Encrypted Matrix-Vector Products from Secret Dual Codes

Fabrice Benhamouda
fabrice.benhamouda@gmail.com
Amazon Web Services
New York, NY, USA

Caicai Chen
caicai.chen@unibocconi.it
Bocconi University
Milan, Italy

Shai Halevi
shai.halevi@gmail.com
Amazon Web Services
New York, NY, USA

Yuval Ishai
yuval.ishai@gmail.com
Technion and Amazon Web Services
New York, NY, USA

Hugo Krawczyk
hugokraw@gmail.com
Amazon Web Services
New York, NY, USA

Tamer Mour
tamer.mour@gmail.com
Bocconi University
Milan, Italy

Tal Rabin
rabintal@amazon.com
Amazon Web Services
New York, NY, USA

Alon Rosen
alon.rosen@unibocconi.it
Bocconi University
Milan, Italy

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 ℓ (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.

CCS Concepts

• **Security and privacy → Cryptography**.

Keywords

Secure Computation

ACM Reference Format:
Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. 2025. Encrypted Matrix-Vector Products from Secret Dual Codes. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765194

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 [7, 36]. 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 [7] 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 [36]. 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 [7] 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 [7]. Our protocols build on recent techniques for private information retrieval (PIR) in the secret-key setting from [16, 22, 24], 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 $\mathbb{F}$ [1, 48, 56]. 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 [26, 50]. 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.

[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, sending encrypted shares of a query to one of the parties could be more efficient than encrypting the query with HE.

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 ℓm, 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 [7], 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.

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 [57], 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 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 the full version [4].

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 [30, 42, 43], 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 the full version [4].

1.2 The Underlying Assumptions

Our protocols rely on different flavors of "learning subspace with noise" (LSN) assumptions that were first introduced by Dodis et al. [33] in the context of leakage-resilient cryptography and revisited by Chen et al. [24] 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 [16, 22]. 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 [8], 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 [23, 24, 33], there are relations between LSN and LPN, where some LSN variants are implied by LPN, and most imply LPN.

Unlike standard homomorphic encryption or PIR in the plain model [31, 45], 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 [33], 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 6 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 [24]. A key difference is that the PIR server's computational cost (in the RAM model) can be dramatically improved using a lattice-based approach [51]. In contrast, our LPN-based solution for EMVP seems[2] close to optimal even under this metric.

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 [33]. 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 [23] 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 [16, 22]. The latter have already withstood some analysis [6, 14, 15]. 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 [37, 46]. 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 [25].

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 [16, 22, 24], and in particular the split-LSN assumption recently put forward in [24]. 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 [26], our optimized EMVP protocols and concrete analysis of LSN-type assumptions are directly relevant also to this use case of EMVP.

In the full version, we apply EMVP over $\mathbb{F}_4$ to improve the Boolean circuit complexity of secret-key PIR. Concretely, the most efficient construction from [24], 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 [18, 59] 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 [18] 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 [18] inherently requires the

[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 [27].

[3]A similar idea appears in an earlier work by Sotiraki [58], where a weaker security notion is considered.

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 [18] 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 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 [18, 59] and that are easier to implement in a distributed client setting. In particular, we give the first *linear-time* TDM candidates.

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) [29, 57]. 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 [7, 36]) or provide efficient semantic search functionality (as in [42]) 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 (as we show in the full version [4]). 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 [41] and references therein) are significantly slower than our EMVP protocol and require much wider matrices to attain their maximal level of efficiency.

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 [16, 22, 24]: 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 $\mathbf{M}\mathbf{q}_i$ 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 [24, 33] 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 [33] to Ran 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 [24], using LSN with a special noise distribution. We further attempt to optimize the concrete efficiency of this construction, and provide some data points in the full version [4].

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 use boldface letters for matrices and vectors and capital letters for linear spaces. We use $(a \mid b)$ and $[A \mid B]$ for horizontal concatenation of vectors and matrices and $(a \mathbin{/\!/} b)$ and $[A \mathbin{/\!/} B]$ for vertical concatenation.

Probability. We use $x \leftarrow X$ to denote a uniformly random choice of x from a set X. We denote by $\mathrm{Ber}(\mu)$ a Bernoulli random variable which takes the value 1 with probability μ and 0 with probability $1 - \mu$. More generally, for a finite field $\mathbb{F}$, the Bernoulli variable $\mathrm{Ber}_{\mathbb{F}}(\mu)$ 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) [39].

