lambda, 1^m, 1^\ell)$:
  1. Let $\lambda_0 = \min\{\lambda' \ : \ \lambda' \geq \lambda,\ (1-\varepsilon(\lambda'))^\ell \geq \tau\}$ and let $k = k(\lambda_0)$ and $\varepsilon = \varepsilon(\lambda_0)$.
  2. Let $\ell_0 = \lfloor \ln(\tau)/\ln(1-\varepsilon) \rfloor$ and $m_0 = m_1 m_2$, where $m_1 = \lceil 5\ln(1/\eta)/(1-\varepsilon)^{\ell} \rceil$ and $m_2 = \lceil m/m_1 \rceil$.
     // In the following, we assume w.l.o.g. that $\ell = \ell_0$ and $m = m_0$. Otherwise, the input matrix is padded with the necessary amount of zeros to match the new dimensions (it always holds that $\ell_0 \geq \ell$ and $m_0 \geq m$).
  3. Return $\mathsf{sk} = (\lambda_0, k, \varepsilon, \ell, m_1, m_2, \mathbf{k})$ where $\mathbf{k} \leftarrow \{0,1\}^\lambda$ is a PRF key.
- $\mathsf{E}(\mathsf{sk}, \mathbf{M})$: Use $\mathsf{PRF}(\mathbf{k}, \cdot)$ to determine the randomness in the following.
  1. Let $n = k + \ell$, $m_1' = \lceil 3m_1/(1-\varepsilon)^\ell \rceil$ and $m' = m_1' m_2$.
  2. Let $\mathbf{G} \in \mathbb{F}^{m_1' \times m_1}$ be a generator matrix for $\mathsf{RS}(m_1, m_1')$.
  3. Sample $\mathbf{S} \leftarrow \mathbb{F}^{k \times \ell}$.
  4. Sample $\mathsf{td} \leftarrow \mathbb{F}^{\kappa(\lambda_0, m', n)}$ and $(\mathbf{R}, \mathsf{EC}) \leftarrow \mathsf{TDM}(1^\lambda, 1^{m'}, 1^n, \mathsf{td})$.
  5. Return $\widehat{\mathbf{M}} = (I_{m_2} \otimes \mathbf{G}) \cdot \mathbf{M} \cdot [I_\ell \,|\, -\mathbf{S}^\intercal] + \mathbf{R} \in \mathbb{F}^{m' \times n}$.
     // Encode the rows of $\mathbf{M}$ using $[I_\ell \,|\, -\mathbf{S}^\intercal]$ and every $m_1$-size block in its columns using $\mathbf{G}$. Then, mask by $\mathbf{R}$.
- $\mathsf{Q}(\mathsf{sk}, \mathbf{q})$:
  1. For $j = 1, \ldots, m_1'$,
     1.1. Sample $\mathbf{r}_j \leftarrow \mathbb{F}^k$ and $\mathbf{e}_j \leftarrow \mathsf{Ber}^\ell(\varepsilon)$.
     1.2. Let $\widehat{\mathbf{q}}_j = (\mathbf{q} \,|\, 0^k) + (\mathbf{r}_j \mathbf{S} + \mathbf{e}_j \,|\, \mathbf{r}_j) \in \mathbb{F}^n$.
     1.3. Let $b_j$ be the predicate for $\mathbf{e}_j = 0^\ell$.
     1.4. Let $\mathbf{r}_j' = \mathsf{EC}(\widehat{\mathbf{q}}_j)$.
  2. Return $\widehat{\mathbf{q}} = (\widehat{\mathbf{q}}_1, \ldots, \widehat{\mathbf{q}}_{m_1'})$ and $\mathbf{q}' = ((\mathbf{r}_1', \ldots, \mathbf{r}_{m_1'}'), (b_1, \ldots, b_{m_1'}))$.
- $\mathsf{A}(\widehat{\mathbf{M}}, \widehat{\mathbf{q}})$:
  1. Parse $\widehat{\mathbf{q}}$ as $(\widehat{\mathbf{q}}_1, \ldots, \widehat{\mathbf{q}}_{m_1'})$.
  2. Return $\mathbf{M}' = \sum_{j=1}^{m_1'} (I_{m_2} \otimes \mathbb{1}_{m_1'}(j)) \cdot \widehat{\mathbf{M}} \widehat{\mathbf{q}}_j \in \mathbb{F}^{m'}$, where $\mathbb{1}_{m_1'}(j)$ is the $m_1' \times m_1'$ matrix that has 1 at coordinate $(j, j)$ and 0 everywhere else.
     // Multiply the $j^{th}$ row in every block of $m_1$ rows in $\widehat{\mathbf{M}}$ by $\widehat{\mathbf{q}}_j$.
- $\mathsf{D}(\mathsf{sk}, \mathbf{M}', \mathbf{q}')$:
  1. Parse $\mathbf{q}'$ as $((\mathbf{r}_1', \ldots, \mathbf{r}_{m_1'}'), (b_1, \ldots, b_{m_1'}))$.
  2. Let $E = \{j \mid b_j = 0\}$. If $|E| > m_1' - m_1$, output $\perp$.
  3. Let $\mathbf{y} = \mathbf{M}' - \sum_{j=1}^{m_1'} (I_{m_2} \otimes \mathbb{1}_{m_1'}(j)) \cdot \mathbf{r}_j'$.
     // Use the corresponding value in $\mathbf{r}_j$ to unmask the $j^{th}$ location in every block of $m_1$ locations in $\mathbf{M}'$.
  4. Parse $\mathbf{y} = (\mathbf{y}_1 \,/\!/ \cdots /\!/\, \mathbf{y}_{m_2})$, where $\mathbf{y}_i \in \mathbb{F}^{m_1'}$, and let $\mathbf{y}_i'$ be equal to $\mathbf{y}_i$ except it contains $\perp$ at any $j \in E$.
  5. Output $\mathbf{a} = (\mathsf{dec}(\mathbf{y}_1'), \ldots, \mathsf{dec}(\mathbf{y}_{m_2}')) \in \mathbb{F}^m$.

