Oblivious Ciphertext Compression via Linear Codes

Pascal Giorgi[1], Bruno Grenet[2], and Mark Simkin[3,4]

[1] LIRMM, Univ. Montpellier, CNRS
pascal.giorgi@lirmm.fr
[2] LJK, Univ. Grenoble Alpes, CNRS
bruno.grenet@univ-grenoble-alpes.fr
[3] Flashbots
[4] Aarhus University
mark@univariate.org

Abstract. Oblivious ciphertext compression and decompression transform encrypted dense vectors of length n with at most t non-zero entries into compact encrypted sparse representations, and vice versa. These primitives appear in the context of efficient protocols for encrypted search, PIR, and oblivious message retrieval. Existing schemes suffer from large ciphertext sizes or high computational cost. We present new deterministic and perfectly correct constructions based on linear codes, yielding encrypted sparse representations of optimal size with near-linear compression and decompression times. Our results improve both communication and computation over prior work. A central ingredient of our work is to show that, for carefully chosen generalized Reed–Solomon codes, variants of classical decoding algorithms combined with efficient algebraic techniques enable to recover the error vector directly from the syndrome in quasi-linear time in the syndrome length, rather than in the full block length of the code.

1 Introduction

A vector $\boldsymbol{v}$ of length n with at most t non-zero entries can be described in two very different ways: In its *dense* representation, $\boldsymbol{v}$ is given as an ordered list of all n entries, including the zeros. In its *sparse* representation, it is specified by a list that only contains the non-zero entries and their locations in the vector. The dense form is often preferable for performing computations, whereas the sparse form is advantageous for storage and communication, especially when $t \ll n$.

Switching between the two representations is trivial in the clear, but becomes much more challenging when working with encrypted data, where the locations of the zero entries are hidden. Nevertheless, efficient conversion is essential for the practicality of many protocols, including encrypted search [YSK+13, CKK16, AFS18, CDG+21], private information retrieval [ACLS18, BPSY23], and oblivious message retrieval [LT22, LTW24].

Oblivious Ciphertext Compression. Several works have studied how to obliviously compress an encrypted vector of bounded Hamming weight, converting it from a dense form into a space-efficient sparse representation [CDG+21, LT22, FLS23, FLOS24, BPSY23, LTW24]. In this setting, a server holds a vector $\boldsymbol{v} \in \mathbb{F}^n$ encrypted coordinate by coordinate under a homomorphic encryption scheme, while the client holds the corresponding decryption key. The server's task is to transform the encrypted vector into a compact representation, under the assumption that at most t entries of $\boldsymbol{v}$ are non-zero, and then send this compressed form to the client. Upon decryption, the client should be able to recover a sparse representation of $\boldsymbol{v}$ using work proportional to t rather than n. Some work, like that of Bienstock et al. [BPSY23], also consider a relaxed setting in which the client is additionally given the locations of the non-zero entries as auxiliary input.

	Encoding Size	Time		Without Support	Perfect Correctness	Plaintext Space
		Compression	Decompression			
[CDG+21]	$\mathcal{O}(t \cdot \kappa)$	$\mathcal{O}(n \cdot \kappa)$	$\mathcal{O}(t \cdot \kappa)$	✓	✗	$\mathbb{G}$
[LT22]	$\mathcal{O}\big(t \cdot (\lg^2 t + \lg \kappa)\big)$	$\mathcal{O}(n \cdot t)$	$\mathcal{O}\big(t^3\big)$	✓	✗	$\mathbb{F}_q$ with $q > 2^\kappa$
[FLS23]	$\mathcal{O}(\kappa \cdot t / \lg t)$	$\mathcal{O}(n \cdot \kappa / \lg t)$	$\mathcal{O}(t \cdot \kappa / \lg t)$	✓	✗	$\mathbb{F}_q$ with $q > 2^\kappa$
[FLS23]	2t	$\mathcal{O}(n \lg n)$	$\mathcal{O}(t \cdot \sqrt{n})$	✓	✓	${}^{(\dagger)}\mathbb{F}_q$
[FLOS24]	$\mathcal{O}(t + \kappa \cdot \lg \kappa)$	$\mathcal{O}(n \cdot \kappa)$	$\mathcal{O}(t \cdot \kappa)$	✓	✗	$\mathbb{F}_q$ with $q > 2^\kappa$
[BPSY23]	$t \cdot (1 + \epsilon)$	$\mathcal{O}(n \cdot \kappa \cdot 1/\epsilon)$	$\mathcal{O}(t \cdot \kappa \cdot 1/\epsilon)$	✗	✓	$\mathbb{F}_q$
This Work	$2t$	$\mathcal{O}\big(n \cdot \lg^2 t\big)$	$\mathcal{O}\big(t \cdot \lg^2 t \cdot \lg q\big)$	✓	✓	${}^{(\dagger)}\mathbb{F}_q$
This Work	$2t$	$\mathcal{O}(n \cdot \lg t)$	$\mathcal{O}\big(t \cdot \lg^2 t \cdot \ell\big)$	✓	✓	$\mathbb{F}_{2^\ell}$
This Work	t	$\mathcal{O}(n \cdot \lg t)$	$\mathcal{O}\big(t \cdot \lg^2 t\big)$	✗	✓	${}^{(\dagger)}\mathbb{F}_q$
This Work	t	$\mathcal{O}(n \cdot \lg t)$	$\mathcal{O}\big(t \cdot \lg^2 t\big)$	✗	✓	$\mathbb{F}_{2^\ell}$

Table 1. Comparison of oblivious compression schemes. Randomized constructions are assumed to fail with probability at most $2^{-\kappa}$ and we assume $\kappa > \lg n$. The value $\epsilon > 0$ is a constant. Encoding sizes are counted in ciphertexts. Computation times assume arithmetic operations and hash function evaluations are constant time operations. $\mathbb{G}$ denotes any abelian group, $\mathbb{F}_q$ a finite field, and ${}^{(\dagger)}\mathbb{F}_q$ a finite field with $q > n$, where FFTs can be computed in time $\mathcal{O}(n \lg n)$. When the client requires the locations of non-zero indices as auxiliary input, we say it requires support and otherwise the construction is said to work without support.

Oblivious Ciphertext Decompression. Complementary to oblivious compression, Bienstock et al. [BPSY23] recently studied the task of oblivious ciphertext decompression. In this setting, the client holds a sparse representation $\{(i, \boldsymbol{v}_i)\}_{i \in I}$ of a vector $\boldsymbol{v}$ with at most t non-zero entries. The goal is to encrypt this sparse input so that the server can efficiently recover an encrypted dense vector $\tilde{\boldsymbol{v}}$, where $\boldsymbol{v}_i = \tilde{\boldsymbol{v}}_i$ for all $i \in I$. The values at the remaining coordinates of $\tilde{\boldsymbol{v}}$ may be arbitrary, making this a somewhat unusual form of sparse-to-dense transformation. Bienstock et al. showed that oblivious ciphertext decompression is a key building block for batch private information retrieval schemes with improved communication complexity.