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 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 [1, 48]. In fact, most of our protocols can be cast in the stricter black-box model from [1] 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 [16, 22, 24], 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 λ and matrix dimensions m, ℓ, and returns a* secret key sk*. We assume that* sk *contains $\mathbb{F}, \lambda, m, \ell$. In the field-agnostic case, $\mathbb{F}$ is not given as input and is not part of* sk.
- $\mathsf{E}(\mathsf{sk}, \mathbf{M})$*: The* matrix encryption algorithm *takes a secret key* 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* 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* 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 δ 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 Trapdoored Matrices

To minimize the computational overhead of the EMVP client, we will rely on variants of the recent notion of *trapdoored matrices* from [18, 59]. 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 $\mathbf{v} \mapsto \mathbf{Rv}$.

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 $\mathbf{Mq}_i$, it uses P' to query $\mathbf{M}'\mathbf{q}_i$, from which $\mathbf{Mq}_i$ can be computed by subtracting $\mathbf{Rq}_i$. 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 $\mathbf{v} \mapsto \mathbf{Rv}$ as having a near-linear size evaluation circuit EC that may depend on the trapdoor. However, for the distributed-client variant of our EMVP protocols considered in the full version [4], it is helpful to have a near-linear size *public* EC that computes $\mathbf{Rv}$ given $\mathbf{v}$ together with a random trapdoor 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 2.2 (Trapdoored Matrix). *A* trapdoored matrix *with trapdoor size* (κ_1, κ_2) *is a polynomial time algorithm* TDM *with the following syntax:* $\mathsf{TDM}(1^\lambda, 1^m, 1^n, \mathsf{td})$ *takes* λ, 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* 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* EC *is given* 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 EC to be of nearly optimal size, namely close to linear in $m + n$.

The recent works of Braverman and Newman [18] and Vaikuntanathan and Zamir [59] present recursive constructions of TDM in which 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 [24, 33] 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 [24, 33]). *The (basic)* learning subspace with noise *assumption* (k, n, μ)-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 $\mathbf{c}_i = \mathbf{a}_i^\top \mathbf{C}$ *for* $\mathbf{a}_i \leftarrow \mathbb{F}^k$*,* $\mathbf{e}_i$ *is uniformly random in* $\mathbb{F}^n$ *with probability* μ *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, μ are all functions of the security parameter λ.

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 [23, 24, 33]. 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 μ is equivalent to the LPN assumption with noise rate $\mu/2$ [23, 24, 33].) 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 [24]. 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 6.1).

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 [2, 21, 49, 52] 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 6.1.

In the most efficient variants of our EMVP protocols, the space V_i is a product of s linear subspaces $V_{i,j}$ of $\mathbb{F}^{n/s}$. This can be viewed as splitting $\mathbf{c}_i$ 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 [24]). *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:*

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

where $\mathbf{c}_i = \mathbf{a}_i^\top \mathbf{C}$ *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* $\mathbf{e}_{i,j} = \mathbf{d}_{i,j}^\top \mathbf{V}_{i,j}$ *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. [24]:

Conjecture 3.1 (Split-LSN conjecture [24]). *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 [16, 22] and further analyzed in [6, 14, 15]. See [24] 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 6) 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 split-LSN assumption. To get around this difficulty, we modify the assumption 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 [16].[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:*

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

where $\mathbf{u}_i \in \mathbb{F}^n$, $\mathbf{c}_i = \mathbf{a}_i^{\mathsf{T}} \mathbf{C}$ for $\mathbf{a}_i \leftarrow \mathbb{F}^k$, $(\mathbf{c}_{i,1}, \dots, \mathbf{c}_{i,s})$ 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:

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 6 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 6, we will also consider a more conservative variant of 1D-SLSN where each sample uses an independently random partition into s blocks.

[4]A different variant of this assumption from [22] 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.

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 ℓ. To make these costs scale quasilinearly or even linearly with ℓ, we can choose C from suitable families of *structured* codes. Indeed, the secret-key PIR constructions from [16, 22] 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, 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^{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 [9].

Ring variants of this kind serve as common substitutes for random linear codes in the context of cryptography from noisy linear algebra [13, 44, 53, 55]. 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 [3, 11, 32, 35, 40]. Alternatively, one can use a recent construction of fast dual codes from [20]. 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 $\mathbf{C} = [\, I \mid \mathbf{C}_1 \,]$ and $\mathbf{D} = [\, -\mathbf{C}_1^{\mathsf{T}} \mid I \,]$ 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 [52].