Figure 3: Conservative EMVP Protocol from LPN.

the value computed at step 3. of the decoding is

$$\mathbf{y} = \sum_{j=1}^{m_1'} (I_{m_2} \otimes \mathbf{G}_j) \cdot \mathbf{M} \cdot (\mathbf{q} + \mathbf{e}_j).$$

Letting $\mathbf{M} = [\mathbf{M}_1 \,//\, \cdots \,//\, \mathbf{M}_{m_2}]$ for $\mathbf{M}_i \in \mathbb{F}^{m_1 \times \ell}$ and $\mathbf{y} = (\mathbf{y}_1 \,//\, \cdots \,//\, \mathbf{y}_{m_2})$, it holds for all $i \in [m_2]$ that

$$\mathbf{y}_i = \sum_{j=1}^{m_1'} \mathbf{G}_j \cdot \mathbf{M}_i \cdot (\mathbf{q} + \mathbf{e}_j) = \mathbf{G}(\mathbf{M}_i\mathbf{q}) + \underbrace{\sum_{j=1}^{m_1'} \mathbf{G}_j \cdot \mathbf{M}_i \cdot \mathbf{e}_j}_{\mathbf{z}_i}.$$

Note that $\mathbf{z}_i \in \mathbb{F}^{m_1'}$ is the column vector with $\mathbf{G}\mathbf{M}_i\mathbf{e}_j$ at the $j^{th}$ location.

Now, let us assume that the client does not output $\perp$ in the decoding, namely that the set of error locations $E$ defined in step 2. of the decoding has size less than $m_1' - m_1$. We argue that, conditioned on this event, the client indeed decodes the correct value $\mathbf{a} = \mathbf{M}\mathbf{q}$. For any $j \notin E$, it holds that $\mathbf{e}_j = 0^\ell$ and therefore in such a case, $\mathbf{z}_i$ contains at least $m_1$ zeros and $\mathbf{y}_i'$ is equal to $\mathbf{y}_i$ in all of the corresponding locations. Since this falls within the decoding capacity of a Reed-Solomon code with dimension $m_1$, it holds that $\mathsf{dec}(\mathbf{y}_i') = \mathbf{M}_i\mathbf{q}$ and hence $\mathbf{a} = \mathbf{M}\mathbf{q}$.