Although both problems have received considerable attention in recent works, existing approaches still suffer from suboptimal compression or decompression rates or nontrivial computational overheads.

1.1 Our Contribution

In this work, we study oblivious ciphertext compression and decompression through the lens of linear codes and present new constructions that improve upon prior results in multiple ways. We show a general blueprint for reducing the problem of oblivious ciphertext compression and decompression to that of constructing certain good linear codes. Instantiating our blueprint carefully, we obtain compression (for both the setting with and without auxiliary client input) and decompression schemes with encrypted sparse representations of optimal size and improved computational complexities. Notably, our constructions are all fully deterministic and perfectly correct[5].

A key ingredient in our constructions is a new algorithmic result for (generalized) Reed–Solomon codes. In coding theory, a syndrome is a compressed representation of an error vector and classical decoding algorithms usually aim to reconstruct the original noise-free codeword with the help of the syndrome. In contrast to this, we show that by carefully choosing the underlying generalized Reed–Solomon code, one can efficiently recover the error vector directly from the syndrome, without ever accessing the noisy codeword itself. Our algorithms achieve quasi-linear running time in the syndrome length, rather than in the full block length of the code.

1.2 Related Works

The task of compressing and decompressing encrypted data has been studied in the literature before. In the following, we summarize and compare our results to those that are closest to ours.

[5] Encryption schemes by themselves may not be perfectly correct. Our compression/decompression schemes are perfectly correct in the sense that they do not introduce any additional error probabilities.

Strawman Solution. Fully homomorphic encryption [RAD78, Gen09] enables arbitrary computations on encrypted data. In principle, this makes oblivious ciphertext compression and decompression trivial, as any plaintext algorithm can be emulated on the encrypted data. However, such an approach is computationally inefficient, as it relies on homomorphic multiplications between ciphertexts, which are significantly more expensive than additions or multiplications by constants. For this reason, we restrict attention to oblivious compression and decompression schemes that only rely on homomorphic additions and multiplications by constants.

Oblivious Ciphertext Compression Schemes. Over the last few years, several works have presented constructions for oblivious ciphertext compression [CDG+21, LT22, FLS23, FLOS24, BPSY23, LTW24]. The works of Choi et al. [CDG+21], Liu and Tromer [LT22], and Fleischhacker, Larsen, Obremski and Simkin [FLS23, FLOS24], and Liu, Tromer, and Wang [LTW24] all present randomized constructions that compress an encrypted vector of length n with hamming weight at most t, into an encrypted sparse representation comprised of $\omega(t)$ ciphertexts[6]. Bienstock et al. [BPSY23] consider a slightly relaxed setting, where the client is given the locations of the non-zero entries in the server's vector as auxiliary input. The authors show that the relaxed problem with auxiliary input can be solved with close to optimal bandwidth overhead and moderate computational overheads. Concretely, they show how to compute an encrypted sparse representation of size $t \cdot (1 + \epsilon)$, where $\epsilon > 0$ is some small constant.

In contrast to these works, we present constructions that result in encrypted sparse representations of size t and $2t$ for the setting with and without auxiliary input. The server can compress their vector of length n in time $\mathcal{O}(n \operatorname{polylog} t)$ and the client can recover the plaintext sparse representation in time $\mathcal{O}(t \operatorname{polylog} t)$.

The only existing construction that is also perfectly correct and compresses the server's encrypted vector down to $2t$ ciphertexts is due to Fleischhacker, Larsen, and Simkin [FLS23]. Unfortunately, their construction is prohibitively expensive for the client, who needs to perform $\mathcal{O}(t\sqrt{n})$ arithmetic operations to recover the sparse vector representation. A detailed comparison of existing oblivious compression schemes is provided in Table 1.

Liu and Tromer [LT22] use Sparse Random Linear Codes in one of their constructions. Nevertheless, their analysis is only based on the properties of random matrices rather than coding-theoretic considerations. And subsequent constructions such as [BPSY23] directly work with random structured matrices. A novelty of our work is to exhibit the coding-theoretic nature of any oblivious compression scheme based on additively homomorphic encryption schemes and the role of Syndrome Decoding in this problem.

Oblivious Ciphertext Decompression Schemes. Bienstock et al. [BPSY23] present a very efficient randomized oblivious decompression scheme. Their construction enables the client to encrypt their sparse input vector into $t \cdot (1 + \epsilon)$ ciphertexts, where $\epsilon > 0$ is again some small constant. In terms of computational overhead, the client and the server need to perform $\mathcal{O}(t\kappa \cdot 1/\epsilon)$ and $\mathcal{O}(n\kappa \cdot 1/\epsilon)$ homomorphic operations respectively, where $\mathcal{O}(2^{-\kappa})$ is the probability of their construction producing an incorrect result. In contrast to their work, our approach is fully deterministic, perfectly correct, encrypts the sparse input vector into exactly t ciphertexts, and the client and the server only need to perform $\mathcal{O}\big(t \lg^2 t\big)$ and $\mathcal{O}(n \lg t)$ homomorphic operations respectively.

1.3 Applications

Oblivious ciphertext compression and decompression schemes are useful building blocks in a variety of applications. In the following we highlight a few of them.

Secure Search. In the encrypted search setting, a client wishes to retrieve all database entries matching an encrypted query, while the server only operates on encrypted data. Protocols based on fully homomorphic encryption [AFS18, AGHL19, WYXZ20, CDG+21] typically follow a two-step approach. First, the server

[6] For the sake of simplicity, we assume that ciphertexts are of some unit size and that homomorphic operations on ciphertexts do no increase their size.

homomorphically evaluates the query on every entry in the database, replacing non-matches with encryptions of zero. Then, under the assumption that only a bounded number of entries match, the server applies oblivious ciphertext compression to the resulting sparse vector before returning it to the client.

Oblivious Message Retrieval. In the oblivious message retrieval setting [LT22, LTW24], a server stores encrypted messages for many clients without knowing which messages belong to whom. When a client connects, it should efficiently obtain exactly the messages intended for them, without revealing their identity or which messages were retrieved. Existing approaches closely mirror the structure of encrypted search protocols: the client submits an encrypted query identifying its messages, the server homomorphically filters the database by zeroing out irrelevant entries, and then uses oblivious ciphertext compression to compactly return the remaining ciphertexts to the client.