4 EMVP from 1D-SLSN

In this section we describe our main EMVP protocol, based on 1D-SLSN, which maximizes efficiency. In the full version [4], we describe our LPN-based protocol, which yields near-optimal *asymptotic* (but not concrete) efficiency under a standard assumption.

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 α_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, the rows of $\mathbf{M}$ 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 $\tilde{\mathbf{q}} = \mathbf{c} + (\mathbf{q} \mid 0^k)$, the clients wants to obtain the product $\tilde{\mathbf{M}}\tilde{\mathbf{q}}$ (which is equal to $\mathbf{M}\mathbf{q}$ 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 α_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 α_i by taking a linear combination of the s columns vectors with coefficients α_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

$$\widehat{\mathbf{M}}\tilde{\mathbf{q}} = \mathbf{M}\mathbf{D}\tilde{\mathbf{q}} + \mathbf{R}\tilde{\mathbf{q}} = \mathbf{M}\big(\mathbf{D}(\mathbf{q} \mid 0^k) + \mathbf{D}\mathbf{c}\big) + \mathbf{r}' = \mathbf{M}\mathbf{q} + \mathbf{r}',$$

where the last equality follows by the fact that $\mathbf{D}$ is of the form $[I_\ell \mid \mathbf{D}']$ 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 $\widehat{q} = (\alpha_1 \cdot \tilde{\mathbf{q}}_1 \mid \ldots \mid \alpha_s \cdot \tilde{\mathbf{q}}_s)$, where $\tilde{\mathbf{q}} = (\tilde{\mathbf{q}}_1 \mid \ldots \mid \tilde{\mathbf{q}}_s)$ is a partition of $(\mathbf{q} \mid 0^k) + \mathbf{c}$ 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 $(\Delta_1 \mid \ldots \mid \Delta_s) = (\mathbf{q} \mid 0^k)$ (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-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 [17, 34, 38]. With this extension, the protocol is no longer field agnostic. However, the non-agnostic part is asymptotically dominated by the other costs.

Variations. 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 6), 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.

5 New Trapdoored Matrix Constructions

In this section we present new TDM constructions (definition 2.2) 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 [18, 59], 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

```
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 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 $\mathbf{D} = [I_\ell \mid \mathbf{D}']$
  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 $\widehat{\mathbf{M}} = \mathbf{M}\mathbf{D} + \mathbf{R} \in \mathbb{F}^{m \times n}$
• $\mathsf{Q}(\mathsf{sk}, \mathbf{q})$:
  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 $\tilde{\mathbf{q}} = (\mathbf{q} \mid 0^k) + \mathbf{c}$ 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 $\widehat{\mathbf{q}} = (\alpha_1 \cdot \tilde{\mathbf{q}}_1 \mid \ldots \mid \alpha_s \cdot \tilde{\mathbf{q}}_s) \in \mathbb{F}^n$, where $(\tilde{\mathbf{q}}_1 \mid \ldots \mid \tilde{\mathbf{q}}_s)$ 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 $[\widehat{\mathbf{M}}_1 \mid \ldots \mid \widehat{\mathbf{M}}_s]$ and $\widehat{\mathbf{q}}$ as $(\widehat{\mathbf{q}}_1 \mid \ldots \mid \widehat{\mathbf{q}}_s)$.
  2. Return $\mathbf{M}' = [\widehat{\mathbf{M}}_1\widehat{\mathbf{q}}_1 \mid \ldots \mid \widehat{\mathbf{M}}_s\widehat{\mathbf{q}}_s] \in \mathbb{F}^{m \times s}$
• $\mathsf{D}(\mathsf{sk}, \mathbf{M}', \mathbf{q}')$: Return $\mathbf{M}'\mathbf{p}' - \mathbf{r}' \in \mathbb{F}^m$
```

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

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 $\mathbf{R} = \mathbf{H}\mathbf{E}$, where $\mathbf{H} \in \mathbb{F}^{k \times n}$ is a public parity-check matrix for which dual LPN holds, namely $(\mathbf{H}, \mathbf{H}\mathbf{e}) \approx_c (\mathbf{H}, \mathbf{u})$ 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 td. (Here the κ_1 field elements of td determine the values of the nonzero entries of $\mathbf{E}$ and the κ_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 $\mathbf{e} \mapsto \mathbf{H}\mathbf{e}$ 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 $\mathbb{F}$ [13, 44], or pick it from any family of linear-time encodable codes that are conjectured to be (dual-)LPN-friendly [11, 19, 35].