It remains to bound the probability that the client outputs $\perp$ by $\eta$. Note that $|E|$ is a sum of $m_1'$ i.i.d. Bernoullis, each with probability $\Pr[\mathbf{e}_i \neq 0^\ell] = 1 - (1-\varepsilon)^\ell$. It holds then, that $\tilde{E} := \mathbb{E}[|E|] = (1-(1-\varepsilon)^\ell)m_1' \leq 3m_1(1/(1-\varepsilon)^\ell - 1) + 1$ and $m_1' - m_1 \geq (3/(1-\varepsilon)^\ell - 1)m_1 \geq (1 + \frac{2}{3}(1-\varepsilon)^\ell)\tilde{E}$. We conclude the proof using Chernoff inequality:

$$\begin{aligned}
\Pr[|E| \geq m_1' - m_1] &\leq \Pr[|E| \geq (1 + \tfrac{2}{3}(1-\varepsilon)^\ell)\tilde{E}] \\
&\leq \exp\Big( - (\tfrac{2}{3}(1-\varepsilon)^\ell)^2 \tilde{E}/3 \Big) \\
&\leq \exp\Big( - \tfrac{4}{9}(1-\varepsilon)^{2\ell} m_1 (1/(1-\varepsilon)^\ell - 1) \Big) \\
&\leq \exp\Big( - \tfrac{4}{9}((1-\varepsilon)^\ell - (1-\varepsilon)^{2\ell}) m_1 \Big) \leq \eta,
\end{aligned}$$

where the last inequality holds since $(1-\varepsilon)^\ell \leq \tau + o(1) \leq 11/20$ by the assumption on $\tau$ for sufficiently large $\lambda_0$. □

**Lemma A.2** (EMVP from LPN: Security)**.** *The EMVP protocol from Fig. 3 is secure under* $(k, \varepsilon)$-$\mathsf{LPN}$ *(Definition 2.2).*

Our proof of security goes through a variant of LPN, where samples are given w.r.t. polynomially many i.i.d. secrets, which is known to be polynomially equivalent to LPN (see, e.g. [Döt14, CIMR25]).

**Lemma A.3** (From LPN to Matrix-LPN)**.** *The* $(m, k, \varepsilon)$-$\mathsf{LPN}$ *assumption implies that, for any polynomial* $n := n(\lambda)$*, the following two distributions are computationally indistinguishable*

$$\big( \mathbf{r}_i, \mathbf{r}_i^\intercal \mathbf{S} + \mathbf{e}_i \big)_{i=1,\ldots,m} \quad \overset{c}{\approx} \quad \big( \mathbf{r}_i, \mathbf{u}_i \big)_{i=1,\ldots,m},$$

*where* $\mathbf{S} \leftarrow \mathbb{F}^{k \times n}$ *and, for any* $i$, $\mathbf{r}_i \leftarrow \mathbb{F}^k$ *and* $\mathbf{u}_i \leftarrow \mathbb{F}^n$ *are uniform, and* $\mathbf{e}_i \leftarrow \mathsf{Ber}^n(\varepsilon)$.

We now sketch the proof of Lemma A.2.

*Proof.* The proof follows similar lines of those in the proof of Lemma 4.2. Once again, under the security of the PRF and the $\mathsf{TDM}$, we switch to a hybrid experiment where the LPN secret $\mathbf{S}$ and the mask $\mathbf{R}$ are both uniformly random. Since the database encoding is masked by $\mathbf{R}$, which is independent in the queries generated by the client, the adversary's view can be simulated only given the queries, which consist of polynomially many samples of the form $\widehat{\mathbf{q}} = (\widehat{\mathbf{q}}_1, \ldots, \widehat{\mathbf{q}}_{m'_1})$, where $\widehat{\mathbf{q}}_j = (\mathbf{q} \,|\, 0^k) + (\mathbf{r}_j\mathbf{S} + \mathbf{e}_j \,|\, \mathbf{r}_j)$. These queries are pseudorandom under LPN due to Lemma A.3. □

Table 2: Communication and storage in LPN-Based EMVP with non-trivial download cost. We instantiate the protocol from Fig. 3 with field of size $|\mathbb{F}| = 2^{10}$, security parameter $\lambda = 128$ and failure probability $\eta = 2^{-40}$. LPN parameters are chosen such that guessing $k$ noiseless coordinates succeeds with probability less than $2^{-\lambda}$. All quantities are in field elements.