Single-Server PIR. In the private information retrieval (PIR) setting [CGKS95], a client seeks to obtain a single entry from a server's database without revealing which entry is being accessed. The batch PIR variant [BIM00] extends this to the case where the client retrieves t entries simultaneously, aiming for the server's computation to scale sublinearly in t. One of the most practically efficient constructions, due to Angel et al. [ACLS18], shows how to turn any single-query PIR scheme into an efficient batch PIR scheme. Their construction works by deriving $1.5t$ smaller databases from the server's original database, arranged so that each of the t requested entries lies in a distinct derived database. The client then performs one single-query PIR on each of these derived databases: in t cases it queries the desired entries, and in the remaining $0.5t$ cases it submits encryptions of 0. The server evaluates all $1.5t$ queries in parallel, producing ciphertexts of which $0.5t$ decrypt to 0.

Bienstock et al. [BPSY23] showed that oblivious ciphertext decompression and compression can be used to reduce the communication between client and server. Using our schemes instead of those by Bienstock et al., we can further reduce the overhead of the resulting batch PIR protocol, while providing perfect correctness.

Two-Server PIR. In two-server (batch) PIR, the database is fully replicated among two non-colluding machines, both interacting with the client. Bienstock et al. [BPSY23] present a two-server PIR construction based on distributed point functions [GI14, BGI15] and their oblivious ciphertext compression scheme. For this construction, they exploit the fact that compressing a vector is done via a matrix-vector product, which is then shown to be compatible with the distributed point function approach. Since our compression scheme is also comprised of a matrix-vector product with an appropriately chosen matrix, we can directly apply our results to their two-server batch PIR construction to obtain a perfectly correct construction with improved bandwidth overhead and reduced computational complexity.

2 Preliminaries

Notation. We denote the computational security parameter by λ. We write $x \leftarrow X$ to denote the process of sampling x uniformly at random from the set of elements X. For a random algorithm A and input x, we write $y \leftarrow A(x)$ to denote the process of running A with input x and uniformly random coins and assigning the output to y. For a bit string x, we write $|x|$ to denote its bit length. We write $[n]$ to denote the set $\{1, \ldots, n\}$. For a vector $\boldsymbol{x} \in X^n$, we write x_i to denote the ith component of the vector. For a function $f\colon X \to Y$ and a vector $\boldsymbol{x} \in X^n$ we will sometimes abuse notation and write $f(\boldsymbol{x})$ to denote the vector $(f(x_1), \ldots f(x_n))$. For a vector $\boldsymbol{x} \in X \cup \{0\}$ for some set X, we write $\mathsf{hw}(\boldsymbol{x})$ to denote the hamming weight, i.e. the number of non-zero entries, of $\boldsymbol{x}$. We define the sparse representation of $\boldsymbol{x}$ as $\mathsf{sparse}(\boldsymbol{x}) := \{(x_i, i) \mid x_i \neq 0\}$ and refer to $\boldsymbol{x}$ as the dense representation, i.e. $\mathsf{dense}(\mathsf{sparse}(\boldsymbol{x}))$.

We will often use the classical complexity function notation ($\mathsf{M} : \mathbb{N} \mapsto \mathbb{N}$) for the algebraic cost of polynomial multiplication, that is, two polynomials of degree $< n$ can be multiplied using $\mathsf{M}(n)$ algebraic operations. Note that the value of $\mathsf{M}(n)$ depends on the underlying ring of coefficients. We will also make heavy use of the so-called transposition principle [BLS03] which states that a linear algorithm that performs

a matrix-vector product can be *transposed* to produce another algorithm for computing the transposed matrix-vector product with the same algebraic complexity.

2.1 Homomorphic Encryption

In the following, we formally define homomorphic encryption. For the sake of simplicity, we focus on fully homomorphic encryption schemes that allow for repeatedly evaluating arithmetic circuits on a vector of ciphertexts. All of our results trivially generalize to somewhat homomorphic encryption schemes.

Definition 1. *A homomorphic encryption scheme* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is a tuple of PPT algorithms that are defined as follows:*

$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$**:** *The key generation algorithm takes security parameter* λ *as input and returns encryption key* ek *and decryption key* dk*. The encryption key implicitly defines the ciphertext space* $\mathcal{E}$*.*

$c \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$**:** *The encryption algorithm takes encryption key* ek *as well as message* $m \in \mathcal{M}$ *as input and returns a ciphertext* $c \in \mathcal{E}$*.*

$\tilde{\boldsymbol{c}} \leftarrow \mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{c})$**:** *The evaluation algorithm takes encryption key* ek*, function* $f : \mathcal{M}^n \to \mathcal{M}^m$*, and ciphertext vector* $\boldsymbol{c} \in \mathcal{E}^n$ *as input and returns a new ciphertext vector* $\tilde{\boldsymbol{c}} \in \mathcal{E}^m$*.*

$m \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$**:** *The decryption algorithm takes decryption key* dk *and ciphertext* $c \in \mathcal{E}$ *as input and returns message* $m \in \mathcal{M} \cup \{\perp\}$*.*

Definition 2 (Correctness). *Let* $\ell, n_1, \ldots, n_\ell, m_1, \ldots, m_\ell \in \mathbb{N}$ *with* $m_i = n_{i+1}$ *for all* $i \in [\ell - 1]$*. Let* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a homomorphic encryption scheme with message space* $\mathcal{M}$*. Let* $C_1, \ldots, C_\ell$ *be arithmetic circuits with* $C_i : \mathcal{M}^{n_i} \to \mathcal{M}^{m_i}$ *for* $i \in [\ell]$*. Then* E *is said to be correct, if for any security parameter* $\lambda \in \mathbb{N}$ *and any vector* $\boldsymbol{m} \in \mathcal{M}^n$*, it holds that*

$$\Pr\left[C_\ell(\ldots C_1(\boldsymbol{m})\ldots) \neq \boldsymbol{m}' \;:\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}_i \leftarrow \mathsf{Eval}(\mathsf{ek}, C_i, \mathbf{ct}_{i-1}) \;\forall\; i \in [\ell] \\ \boldsymbol{m}' \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathbf{ct}_\ell) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

there the probability is taken over the random coins of all involved algorithms.

We say an encryption scheme is additively homomorphic, if it supports addition of encrypted values and multiplication of encrypted values by public constants, but not necessarily multiplication of two encrypted values. For notational convenience, we will write $c_1 + c_2$ as shorthand notation for $\mathsf{Eval}(\mathsf{ek}, f(x, y) := x + y, (c_1, c_2))$ and $\alpha \cdot c$ for $\alpha \in \mathcal{M}$ as shorthand notation for $\mathsf{Eval}(\mathsf{ek}, f_\alpha(x) := \alpha \cdot x, c)$, where we assume that ek is always clear from the context. Furthermore, we assume that the ciphertext size is fixed and does not increase, when homomorphic operations are performed.