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

The $(k, n, \mathcal{H}, \varepsilon)$-dual-LPN assumption is equivalent to LPN with dimension $k' = n - k$, noise rate ε 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., [12, Section 3.3.1]).

A standard hybrid argument implies that if $(\mathbf{H}, \mathbf{H}\mathbf{e})$ is pseudorandom for a Bernoulli noise vector $\mathbf{e}$, then so is $(\mathbf{H}, \mathbf{H}\mathbf{E})$ 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 $m := m(\lambda)$, $(\mathbf{H}, \mathbf{H}\mathbf{E}) \approx_c (\mathbf{H}, \mathbf{U})$, 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 λ 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* EC *is $\varepsilon k n + |\mathsf{EC}_\mathbf{H}|$, 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* $\mathbf{e} \mapsto \mathbf{H}\mathbf{e}$.

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.

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 $\mathbb{F}^{k\times n}$

```
TDM(1^λ, td) // Generate k × k trapdoored matrix for k = k(λ)
  (1) Use td to sample $\mathbf{H} \leftarrow \mathcal{H}(1^k, 1^n)$ and $\mathbf{E} \leftarrow \mathrm{Ber}_{\mathbb{F}}^{n\times k}(\varepsilon)$
  (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**.**

5.1.1 TDM from Ring-LPN. Ring-LPN [13, 44, 55], an LPN analogue of ring-LWE [53], 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 $\mathbf{H} = [\, I \mid \mathbf{C} \,]$, 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 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 [18, 59], where 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 quasilinear 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 [10, 12] and can be based on different families of linear-time encodable codes from [3, 11, 32, 35, 40]. 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 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 [5, 9], 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 EC superlinear. However, if $\mathbf{R}$ is a rectangular $Tk \times k$ matrix, where $T \geq t$, we can make 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 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 EC, ruling out a public EC.

Instead, we propose a heuristic approach for a universal TDM that can also support 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

$$\mathbf{R} = \mathbf{S}_L \cdot \Pi_L \cdot \mathbf{S} \cdot \Pi_R \cdot \mathbf{S}_R, \qquad (1)$$

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 matrices,
- Π_L, Π_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 [47], 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 td, where td contains $\Omega(k)$ field elements. Thus, the mapping from td to $\mathbf{R}$ is a degree-3 polynomial map determined by Π_L and Π_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 [54]), 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 Π_L, Π_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 Π_1 and Π_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 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 [35, 47]. We conjecture that for most choices of B as above (and in particular the ones from the literature), the TDM defined by eq. (1) is secure.

An RAA-style TDM candidate. In the full version, we propose a leaner variant of eq. (1) based on the design principle of Repeat-Accumulate-Accumulate (RAA) codes [3, 32, 40].

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 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 6.2 below. (In this section we use row vectors by default, vs. column vectors in the rest of the document.)

6.1 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 $\mathbf{G} \in \mathbb{F}^{k \times n}$ (which we almost always assume is systematic, $\mathbf{G} = [\mathbf{I}_k | \mathbf{X}]$ 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 $\mathbf{c} = \mathbf{r}\mathbf{G}$ 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 in each vector is uniform in one block and 0 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 distance between the two choices of samples is $\leq 1/|\mathbb{F}|$. 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 γ_d is also uniform in $\mathbb{F}^d$. But since $\mathbf{c}$ is itself a random codeword, the only difference between these distributions is that in the linear case we allow $\gamma_d = 0$ whereas the affine case (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 described below against the linear-space distribution imply an attacks 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 basis of vectors spanning the same subspace.

Finally, for the 1D-SLSN case from Definition 3.3, all our attacks use 0 shifts, and it is unclear how to use shifts to improve them.

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 6.1.1 below, this attack can be thought of as an adaptation of the Arora-Ge type of attacks [2] to our secret-code setting.) The attack uses the following simple lemma:

Lemma 6.1. *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 exists coefficients γ_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., π is of full rank). Such a linear 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 $\pi(\mathbf{v}_i)^\top$ as columns, $\mathbf{V}_\pi = [\pi(\mathbf{v}_1)^\top | \cdots | \pi(\mathbf{v}_d)^\top]$. 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 π 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 γ_i that are not all zero. As C is in the kernel of π, 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 γ_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, $P_C(\mathbf{v}_1, \ldots, \mathbf{v}_d) = \det([\pi(\mathbf{v}_1)^\top | \cdots | \pi(\mathbf{v}_d)^\top]) = 0$. □

The basic attack. We note that regardless of C or π, the monomials in $\det([\pi(\mathbf{v}_1)^\top | \cdots | \pi(\mathbf{v}_d)^\top])$ 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, each can be described by at most nd scalars. Hence we consider the complexity of this attack to be "essentially n^d".

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 π satisfying the conditions in the proof of lemma 6.1, where the set of possible monomials in $\det(\mathbf{V}_\pi)$ is smaller than n^d.

6.1.1 *Relations to Algebraic Attacks from the Literature.*

As we mentioned above, we can view our algebraic attacks as a variant of the Arora-Ge attacks [2] on LPN, adjusted to handle our setting where the code is secret. In the attacks from [2], the LPN secret $\mathbf{s}$ plays the same role as the code C in our setting. They consider an 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 sophisticated attacks based on Gröbner bases/XL [28] 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, they allow better usage of the limited samples available 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 [21, 49]) 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.

6.1.2 *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 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]

6.1.3 *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 $\mathbf{G} = [\mathbf{I}_k | \mathbf{X}]$, and extend $\mathbf{G}$ 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 $(\mathbf{u}|\mathbf{v}) = \mathbf{r}\mathbf{G} = (\mathbf{r}|\mathbf{0})\mathbf{M} = (\mathbf{r}|\mathbf{r}\mathbf{X})$ 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 π. At the same time, the image of π 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 π therefore satisfies the conditions of lemma 6.1, and we can use it in the attack from above. With this π, the polynomial P_C from lemma 6.1 is the

[5] Even for blocks that are rather structured but not quite fixed then we don't know how to do better than $(k+1)^d$. For example 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.

determinant

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

which is a sum of all the diagonal products of that matrix with coefficients ± 1. Each one of these diagonal products has the form $\prod_{i=1}^{d}(v_{ij}-\langle\mathbf{u}_j,\mathbf{x}_i\rangle)$ 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 $S\subseteq\{1,\dots,d\}$, the determinant therefore has a term corresponding the 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 $k^{|S|}$). This can be expressed as a big inner product that 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 $i=1,\dots,d$), $\binom{d}{2}$ 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, which would happen after at most $(k+1)^d$ regular-LSN samples (but whp only after $(k+1)^d+1$ random samples). Hence the attack complexity is essentially $(k+1)^d$. We describe more cryptanalysis results in the full version of the paper [4].

6.2 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-ℓ vector over the field $\mathbb{F}$;
- Records are encoded using an (ℓ,n)-linear code D (so the server's overhead is a factor of n/ℓ).
- Denoting $k=n-\ell$, the code that we want to hide using 1D-Split-LSN is the dual of D, which we denote by C. This is 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 λ the security parameter, and choose parameters so that the attacks from section 6 all have complexity more than 2^λ. For the 1D-Split-LSN constructions we need $b^{\lceil k/(b-1)\rceil}\ge 2^\lambda$ due to the algebraic attack from section 6.1.2.

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 sending the full matrix is: $\ell/s=b/f$, since the server returns s field elements per row, instead of ℓ 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 ℓ. 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 $b^{\lceil k/(b-1)\rceil}\ge 2^\lambda$ as $\lceil k/(b-1)\rceil\log_2 b\ge\lambda$, we can ignore the ceiling and get the sufficient condition $k\ge\lambda\cdot(b-1)/\log_2 b$. This constraint, together with $k=\ell(f-1)$ imply 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 dataset to the client for every query, and since b is an integer then $b\ge\lfloor f\rfloor+1$ (and in particular $b\ge 2$).