| $\log m$ | $\log \lvert\mathbf{M}\rvert$ | log download | log comm. | log storage |
|---|---|---|---|---|
| $\ell = 1000$ | | | | |
| 10 | 19.96 | 12.34 | 27.63 | 29.95 |
| 20 | 29.96 | 22.23 | 27.67 | 39.85 |
| $\ell = 10000$ | | | | |
| 10 | 23.28 | 12.34 | 30.95 | 33.28 |
| 20 | 33.28 | 22.23 | 30.95 | 43.17 |
| $\ell = 20000$ | | | | |
| 10 | 24.29 | 12.34 | 31.95 | 34.27 |
| 20 | 34.29 | 22.23 | 31.95 | 44.17 |

# B Proof of Lemma 5.1

*Proof.* We define $m$ hybrid distributions as follows. For $i = 0, \ldots, m$, the distribution $D^i$ consists of $\big(\mathbf{H}, (\mathbf{u}_1, \ldots, \mathbf{u}_i, \mathbf{H}\mathbf{e}_{i+1}, \ldots, \mathbf{H}\mathbf{e}_m)\big)$ where $\mathbf{H} \leftarrow \mathcal{H}(1^k, 1^n)$ and, for all $i$, $\mathbf{e}_i \leftarrow \mathsf{Ber}^n_{\mathbb{F}}(\varepsilon)$ and $\mathbf{u}_i \leftarrow \mathbb{F}^k$. Note that $D^0 \equiv (\mathbf{H}, \mathbf{HE})$ and $D^m \equiv (\mathbf{H}, \mathbf{U})$. Therefore, by triangle inequality, if an efficient nonuniform adversary $\mathcal{A}$ is able to distinguish between $(\mathbf{H}, \mathbf{HE})$ and $(\mathbf{H}, \mathbf{U})$, then there is such an adversary which is to distinguish between $D^i$ and $D^{i-1}$ for some $i := i(\lambda)$. We use such an adversary to break dual-LPN via the following reduction. Given $(\mathbf{H}, \mathbf{b})$, the reduction samples $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1} \leftarrow \mathbb{F}^k$ uniformly at random and $\mathbf{e}_{i+1}, \ldots, \mathbf{e}_m \leftarrow \mathsf{Ber}^n_{\mathbb{F}}(\varepsilon)$. The reduction runs $\mathcal{A}$ with the input $(\mathbf{H}, \mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, \mathbf{b}, \mathbf{H}\mathbf{e}_{i+1}, \ldots, \mathbf{H}\mathbf{e}_m)$. When $(\mathbf{H}, \mathbf{b})$ is a uniform sample, then the corresponding sample given to the adversary is from the distribution $D^i$ and, if it is a dual-LPN sample, namely $\mathbf{b} = \mathbf{He}$ for a sparse $\mathbf{e}$, then the sample given to $\mathcal{A}$ follows the distribution $D^{i-1}$. □

# C Improving the Circuit Complexity of PIR via EMVP over $\mathbb{F}_4$

When $|\mathbb{F}| \geq 3$, the $\mathsf{1D\text{-}SLSN}$ assumption allows the online computational complexity of the server to be arbitrarily close to plaintext computation: If $\mathbf{M}$ has $N$ entries in total, then the online

computation of the server consists of computing $s$ matrix-vector products $\widehat{\mathbf{M}}_j\widehat{\mathbf{q}}_j$, where the $\mathbf{M}_j$ have a total of $(1+o(1))N$ entries. This comes with near-optimal server storage of $(1+o(1))N$.

However, in the case $\mathbb{F}=\mathbb{F}_2$, which suffices for secret-key PIR, the 1D-SLSN assumption is not meaningfully defined. One option, used to minimize the server's Boolean circuit complexity in [CIMR25], is to rely on the 2-dimensional variant $(k,n,2,s)$-SLSN. However, this doubles the server computational overhead: instead of multiplying each $\widehat{\mathbf{M}}_j$ by a single $\widehat{\mathbf{q}}_j$, now it is multiplied by two such vectors. This implies secret-key PIR in which the Boolean circuit complexity of the server on an $N$-bit database is $(4+o(1))N$, with $(1+o(1))N$ server storage. Another option is to reduce PIR to EMVP over $\mathbb{F}_3$ or $\mathbb{F}_4$. However, a naive implementation of this approach will further increase the Boolean circuit complexity of the server due to the cost of implementing field multiplications.