Definition 3 (IND-CPA). *Let* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a homomorphic encryption scheme with message space* $\mathcal{M}$*. We say* E *is indistinguishable under chosen message attacks, if for any PPT adversary* $\mathcal{A}$ *and any* $\lambda \in \mathbb{N}$*, it holds that*

$$\Pr\left[b = b' : \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek}) \\ b \leftarrow \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{ct}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.2 Oblivious Ciphertext Compression

Now let us define oblivious ciphertext compression. In this setting, we have a compressor and a decompressor. The compressor is given an encrypted vector $\mathbf{ct}$ of length n of bounded hamming weight and is asked to compress the vector. The decompressor holds the decryption key, is given the compressed vector, and is asked to output a sparse[7] representation of the original plaintext vector.

The setting we consider is more general than that of Bienstock et al. [BPSY23], but for the sake of simplicity less general than that of Fleischhacker, Larsen, and Simkin [FLS23]. In comparison to Bienstock et al., we do not provide the decompressor with the indices of the non-zero vector entries. In comparison to Fleischhacker, Larsen, and Simkin, we formally consider the task of compressing fresh encryptions, rather than encryptions that are the output of possible previous homomorphic evaluation. We note, however, that all of our results trivially generalize to the formal setting of their work as well.

Definition 4 (Oblivious Ciphertext Compression). *Let $n, t \in \mathbb{N}$, let $\mathcal{M}$ be a set containing zero, and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a homomorphic encryption scheme with message space $\mathcal{M}$ and ciphertext bit length $\xi = \xi(\lambda)$. A pair of PPT algorithms $(\mathsf{OblivCompress}, \mathsf{Decompress})$ is a ciphertext compression scheme for E for message space $\mathcal{P}_t$ with error ϵ and compression rate δ, if for any message vector $\boldsymbol{m} \in \mathcal{P}_t$ and any $\lambda \in \mathbb{N}$, it holds that*

$$\Pr\left[\mathsf{sparse}(\boldsymbol{m}) = \boldsymbol{m}' \wedge |\mathbf{ct}'| \leq \delta \cdot \xi \cdot n \;:\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}' \leftarrow \mathsf{OblivCompress}(\mathsf{ek}, \mathbf{ct}) \\ \boldsymbol{m}' \leftarrow \mathsf{Decompress}(\mathsf{dk}, \mathbf{ct}') \end{array}\right] \geq 1 - \epsilon,$$

where the probability is taken over the random coins of all involved algorithms.

Below we provide a definition for the setting that was considered by Bienstock et al. [BPSY23]. In it, we provide the decompressor with a set I that contains the non-zero indices.

Definition 5 (Oblivious Ciphertext Compression with Auxiliary Input). *Let $n, t \in \mathbb{N}$, let $\mathcal{M}$ be a set containing zero, and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a homomorphic encryption scheme with message space $\mathcal{M}$ and ciphertext bit length $\xi = \xi(\lambda)$. A pair of PPT algorithms $(\mathsf{OblivCompressAux}, \mathsf{DecompressAux})$ is an auxiliary-input ciphertext compression scheme for E for message space $\mathcal{P}_t$ with error ϵ and compression rate δ, if for any message vector $\boldsymbol{m} \in \mathcal{P}_t$ with $I = \{i \in [n] \mid \boldsymbol{m}_i \neq 0\}$ and any $\lambda \in \mathbb{N}$, it holds that*

$$\Pr\left[\mathsf{sparse}(\boldsymbol{m}) = \boldsymbol{m}' \wedge |\mathbf{ct}'| \leq \delta \cdot \xi \cdot n \;:\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}' \leftarrow \mathsf{OblivCompressAux}(\mathsf{ek}, \mathbf{ct}) \\ \boldsymbol{m}' \leftarrow \mathsf{DecompressAux}(\mathsf{dk}, \mathbf{ct}', I) \end{array}\right] \geq 1 - \epsilon,$$

where the probability is taken over the random coins of all involved algorithms.

2.3 Oblivious Ciphertext Decompression

In the oblivious ciphertext decompression setting, the roles of compressor and decompressor are effectively switched. The compressor is given the sparse representation of a plaintext vector $\boldsymbol{m}$ of length n of bounded hamming weight and is asked to produce a succinct encoding ct of this vector. The encoding should not reveal the locations of the non-zero positions, but should still allow the decompressor to expand the succinct

[7] By only asking the decompressor to output a sparse representation, we enable them to have a computational complexity that is dependent on the maximum number of non-zero entries and in particular sub-linear in the length of the compressed vector.

encoding obliviously into an encrypted "relaxed" dense representation. That is, all non-zero entries in the sparse representation should be correctly expanded in the dense representation, but all other entries in the expanded vector can be arbitrary values and do not need to be zero.

Definition 6 (Oblivious Ciphertext Decompression). *Let $n, t \in \mathbb{N}$, let $\mathcal{M}$ be a set containing zero, and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a homomorphic encryption scheme with message space $\mathcal{M}$ and ciphertext bit length $\xi = \xi(\lambda)$. A pair of PPT algorithms $(\mathsf{Compress}, \mathsf{OblivDecompress})$ is a ciphertext decompression scheme for E for message space $\mathcal{P}_t$ with error ϵ and compression rate δ, if for any message vector $\boldsymbol{m} \in \mathcal{P}_t$ and any $\lambda \in \mathbb{N}$, it holds that*

$$\Pr\left[\begin{array}{r|r} & (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m_i = 0 \vee m_i = m'_i)\ \forall\ i \in [n] \wedge & \mathbf{ct} \leftarrow \mathsf{Compress}(\mathsf{ek}, \mathsf{sparse}(\boldsymbol{m})) \\ |\mathbf{ct}| \leq \delta \cdot \xi \cdot n & \mathbf{ct}' \leftarrow \mathsf{OblivDecompress}(\mathsf{ek}, \mathbf{ct}) \\ & \boldsymbol{m}' \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathbf{ct}') \end{array}\right] \geq 1 - \epsilon,$$

and for any PPT adversary $\mathcal{A}$, it holds that

$$\Pr\left[\begin{array}{r|r} & (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ & \boldsymbol{m}_0, \boldsymbol{m}_1 \leftarrow \mathcal{A}(\mathsf{ek}, \mathsf{dk}) \\ b = b' \wedge \boldsymbol{m}_0 \in \mathcal{P}_t \wedge \boldsymbol{m}_1 \in \mathcal{P}_t : & b \leftarrow \{0,1\} \\ & \mathbf{ct} \leftarrow \mathsf{Compress}(\mathsf{ek}, \mathsf{sparse}(\boldsymbol{m}_b)) \\ & b' \leftarrow \mathcal{A}(\mathbf{ct}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

3 Linear Codes

In this section, we first recall some standard terminology from coding theory that we will use throughout the paper. In particular, we will properly introduce the *Syndrome Decoding* and *Syndrome Decoding with Known Support* problems that will be the key ingredient for designing efficient oblivious ciphertext compression/decompression schemes. Then, we will study these problems for the case of the well known *Generalized Reed-Solomon* code where our aim is to provide a thorough complexity analysis.