- For any value of ℓ above this minimal value, we can maximize the compression ratio b/f by using the largest value of b that satisfies $(b-1)/\log_2 b\le k/\lambda$.

We list some parameter examples in table 1, from the smallest possible record size ℓ upto $\ell=10000$ (which is roughly the size used in the World ID application [7]).

6.2.1 *Can We Get Better Parameters?*

As we commented at the bottom of section 6.1.2, 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 $\lceil k/(b-1)\rceil\log_2(k+1)\ge\lambda$, we ignore the +1 and ceiling and get the sufficient condition $k\log_2 k\ge\lambda\cdot(b-1)$. 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 ℓ. For values of ℓ 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 $k\ge 28$, 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.

Acknowledgments

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.

References

[1] Benny Applebaum, Jonathan Avron, and Chris Brzuska. 2017. Arithmetic Cryptography. *J. ACM* 64, 2 (2017), 10:1–10:74. doi:10.1145/3046675

[2] Sanjeev Arora and Rong Ge. 2011. New Algorithms for Learning in Presence of Errors. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 6755)*, Luca Aceto, Monika Henzinger, and Jirí Sgall (Eds.). Springer, 403–415. doi:10.1007/978-3-642-22006-7_34

[3] Sergio Benedetto, Daniel Divsalar, Guido Montorsi, and Fabrizio Pollara. 1998. Analysis, Design, and Iterative Decoding of Double Serially Concatenated Codes with Interleavers. *IEEE Journal on Selected Areas in Communications* 16, 2 (Feb. 1998), 231–244. doi:10.1109/49.661107

[4] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. 2025. Encrypted Matrix-Vector Products from Secret Dual Codes. Cryptology ePrint Archive, Paper 2025/858. https://eprint.iacr.org/2025/858

[5] Daniel J. Bernstein. [n. d.]. The Transposition Principle. https://cr.yp.to/transposition.html. Accessed: 2025-05-06.

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 $\widehat{M} \in \mathbb{F}^{m\times n}$ by $\widehat{q} \in \mathbb{F}^n$, the answer is $M' \in \mathbb{F}^{m\times s}$ for $s = n/b$, and the client's local work (with ring-LSN) is $\tilde{O}(n) + 2sm$ field operations.

record length ℓ	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)

[6] Keller Blackwell and Mary Wootters. 2021. A note on the permuted puzzles toy conjecture. *arXiv preprint arXiv:2108.07885* (2021).

[7] Remco Bloemen, Daniel Kales, Philipp Sippl, and Roman Walch. 2024. Large-Scale MPC: Scaling Private Iris Code Uniqueness Checks to Millions of Users. *IACR Cryptol. ePrint Arch.* (2024), 705. https://eprint.iacr.org/2024/705

[8] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. 1994. Cryptographic Primitives Based on Hard Learning Problems. In *Advances in Cryptology — CRYPTO' 93*, Douglas R. Stinson (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 278–291.

[9] F. Bordewijk. 1956. Interreciprocity Applied to Electrical Networks. *Applied Scientific Research* 6, 1 (1956), 1–14. doi:10.1007/BF03184936

[10] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. 2018. Compressing Vector OLE. In *ACM CCS 2018: 25th Conference on Computer and Communications Security*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, Toronto, ON, Canada, 896–912. doi:10.1145/3243734.3243868

[11] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. 2022. Correlated Pseudorandomness from Expand-Accumulate Codes. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13508)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, 603–633. doi:10.1007/978-3-031-15979-4_21

[12] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. 2019. Efficient Pseudorandom Correlation Generators: Silent OT Extension and More. In *Advances in Cryptology – CRYPTO 2019, Part III (Lecture Notes in Computer Science, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Cham, Switzerland, Santa Barbara, CA, USA, 489–518. doi:10.1007/978-3-030-26954-8_16

[13] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. 2020. Efficient Pseudorandom Correlation Generators from Ring-LPN. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, 387–416. doi:10.1007/978-3-030-56880-1_14

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

[15] Elette Boyle, Justin Holmgren, and Mor Weiss. 2019. Permuted Puzzles and Cryptographic Hardness. In *TCC 2019: 17th Theory of Cryptography Conference, Part II (Lecture Notes in Computer Science, Vol. 11892)*, Dennis Hofheinz and Alon Rosen (Eds.). Springer, Cham, Switzerland, Nuremberg, Germany, 465–493. doi:10.1007/978-3-030-36033-7_18

[16] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. 2017. 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*. Springer, 662–693.