As we show below, using 1D-SLSN-based EMVP over $\mathbb{F}_4$, we can improve the server circuit complexity to $(3+o(1))N$ at the cost of 50% extra storage. Concretely, in this protocol the server stores $(1.5+o(1))N$ bits and performs $\widehat{\mathbf{M}}_j\widehat{\mathbf{q}}_j$ multiplications over $\mathbb{F}_2$ in which the total number of $\mathbb{F}_2$ entries in all $\widehat{\mathbf{M}}_j$ is $(1.5+o(1))N$. Alternatively, by increasing the server circuit complexity to $(3.5+o(1))N$ (still slightly improving over [CIMR25]), we can have the server store only $(1+o(1))N$ bits, eliminating the storage overhead. We sketch the details below.

Recall that in answering query in the online phase of the 1D-SLSN-based protocol, the server must compute inner products over $\mathbb{F}$ of total size $n$ per matrix row. This requires $2nm$ field operations in total. We show how to reduce this online complexity for $\mathbb{F}=\mathbb{F}_4$ by further preprocessing the encoded matrix. When $\mathbb{F}=\mathbb{F}_4$, every field element $x\in\mathbb{F}_4$ in the encoding may be written as $x=x_1+z\cdot x_2$, where $x_1,x_2\in\mathbb{F}_2$ and $z$ is a root of the irreducible polynomial $x^2+x+1$. Since $z^2=z+1$, it holds for any $x,y\in\mathbb{F}_4$ represented by $x_1,x_2\in\mathbb{F}_2$ and $y_1,y_2\in\mathbb{F}_2$, that

$$xy=x_1y_1+z^2x_2y_2+z(x_1y_2+x_2y_1)=(x_1y_1+x_2y_2)+z(x_1y_2+x_2y_1+x_2y_2).$$

This observation allows for the following optimization: Besides storing the two bits $x_1,x_2\in\mathbb{F}_2$ corresponding to any element in the encoding $\widehat{\mathbf{M}}$, we let the server additionally store $\widehat{x}=x_1+x_2$. For any $y_1,y_2\in\mathbb{F}_2$ corresponding to any element in the encoded query vector $\widehat{\mathbf{q}}$ sent by the client, we let the client additionally send $\widehat{y}=y_1+y_2$. To compute the product of an element $(x_1,x_2)$ in $\widehat{\mathbf{M}}$ with an element $(y_1,y_2)$ in $\widehat{\mathbf{q}}$, the server computes the multiplications $w_1=x_1y_1$, $w_2=x_2y_2$ and $\widehat{w}=\widehat{x}\widehat{y}$, which give $xy=(w_1+w_2)+z(\widehat{w}+w_1)$ by the above. Hence, by extending its storage by a factor of 1.5 – storing $3n$ bits per row instead of $2n$ – the server can compute its answer by computing inner products of total size $3n$ over $\mathbb{F}_2$, instead of inner products of total size $n$ over $\mathbb{F}_4$. This saves 25% in the number of $\mathbb{F}_2$-operations performed by the server compared to the $(k,n,2,s)$-SLSN based solution from [CIMR25], at the expense of 50% extra storage. Alternatively, if we compute $\widehat{x}$ on the fly instead of precomputing it, we eliminate the extra storage using $3.5n$ $\mathbb{F}_2$ operations per row, still slightly improving over the $4n$ operations in [CIMR25].

# D Private Matrix-Vector Product with a Server-Owned Matrix

In this section we consider an alternative EMVP setting in which the matrix $\mathbf{M}$ is owned by the server and the only goal is to hide the queries $\mathbf{q}_i$ from the server. In this setting, we aim to support multiple clients without making non-collusion assumptions by allowing each client to obtain a short and reusable "hint" about $\mathbf{M}$.