Definition 7 (Linear Codes). *Let $n, k, q \in \mathbb{N}$. A linear $[n, k, d]$ code over $\mathbb{F}_q$ is an injective mapping $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$. We call k the message length, n the block length, and any element in the image of $\mathcal{C}$ a codeword. The minimal distance d is the smallest number of non-zero entries in any non-zero $c \in \mathcal{C}$.*

Remark 1. We will sometimes abuse notation and write $\mathcal{C}$ to refer to the set of all possible codewords, rather than the injective mapping itself.

Definition 8 (Generator/Parity Check Matrices). *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. A matrix $G \in \mathbb{F}^{k \times n}$ is said to be a generator matrix of $\mathcal{C}$, if for all $\boldsymbol{m} \in \mathbb{F}_q^k$, it holds that $\boldsymbol{m}^T \cdot G = \mathcal{C}(\boldsymbol{m})$. A matrix $H \in \mathbb{F}^{(n-k) \times n}$ is said to be a parity-check matrix of $\mathcal{C}$, if $\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n \mid H \cdot \boldsymbol{c} = 0\}$.*

Proposition 9 (Singleton bound). *Let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. Then $d \leq n - k + 1$.*

Definition 10 (Maximum Distance Separable Codes). *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. The code $\mathcal{C}$ is said to be maximum distance separable (MDS), if $d = n - k + 1$.*

For the purpose of compressing encrypted data, we will need to use the concept of efficient syndrome decoding. Let $\boldsymbol{c}$ be a codeword and let $\boldsymbol{e}$ be a sparse noise vector. In the classical error correction setting, we aim to recover $\boldsymbol{c}$, when given $\boldsymbol{c} + \boldsymbol{e}$. We will say that a code is syndrome decodable, when $\boldsymbol{e}$ can be recovered from H and $H \cdot (\boldsymbol{c} + \boldsymbol{e})$ alone, without access to $\boldsymbol{c} + \boldsymbol{e}$ itself.

Definition 11 (Syndrome Decoding). *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. Let $G \in \mathbb{F}^{k \times n}$ be a generator and $H \in \mathbb{F}^{(n-k) \times n}$ a parity-check matrix of $\mathcal{C}$. The code $\mathcal{C}$ is said to be efficiently syndrome decodable up to error t, if there exists a $\mathsf{poly}(n-k)$-time algorithm, which for any $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ of hamming weight at most t (i.e. $\mathsf{hw}(\boldsymbol{e}) \leq t$) recovers $\mathsf{sparse}(\boldsymbol{e})$ from H and $H \cdot (\boldsymbol{c} + \boldsymbol{e})$.*

Definition 12 (Syndrome Decoding with Known Support). *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. Let $G \in \mathbb{F}^{k \times n}$ be a generator and $H \in \mathbb{F}^{(n-k) \times n}$ a parity-check matrix of $\mathcal{C}$. The code $\mathcal{C}$ is said to be efficiently syndrome decodable from known support up to error t, if there exists a $\mathsf{poly}(n-k)$-time algorithm, which for any $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ of hamming weight at most t recovers $\mathsf{sparse}(\boldsymbol{e})$ from H, $H \cdot (\boldsymbol{c} + \boldsymbol{e})$, and $I = \{i \in [n] \mid e_i \neq 0\}$.*

Now for the purpose of encrypted decompression, we observe that we can, in spirit, do the opposite of syndrome decoding. Given a long vector $\boldsymbol{p} \in \mathbb{F}^n$ and some indices of interest $I \subset [n]$, we can compute a short vector $\boldsymbol{m}$, such that for $\boldsymbol{c} = \boldsymbol{m} \cdot H$, we have that $\boldsymbol{p}$ and $\boldsymbol{c}$ agree on their values in all positions in I. In other words, we can program a short seed $\boldsymbol{m}$ that can be expanded, via a matrix vector multiplication, into a long vector $\boldsymbol{c}$, such that some entries in $\boldsymbol{c}$ take on the desired programmed values. Note that this can be interpreted as erasure correction in the dual code.

Lemma 13 ([Rot06, Theorem 2.2]). *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C} \neq \{\mathbf{0}\}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$ and let $H \in \mathbb{F}_q^{(n-k) \times n}$ be a parity-check matrix of $\mathcal{C}$. The minimum distance of $\mathcal{C}$ is the largest integer d, such that every set of $d-1$ columns in H is linearly independent.*

Corollary 14. *Let $n, k, q \in \mathbb{N}$ and let $\mathcal{C} \neq \{\mathbf{0}\}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$ and let $H \in \mathbb{F}_q^{(n-k) \times n}$ be a parity-check matrix of $\mathcal{C}$. For any $\boldsymbol{p} \in \mathbb{F}_q^n$ and any subset $I \subset [n]$ of size at most $d-1$, one can efficiently compute a vector $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ satisfies $c_i = p_i$ for all $i \in I$.*

Proof. Consider the vector $\boldsymbol{p}_I = (p_{i_1}, \ldots, p_{i_t})$ where $I = \{i_1, \ldots, i_t\}$, and H_I be the submatrix of H comprised of columns $i_1, \ldots, i_t$ of H. Since $\mathcal{C}$ has minimal distance d, Lemma 13 implies that H_I is full rank. Therefore, the system $\boldsymbol{m} \cdot H_I = \boldsymbol{p}_I$ has $t \leq d-1$ linearly independent equations in $n - k \geq d - 1 \geq t$ variables, hence has at least one solution $\boldsymbol{m}$. □

3.1 Reed-Solomon Codes

In this section, we present (generalized) Reed-Solomon codes, and we show they are indeed very efficient *syndrome decodable* codes, by giving a precise complexity analysis.