[17] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2019. Leveraging Linear Decryption: Rate-1 Fully-Homomorphic Encryption and Time-Lock Puzzles. In *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11892)*, Dennis Hofheinz and Alon Rosen (Eds.). Springer, 407–437. doi:10.1007/978-3-030-36033-7_16

[18] Mark Braverman and Stephen Newman. 2025. Sublinear-Overhead Secure Linear Algebra on a Dishonest Server. *CoRR* abs/2502.13060 (2025). arXiv:2502.13060 doi:10.48550/ARXIV.2502.13060

[19] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. 2024. Blaze: Fast SNARKs from Interleaved RAA Codes. *IACR Cryptol. ePrint Arch.* (2024), 1609. https://eprint.iacr.org/2024/1609

[20] Martijn Brehm and Nicolas Resch. 2025. Linear time encodable binary code achieving GV bound with linear time encodable dual achieving GV bound. *arXiv preprint arXiv:2509.07639* (2025). https://arxiv.org/abs/2509.07639

[21] Pierre Briaud and Morten Øygarden. 2023. A New Algebraic Approach to the Regular Syndrome Decoding Problem and Implications for PCG Constructions. In *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 (Lecture Notes in Computer Science, Vol. 14008)*, Carmit Hazay and Martijn Stam (Eds.). Springer, 391–422. doi:10.1007/978-3-031-30589-4_14

[22] Ran Canetti, Justin Holmgren, and Silas Richelson. 2017. 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*. Springer, 694–726.

[23] Aidao Chen, Anindya De, and Aravindan Vijayaraghavan. 2021. Learning a mixture of two subspaces over finite fields. In *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide (Proceedings of Machine Learning Research, Vol. 132)*, Vitaly Feldman, Katrina Ligett, and Sivan Sabato (Eds.). PMLR, 481–504. http://proceedings.mlr.press/v132/chen21a.html

[24] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. 2025. Secret-Key PIR from Random Linear Codes. Cryptology ePrint Archive, Paper 2025/646. https://eprint.iacr.org/2025/646

[25] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017, Part I (Lecture Notes in Computer Science, Vol. 10624)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer, Cham, Switzerland, Hong Kong, China, 409–437. doi:10.1007/978-3-319-70694-8_15

[26] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private Information Retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*. IEEE Computer Society, 41–50. doi:10.1109/SFCS.1995.492461

[27] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. 2015. New Unconditional Hardness Results for Dynamic and Online Problems. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, Venkatesan Guruswami (Ed.). IEEE Computer Society, 1089–1107. doi:10.1109/FOCS.2015.71

[28] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. 2000. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding (Lecture Notes in Computer Science, Vol. 1807)*, Bart Preneel (Ed.). Springer, 392–407. doi:10.1007/3-540-45539-6_27

[29] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM CCS 2006: 13th Conference on Computer and Communications Security*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, Alexandria, Virginia, USA, 79–88. doi:10.1145/1180405.1180417

[30] Alex Davidson, Gonçalo Pestana, and Sofía Celi. 2023. FrodoPIR: Simple, Scalable, Single-Server Private Information Retrieval. *Proc. Priv. Enhancing Technol.* 2023, 1 (2023), 365–383. doi:10.56553/POPETS-2023-0022

[31] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. 2000. Single Database Private Information Retrieval Implies Oblivious Transfer. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding (Lecture Notes in Computer Science, Vol. 1807)*, Bart Preneel (Ed.). Springer, 122–138. doi:10.1007/3-540-45539-6_10

[32] Daniel Divsalar, Hui Jin, and Robert J. McEliece. 1998. Coding Theorems for "Turbo-Like" Codes. In *Proc. 36th Annual Allerton Conference on Communication, Control, and Computing*. Monticello, IL, USA, 201–210.

[33] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. 2009. On cryptography with auxiliary input. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, Michael Mitzenmacher (Ed.). ACM, 621–630. doi:10.1145/1536414.1536498

[34] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. 2019. Trapdoor Hash Functions and Their Applications. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 3–32. doi:10.1007/978-3-030-26954-8_1

[35] Erez Druk and Yuval Ishai. 2014. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, Moni Naor (Ed.). ACM, 169–182. doi:10.1145/2554797.2554815

[36] Kasra Edalatnejad, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, Massimo Marelli, and Carmela Troncoso. 2024. 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*. IEEE, 655–672. doi:10.1109/SP54263.2024.00116

[37] Adrià Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. 2024. Computationally Secure Aggregation and Private Information Retrieval in the Shuffle Model. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 4122–4136. doi:10.1145/3658644.3670391