Concretely, a matrix $\mathbf{M}\in\mathbb{F}^{m\times\ell}$ is held by a server in the clear. The offline phase consists of a two-message protocol between the server and any client, after which the client obtains a small hint that depends on $\mathbf{M}$ and its private randomness. After the offline phase is complete, the client,

given inputs $\mathbf{q}_i \in \mathbb{F}^\ell$, can use the hint to obtain the product $\mathbf{M}\mathbf{q}_i$ while keeping $\mathbf{q}_i$ private from the server (as long as the hint is kept secret).

We describe how to convert the 1D-SLSN-based protocol from Fig. 1 to the above setting. An analogous modification can be applied to the LPN-based protocol from Fig. 3. Recall that the matrix encoding in the protocol is computed as $\widehat{\mathbf{M}} = \mathbf{M}\mathbf{D} + \mathbf{R}$, where $\mathbf{D} = [I_\ell \mid \mathbf{D}']$ is a systematic generator matrix of a random code $D$ and $\mathbf{R}$ is a pseudorandom trapdoor matrix used to hide $\mathbf{M}\mathbf{D}$ from the server. Since $\mathbf{D}$ is systematic, we may write $\mathbf{M}\mathbf{D} = [\mathbf{M} \mid \mathbf{D}'\mathbf{M}] = [\mathbf{M} \mid \mathbf{M}^*]$. At a high level, the matrix $\mathbf{M}^*$ will be used as the client's hint. The client will delegate the "systematic part" of the encoded query to the server, while computing the inner product over the non-systematic part locally using $\mathbf{M}^*$. In such a setting, since we do not need to hide $\mathbf{M}$ and since $\mathbf{M}^*$ is anyway stored locally, there is no need for the mask $\mathbf{R}$ and therefore for the TDM component.

In the offline phase of the protocol, the client sends a linearly homomorphic encryption of a uniformly random matrix $\mathbf{D}' \leftarrow \mathbb{F}^{\ell \times (n-\ell)}$, which defines a random code $D$ as above. (Using the structured codes discussed in Section 3.1, it suffices to encrypt the $O(n)$ field elements defining this code.) The server, in turn, computes an encryption of $\mathbf{M}^* = \mathbf{M}\mathbf{D}'$ by homomorphic evaluation and sends it to the client, who decrypts it and stores it locally. Analogously to Fig. 1, we split $\widehat{\mathbf{M}} = [\mathbf{M} \mid \mathbf{M}^*]$ to $s$ blocks in total and denote $\mathbf{M} = [\widehat{\mathbf{M}}_1 \mid \cdots \mid \widehat{\mathbf{M}}_{s\ell/n}]$ and $\mathbf{M}^* = [\widehat{\mathbf{M}}_{s\ell/n+1} \mid \cdots \mid \widehat{\mathbf{M}}_s]$ (we assume w.l.o.g. that $s\ell/n$ is an integer).

In the online query phase, the client computes the encoded query vector $\widehat{\mathbf{q}}$ as described in Fig. 1, using a uniformly random codeword $\mathbf{c}$ sampled from the code $C = D^\perp$. Let $\widehat{\mathbf{q}}^1 = (\widehat{\mathbf{q}}_1 \mid \cdots \mid \widehat{\mathbf{q}}_{\ell s/n}) \in \mathbb{F}^\ell$ denote the first $\ell$ field elements in $\widehat{\mathbf{q}}$ and $\widehat{\mathbf{q}}^2 = (\widehat{\mathbf{q}}_{\ell s/n+1} \mid \cdots \mid \widehat{\mathbf{q}}_s) \in \mathbb{F}^{n-\ell}$ denote the last $n - \ell$ ones. The client sends $\widehat{\mathbf{q}}^1$ to the server, who responds by $\widehat{\mathbf{M}}_i\widehat{\mathbf{q}}_i$, for $i = 1, \ldots, \ell s/n$. The client then locally computes $\widehat{\mathbf{M}}_i\widehat{\mathbf{q}}_i$ for $i = \ell s/n + 1, \ldots, s$. As a result, the client obtains all values required to decode $\mathbf{M}\mathbf{q}$ as described in the original protocol.[6]

# E Implementation and Benchmarks

## E.1 Implementation

We implement two variants of the EMVP protocols: one based on the 1D-SLSN assumption (see Section 4.1), and one cased on its ring variant (see Section 3.1). The implementation is written in Go with performance-critical components implemented in C++ backends to support field arithmetic and fast Fourier transform (FFT)-based polynomial operations.