Definition 15 ([Rot06]). *Let n, k, $q \in \mathbb{N}$, and $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q^\times$ pairwise distinct. The* normalized generalized Reed-Solomon code *(normalized GRS code) $\mathcal{C}$ over $\mathbb{F}_q$ with* code locators *$\alpha_1, \ldots, \alpha_n$ is the linear $[n, k, n-k+1]$ code over $\mathbb{F}_q$ defined as $\mathcal{C}(\boldsymbol{m}) = (v_1 m(\alpha_1), \ldots, v_n m(\alpha_n))$ where $m(\alpha_j) = \sum_{i=1}^{k} m_i \alpha_j^{i-1}$ and $v_j = \prod_{i \neq j} \frac{1}{\alpha_j - \alpha_i}$, for $1 \leq j \leq n$. A parity-check matrix of $\mathcal{C}$ is*

$$H = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \ldots & \alpha_n^{n-k-1} \end{pmatrix}.$$

Lemma 16. *Let $H \in \mathbb{F}_q^{(n-k) \times n}$ be the parity-check matrix of a normalized GRS code $[n, k, d]$. Given $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ and $\boldsymbol{p} \in \mathbb{F}_q^n$, the vector-matrix and matrix-vector products $\boldsymbol{m} \cdot H$ and $H \cdot \boldsymbol{p}$ can be computed in $\mathcal{O}\Big(\frac{n}{n-k} \mathsf{M}(n-k) \log(n-k)\Big)$ finite field operations. These operations are linear in the inputs $\boldsymbol{m}$ and $\boldsymbol{p}$.*

If the code locators $\alpha_1, \ldots, \alpha_n$ are assumed to be in geometric progression, the complexity is reduced to $\frac{n}{n-k}\left(\mathsf{M}(n-k) + \mathcal{O}(n-k)\right)$.

Proof. Write $(\boldsymbol{m} \cdot H)_i = \sum_{j=1}^{n-k} m_j \alpha_i^{j-1} = m(\alpha_i)$. This vector-matrix product corresponds to the evaluation of a degree-$(n-k-1)$ polynomial on the n code locators. Since we can process code locators by bunch of $(n-k)$ values, this can be computed with $\lceil \frac{n}{n-k} \rceil$ multi-point evaluations on $(n-k)$ points using $\mathcal{O}\Big(\frac{n}{n-k}\mathsf{M}(n-k)\Big)$ finite field operations [vzGG13]. When the code locators are in geometric progression, we can use the optimized multi-point evaluation of [BS05], yielding the claimed complexity. The *transposition principle* [BLS03] implies that computing $H \cdot \boldsymbol{p}$ has exactly the same cost. □

Lemma 17. *Let $H \in \mathbb{F}_q^{(n-k)\times n}$ be the parity-check matrix of a normalized GRS code $[n,k,d]$. The two following problems can be solved in $\mathcal{O}(\mathsf{M}(n-k)\log(n-k))$ finite field operations:*

- *given $\boldsymbol{p} \in \mathbb{F}_q^n$ and any subset $I \subset [n]$ of size at most $d-1 = n-k$, compute a vector $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ satisfies $c_i = p_i$ for all $i \in I$;*
- *given $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ and any subset $I \subset [n]$ of size at most $n-k$, compute a vector $\boldsymbol{p} \in \mathbb{F}_q^n$ such that $\boldsymbol{m} = H \cdot \boldsymbol{p}$ and $p_i = 0$ for $i \notin I$.*

Proof. Without loss of generality, we assume that I has size $n-k$. Otherwise, one can replace I by a superset of the required size. Using the same notations as in Corollary 14, the goal of the first problem is to compute $\boldsymbol{m}$ such that $\boldsymbol{m} \cdot H_I = \boldsymbol{p}_I$. As in the proof of Lemma 16, $\boldsymbol{m} \cdot H_I$ corresponds to multi-point evaluation. The converse problem, computing $\boldsymbol{m}$ from H_I and $\boldsymbol{p}_I$, is polynomial interpolation. The vector $\boldsymbol{m}$ can thus be computed in $\mathcal{O}(\mathsf{M}(n-k)\log(n-k))$ finite field operations [vzGG13]. The second problem has the same complexity according to the *transposition principle* [BLS03]. □

Theorem 18 (Decoding complexity of GRS Codes). *Let n, $k \in \mathbb{N}$, let $\mathcal{C}$ be a normalized GRS code $[n,k,d]$ over $\mathbb{F}_q$ with code locators $\alpha_1, \ldots, \alpha_n$. Then,*

(i) If the map $\alpha_i \mapsto i$ is efficiently computable, $\mathcal{C}$ is efficiently syndrome decodable up to $\lfloor \frac{n-k}{2} \rfloor$ errors in $\mathcal{O}(\mathsf{M}(n-k)\log(n-k)\log q)$ finite field operations, and

(ii) $\mathcal{C}$ is efficiently syndrome decodable from known support up to $(n-k)$ errors in $\mathcal{O}(\mathsf{M}(n-k)\log(n-k))$ finite field operations.

Proof. The algorithm is an adaptation of Berlekamp-Massey algorithm as described by Roth [Rot06]. The main difference lies in the fact that the complexity is quasi-linear in the syndrome size $(n-k)$ rather than the code length n.

Let $\boldsymbol{c} \in \mathcal{C}$, $\boldsymbol{e} \in \mathbb{F}_q^n$ of hamming weight at most $t \leq \frac{n-k}{2}$ and $\boldsymbol{s} = H \cdot (\boldsymbol{c} + \boldsymbol{e})$ be the syndrome where H is the parity-check matrix of $\mathcal{C}$. Let $I = \{i \in [n] \mid e_i \neq 0\}$. The starting point of Berlekamp-Massey algorithm is the remark that the syndrome $\boldsymbol{s}$ satisfies a linear recurrence with minimal polynomial $\Lambda = \prod_{i \in I}(x - \alpha_i)$. The polynomial Λ can be computed from the $(n-k)$ terms $s_1, \ldots, s_{n-k}$ by means of Padé approximation in $\mathcal{O}(\mathsf{M}(t)\log t)$ finite field operations [vzGG13, Chapter 12]. The roots of Λ can be computed using Berlekamp-Rabin's algorithm in $\mathcal{O}(\mathsf{M}(t)\log(t)\log q)$ finite field operations [Ber70, Rab80]. This provides the set $\{\alpha_i : i \in I\}$. Since the map $\alpha_i \mapsto i$ is efficiently computable, this also provides the set I. Finally, computing the error values from I can be done in $\mathcal{O}(\mathsf{M}(t)\log(t))$ finite field operations according to Lemma 17.

This proves the first item. For the second item, we assume that $I = \{i_1, \ldots, i_t\}$ is known. Only the last step is required, whence the complexity. □

Theorem 18*(i)* requires an efficiently computable mapping $\alpha_i \mapsto i$. If the characteristic of $\mathbb{F}_q$ is large enough, we can choose $\alpha_i = i$. In addition to being efficiently computable, this is an arithmetic progression and we can further improve the complexity statement of Lemma 16 by a constant factor to get $\frac{n}{n-k}\left(\mathsf{M}(n-k)\log(n-k) + \mathcal{O}(n-k)\right)$ [BS05]. For smaller characteristics, say when $q = p^k$ for some small p, elements of $\mathbb{F}_q$ are polynomials over $\mathbb{F}_p$ of degree $< k$. We can choose $\alpha_i = \sum_{j=0}^{k-1} i_j x^j$ where $i = \sum_{j=0}^{k-1} i_j p^j$ is the base-p representation of i.