[38] Craig Gentry and Shai Halevi. 2019. Compressible FHE with Applications to PIR. In *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11892)*, Dennis Hofheinz and Alon Rosen (Eds.). Springer, 438–464. doi:10.1007/978-3-030-36033-7_17

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

[40] Venkatesan Guruswami and Widad Machmouchi. 2008. Explicit interleavers for a Repeat Accumulate Accumulate (RAA) code construction. In *2008 IEEE International Symposium on Information Theory, ISIT 2008, Toronto, ON, Canada, July 6-11, 2008*, Frank R. Kschischang and En-Hui Yang (Eds.). IEEE, 1968–1972. doi:10.1109/ISIT.2008.4595333

[41] Jiaxing He, Kang Yang, Guofeng Tang, Zhangjie Huang, Li Lin, Changzheng Wei, Ying Yan, and Wei Wang. 2024. Rhombus: Fast Homomorphic Matrix-Vector Multiplication for Secure Two-Party Inference. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 2490–2504. doi:10.1145/3658644.3690281

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

[43] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. 2023. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 3889–3905. https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger

[44] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. 2012. Lapin: An Efficient Authentication Protocol Based on Ring-LPN. In *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers (Lecture Notes in Computer Science, Vol. 7549)*, Anne Canteaut (Ed.). Springer, 346–365. doi:10.1007/978-3-642-34047-5_20

[45] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. 2005. Sufficient Conditions for Collision-Resistant Hashing. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3378)*, Joe Kilian (Ed.). Springer, 445–456. doi:10.1007/978-3-540-30576-7_24

[46] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2006. Cryptography from Anonymity. In *47th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Berkeley, CA, USA, 239–248. doi:10.1109/FOCS.2006.25

[47] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2008. Cryptography with constant computational overhead. In *40th Annual ACM Symposium on Theory of Computing*, Richard E. Ladner and Cynthia Dwork (Eds.). ACM Press, Victoria, BC, Canada, 433–442. doi:10.1145/1374376.1374438

[48] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. 2009. Secure Arithmetic Computation with No Honest Majority. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5444)*, Omer Reingold (Ed.). Springer, 294–314. doi:10.1007/978-3-642-00457-5_18

[49] Vladimir Kolesnikov, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. 2025. Stationary Syndrome Decoding for Improved PCGs. *IACR Cryptol. ePrint Arch.* (2025), 295. https://eprint.iacr.org/2025/295

[50] Eyal Kushilevitz and Rafail Ostrovsky. 1997. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *38th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Miami Beach, Florida, 364–373. doi:10.1109/SFCS.1997.646125

[51] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. 2023. 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* (Orlando, FL, USA) *(STOC 2023)*. Association for Computing Machinery, New York, NY, USA, 595–608. doi:10.1145/3564246.3585175

[52] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. 2024. The Hardness of LPN over Any Integer Ring and Field for PCG Applications. In *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 (Lecture Notes in Computer Science, Vol. 14656)*, Marc Joye and Gregor Leander (Eds.). Springer, 149–179. doi:10.1007/978-3-031-58751-1_6

[53] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *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 (Lecture Notes in Computer Science, Vol. 6110)*, Henri Gilbert (Ed.). Springer, 1–23. doi:10.1007/978-3-642-13190-5_1

[54] Tsutomu Matsumoto and Hideki Imai. 1988. Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In *Advances in Cryptology - EUROCRYPT '88 (Lecture Notes in Computer Science, Vol. 330)*, Christof G. Günther (Ed.). Springer, Davos, Switzerland, 419–453.

[55] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. 2018. Efficient Encryption From Random Quasi-Cyclic Codes. *IEEE Trans. Inf. Theory* 64, 5 (2018), 3927–3943. doi:10.1109/TIT.2018.2804444

[56] Moni Naor and Benny Pinkas. 1999. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton (Eds.). ACM, 245–254. doi:10.1145/301250.301312

[57] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. 2000. Practical Techniques for Searches on Encrypted Data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*. IEEE, 44–55. doi:10.1109/SECPRI.2000.848445

[58] Aikaterini Sotiraki. 2016. *Authentication protocol using trapdoored matrices*. Master's thesis. Massachusetts Institute of Technology.

[59] Vinod Vaikuntanathan and Or Zamir. 2025. Improving Algorithmic Efficiency using Cryptography. *CoRR* abs/2502.13065 (2025). arXiv:2502.13065 doi:10.48550/ARXIV.2502.13065