For masking the database, we used the trapdoored matrix constructed from random quasi-cyclic matrices composed with permutations (see Section 5.2).

## E.2 Benchmarks

All experiments are run on a MacBook Pro with an Apple M3 chip and 16 GB RAM, running macOS 14.4.1. In the first set of experiments, we fix the total number of entries in the matrix (before encoding) to $m \times \ell = 2^{20}$. All implementations are single-threaded. Each timing result is obtained using Go's native benchmark framework, which runs the code repeatedly until statistically stable measurements are achieved. In the second set of experiments, we fix $(\ell, k, b)$ and vary $m$, showing that increases in $m$ affect only the server answer and client decode phases, each scaling

[6]Note that the inner product over the non-systematic part – i.e. of $\mathbf{M}^*$ and the last $n - \ell$ elements in $\mathbf{c}$ – can be performed straightforwardly, without going through the encoding $\widehat{\mathbf{q}}^2$, since it is computed locally by the client. We choose the above presentation due to the simple analogy with the original protocol in Fig. 1.

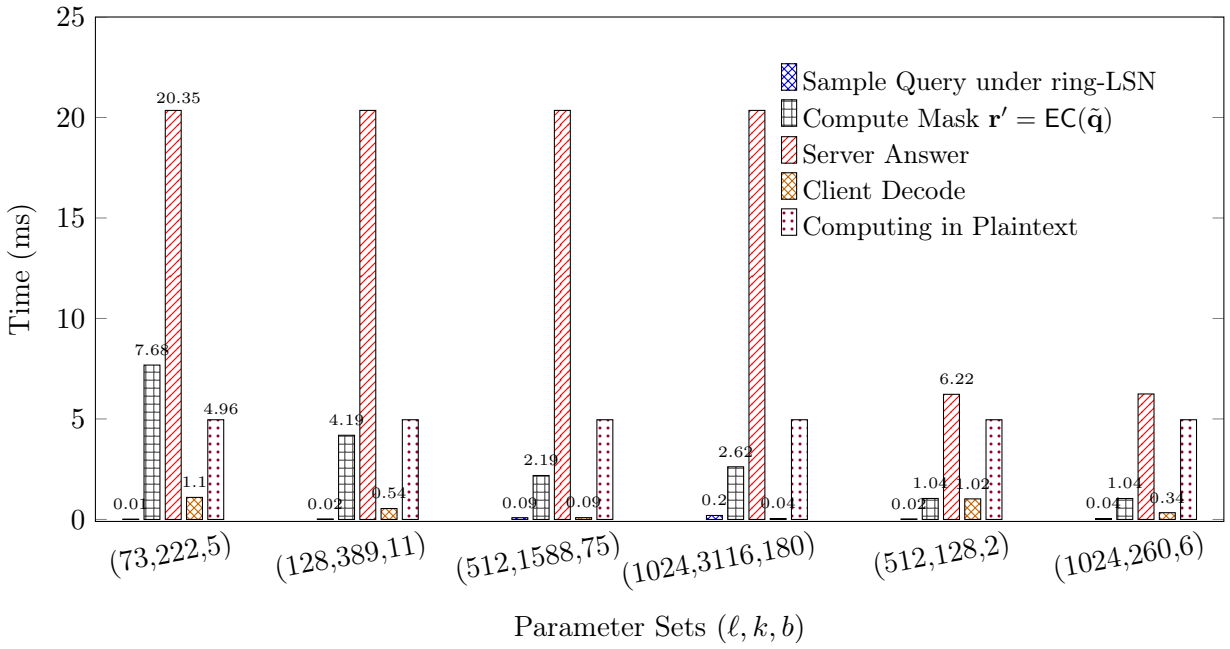


Figure 4: Benchmark for 1D-SLSN construction (Fig. 1) with parameters $(\ell, k, b)$ in Table 1. Each experiment operates over an $m \times \ell$ matrix with $\approx 2^{20}$ entries over a 32-bits prime, where the row dimension $m$ is chosen accordingly. The bars show a breakdown of the total online time, including query generation for the ring variant, trapdoored matrix circuit evaluation for unmasking, server response, and client decoding. The plaintext computation simulates performing the matrix-vector product directly on the unencrypted matrix as a baseline for performance comparison.

linearly in $m$ as expected from the underlying matrix-vector product. These experiments empirically confirm the theoretical performance characteristics of our constructions, such as quasi-linear scaling of query generation under the ring variant of 1D-SLSN and improved client-side efficiency by shifting computational load to the server.