Remark 2. Classical decoding algorithms for GRS codes such as Berlekamp-Massey algorithm have no assumption on the code locators. To get the best complexity in Lemma 16, a standard choice is to use a

geometric progression. But the map $\alpha_i \mapsto i$ becomes a discrete logarithm and is not efficiently computable (except in some very specific finite fields). For standard decoding algorithms that first compute the syndrome from a noisy codeword in quasi-linear time in n, the cost of the discrete logarithm is anyway dominated by the syndrome computation. In our settings where the input is the syndrome, this is not true anymore and the discrete logarithm becomes the dominant term. Although it is phrased in the vocabulary of sparse polynomial interpolation, this is exactly the solution proposed in [FLS23, Section 4] with complexity $\mathcal{O}(\sqrt{n})$ due to the discrete logarithms computations.

3.2 Reed-Solomon Codes over LCH polynomial basis

In the following, we provide an alternative definition of Reed-Solomon codes when the plaintext space is the binary extension field $\mathbb{F}_{2^\ell}$. In such a case, it does exist a definition that allows for faster encoding and decoding. The main idea, introduced in the work of Lin et al. [LCH14, LANH16], is to replace the polynomial basis in the GRS code, i.e. the monomial one, with a more involved one for which the polynomial arithmetic is more efficient. In particular, the new basis is compliant for designing fast additive FFT over affine subspace of $\mathbb{F}_{2^\ell}$, allowing to design fast Reed-Solomon erasure codes [LCH14]. It has been further demonstrated that Reed-Solomon codes over the new basis can also correct errors efficiently: [LANH16] for powers of two syndrome's sizes, and its natural extension to the general case using the truncated evaluation/interpolation over LCH basis from [Cox21].

Our aim is to show that such codes are also efficient *syndrome decodable* codes, yielding a better complexity estimate than the more classical GRS codes.

We start by introducing the LCH polynomial basis of [LCH14], and then provide the necessary materials to provide efficient *syndrome decodable* GRS codes over the LCH basis.

Definition 19 (LCH polynomial basis [LCH14]). *Let $q = 2^\ell$. Let $\bar{v} = (\boldsymbol{v_0}, \ldots, \boldsymbol{v_{\ell-1}})$ denote a basis of $\mathbb{F}_q$ as an $\mathbb{F}_2$-vector space and write $\mathbb{F}_q = \{\omega_0, \ldots, \omega_{2^\ell-1}\}$ where for $0 \le i < 2^\ell$, $\omega_i = i_0 \cdot \boldsymbol{v_0} + \cdots + i_{\ell-1} \cdot \boldsymbol{v_{\ell-1}}$ and i_k denotes the kth bit in the binary representation of i. Let*

$$X_i(x) = \prod_{k=0}^{\ell-1} \left(\prod_{j=0}^{2^k-1} (x - \omega_j) \right)^{i_k}$$

for $0 \le i < 2^\ell$. The set $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell-1}(x)\}$ is the LCH basis *of $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. Each $X_i(x)$ satisfies $\deg X_i(x) = i$.*

For simplicity, one can consider $\bar{v} = (\boldsymbol{e_0}, \ldots, \boldsymbol{e_{\ell-1}})$ in the LCH basis, where $\boldsymbol{e_i} \in \mathbb{F}_2^\ell$ are the canonical vectors, i.e. $\forall j \neq i, \boldsymbol{e_i}[j] = 0$ and $\boldsymbol{e_i}[i] = 1$. The following proposition gathers many complexity results on the computation with polynomials given in the LCH basis.

Proposition 20 ([LCH14, LANH16, Cox21]). *Let $q = 2^\ell$. For any polynomials $f, g \in \mathbb{F}_q[x]$ of degree $d < 2^\ell$, given by the list of their coefficients in the LCH basis, one can compute*

- *the formal derivative of f in LCH basis in time $\mathcal{O}(d \log d)$ operations over $\mathbb{F}_q$,*
- *the product $f \times g$ in LCH basis in time $\mathcal{O}(d \log d)$ operations over $\mathbb{F}_q$,*
- *the quotient and the remainder in the LCH basis of f divided by g in time $\mathcal{O}(d \log d)$ operations over $\mathbb{F}_q$,*
- *the gcd of f and g in the LCH basis, together with its Bézout coefficients, in time $\mathcal{O}\big(d \log^2 d\big)$ operations over $\mathbb{F}_q$.*

We now provide a few other complexity results on the LCH basis that will be key ingredients in the syndrome decoding approach over LCH basis.

Corollary 21. *Let $q = 2^\ell$ and $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell-1}(x)\}$ be an LCH basis of $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. Let $(\alpha_1, \ldots, \alpha_n)$ be pairwise distinct elements of $\mathbb{F}_q$, and $f(x) = \sum_{i=0}^{d-1} f_i \cdot X_i(x)$, with $d \le n < 2^\ell$.*

– (multi-point evaluation in LCH basis) *The computation of* $(f(\alpha_1), \ldots, f(\alpha_n))$ *from the coefficients* $(f_0, \ldots, f_{d-1})$ *and the points* $(\alpha_1, \ldots, \alpha_n)$ *costs* $\mathcal{O}\big(n \log^2 d\big)$ *operations in* $\mathbb{F}_q$.
– (polynomial interpolation in LCH basis) *The computation of* $(f_0, \ldots, f_{d-1})$ *from the values* $f(\alpha_1), \ldots, f(\alpha_n)$ *and the points* $(\alpha_1, \ldots, \alpha_n)$ *costs* $\mathcal{O}\big(n \log^2 d\big)$ *operations in* $\mathbb{F}_q$.
– (root finding in the LCH basis) *Assuming that* $f = \prod_{i=1}^{d}(x - \beta_i)$ *with pairwise distinct* $\beta_i \in \mathbb{F}_q$, *the computation of* $(\beta_1, \ldots, \beta_d)$ *from the coefficients* $(f_0, \ldots, f_{d-1})$ *costs* $\mathcal{O}\big(d \log^2 d \cdot \log q\big)$ *operations in* $\mathbb{F}_q$.

If $\alpha_1, \ldots, \alpha_n$ *lie in a same affine subspace of* $\mathbb{F}_q$ *of dimension* $\mathcal{O}(\log n)$, *the complexity of multi-point evaluation and interpolation is reduced to* $\mathcal{O}(n \log d)$.

Proof. If the α_i's lie in an affine subspace $E + \omega \subset \mathbb{F}_q$, with E is a linear subspace of $\mathbb{F}_q$ and $\omega \in \mathbb{F}_q$, the multi-point evaluation problem over LCH basis is the additive FFT problem of [LCH14] that is solved in $\mathcal{O}(n \log d)$. A similar result is achieved for interpolation using the inverse FFT algorithm of [LCH14].