The results from the first set of experiments are shown in Fig. 4, with parameters from Table 1. The results from the second set of experiments, running with $(\ell, k, b) = (10000, 2600, 140)$, with row dimension $m$ from $2^7$ to $2^{14}$ are shown in Fig. 5.

Our code is available at:
`https://github.com/SecretKeyCrypto/Encrypted-Matrix-Vector-Products`.

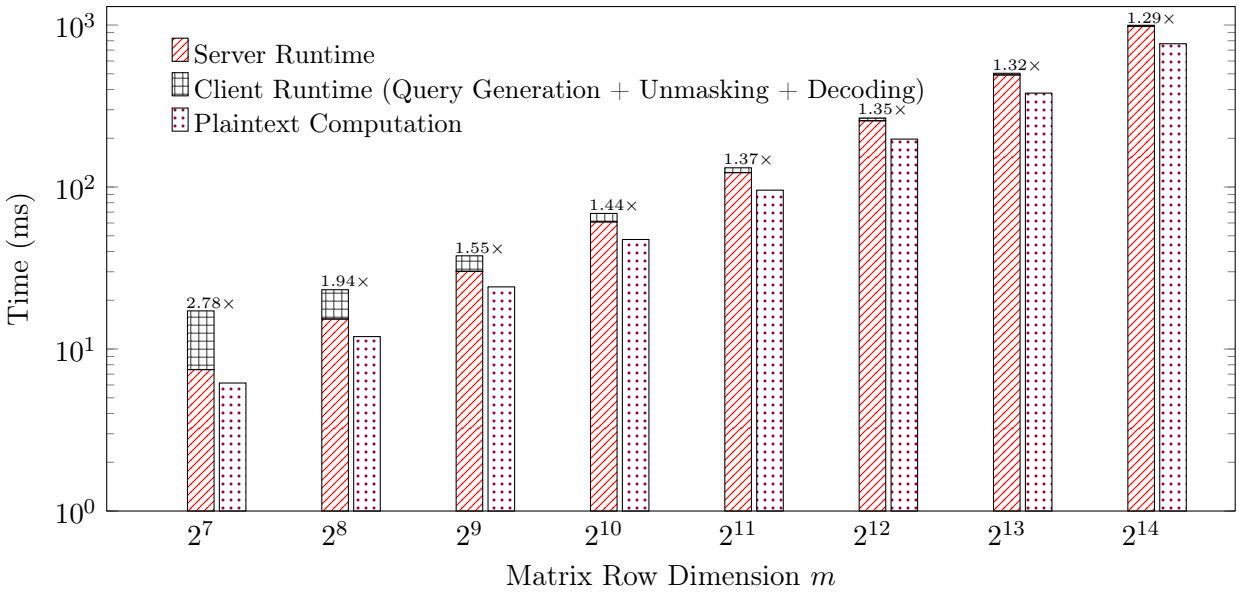


Figure 5: Performance breakdown of total online runtime with varying matrix row dimension $m$ (log scale). The experiments are conducted with parameters $(\ell, k, b) = (10000, 2600, 140)$, where each instance operates over an $m \times \ell$ matrix over a 32-bits prime. The stacked bar represents the total online time, consisting of server answer time and aggregated client time (which combines query generation, TDM unmasking, and decoding). The computing in plaintext bar indicates the time required to compute the matrix-vector product in plaintext, serving as a baseline. The number above each bar group indicates the ratio between the online runtime (server + client) and the plaintext execution time. As $m$ increases, the online time becomes dominated by the server answer component, and the ratio gets closer to the theoretical server overhead $f = \frac{\ell+k}{\ell} = 1.26$.