If there is no assumption on the α_i's, one can directly re-use the fast classical divide-and-conquer approaches for multi-point evaluation and for interpolation over the monomial basis [vzGG13, Sections 10.1 and 10.2]. These algorithms work *mutatis mutandis* over LCH basis. Indeed, by choosing $\bar{v} = \{\boldsymbol{e_1}, \ldots, \boldsymbol{e_\ell}\}$ in the LCH-basis, we know that $X_1(x) = x$ and $X_0(x) = 1$. Then, we can perform $\lceil \frac{n}{d} \rceil$ multi-point evaluations with only d points at a time, so that each multi-point evaluation problem is regular, i.e. d points with a degree-$d-1$ polynomial. For the γth regular problem, we can therefore build the sub-product tree of $\prod_{j=1}^{d}(X_1(x) - \alpha_{\gamma \cdot d + j})$ directly in the LCH basis. The fast multi-point evaluation algorithm [vzGG13, Section 10.1] uses this sub-product tree to perform several divisions starting from the input polynomial $f(x) = \sum_{i=1}^{d-1} f_i \cdot X_i(x)$. Since all the operations are performed in the LCH basis, this approach requires $\lceil \frac{n}{d} \rceil \mathcal{O}\big(d \log^2 d\big) = \mathcal{O}\big(n \log^2 d\big)$ operations in $\mathbb{F}_q$ by Proposition 20. A similar remark applies for interpolation in the LCH basis, using the efficient formal derivative algorithm in the LCH basis of Proposition 20.

Finally, to prove the result for the root finding problem, one may remark that the algorithm of Berlekamp-Rabin [Ber70, Rab80] designed for binary extension field work *mutatis mutandis* over an LCH basis. The complexity statement is achieved through the use of Proposition 20. □

Given an LCH basis for $\mathbb{F}_q[x]/(x^{2^\ell} - x)$, we can define a Reed-Solomon code over an LCH basis, in a very similar manner as Definition 15. These codes, or more precisely their dual codes, have been introduced by Lin et al. [LCH14] and their decoding is studied by Lin et al. [LANH16].

Definition 22 (normalized GRS code in LCH basis [LCH14]). *Let* $q = 2^\ell$ *and* $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell - 1}(x)\}$ *be an LCH basis of* $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. *Let* $n, k \in \mathbb{N}$, *and* $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q^\times$ *be pairwise distinct. The* normalized GRS code in LCH basis *(normalized GRS-LCH code) over* $\mathbb{F}_q$ *with* code locators $\alpha_1, \ldots, \alpha_n$ *is the linear* $[n, k, n-k+1]$ *code over* $\mathbb{F}_q$ *defined as* $\mathcal{C}(\boldsymbol{m}) = (v_1 m(\alpha_1), \ldots, v_n m(\alpha_n))$ *where* $m(\alpha_j) = \sum_{i=0}^{n-k-1} m_i \cdot X_i(\alpha_j)$ *and* $v_j = \prod_{i \neq j} \frac{1}{\alpha_j - \alpha_i}$, *for* $1 \leq j \leq n$. *A parity-check matrix of* $\mathcal{C}$ *is*

$$H = \begin{pmatrix} X_0(\alpha_1) & X_0(\alpha_2) & \ldots & X_0(\alpha_n) \\ X_1(\alpha_1) & X_1(\alpha_2) & \ldots & X_1(\alpha_n) \\ \vdots & \vdots & & \vdots \\ X_{n-k-1}(\alpha_1) & X_{n-k-1}(\alpha_2) & \ldots & X_{n-k-1}(\alpha_n) \end{pmatrix}.$$

Lemma 23. *Let* $q = 2^\ell$ *and* $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell - 1}(x)\}$ *be an LCH basis of* $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. *Let* $H \in \mathbb{F}_q^{(n-k) \times n}$ *be the parity-check matrix of a normalized GRS-LCH code* $\mathcal{C}$ *of parameters* $[n, k, d]$. *Given* $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ *and* $\boldsymbol{p} \in \mathbb{F}_q^n$, *the vector-matrix and matrix-vector products* $\boldsymbol{m} \cdot H$ *and* $H \cdot \boldsymbol{p}$ *can be computed in* $\mathcal{O}\big(n \log^2(n-k)\big)$ *finite field operations. These operations are linear in the inputs* $\boldsymbol{m}$ *and* $\boldsymbol{p}$.

The complexity becomes $\mathcal{O}(n \log(n-k))$ *if the code locators* $\alpha_1, \ldots, \alpha_n$ *lie in an affine subspace of* $\mathbb{F}_q$.

Proof. According to Definition 22, computing $\boldsymbol{m} \cdot H$ corresponds to multi-point evaluation in LCH basis. The complexity is then deduced from Corollary 21 (multi-point evaluation). Computing $H \cdot \boldsymbol{p}$ has the same complexity according to the *transposition principle* [BLS03]. □

Lemma 24. *Let $q = 2^\ell$ and $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell-1}(x)\}$ be an LCH basis of $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. Let $H \in \mathbb{F}_q^{(n-k)\times n}$ be the parity-check matrix of a normalized LCH-GRS code $\mathcal{C}$ of parameters $[n, k, d]$. The two following problems can be solved in $\mathcal{O}\big((n-k)\log^2(n-k)\big)$ finite field operations:*

- *given $\boldsymbol{p} \in \mathbb{F}_q^n$ and any subset $I \subset [n]$ of size at most $d - 1 = n - k$, compute a vector $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ satisfies $c_i = p_i$ for all $i \in I$;*
- *given $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ and any subset $I \subset [n]$ of size at most $n - k$, compute a vector $\boldsymbol{p} \in \mathbb{F}_q^n$ such that $\boldsymbol{m} = H \cdot \boldsymbol{p}$ and $p_i = 0$ for $i \notin I$.*

If we further assume that we have a map that sends $I \subset n$ to a linear subspace of $\mathbb{F}_q$, the complexity becomes $\mathcal{O}((n-k)\log(n-k))$ finite field operations.

Proof. The proof is a direct consequence of Corollary 14, Corollary 21 (polynomial interpolation) and the *transposition principle* [BLS03]. □

Theorem 25 (Decoding complexity of normalized GRS-LCH codes). *Let $q = 2^\ell$ and $\mathbb{X} = \{X_0(x), \ldots, X_{2^\ell-1}(x)\}$ be an LCH basis of $\mathbb{F}_q[x]/(x^{2^\ell} - x)$. Let n, $k \in \mathbb{N}$, let $\mathcal{C}$ be a normalized GRS-LCH code $[n, k, d]$ over $\mathbb{F}_q$ with code locators $\alpha_1, \ldots, \alpha_n$. Then,*

i. If the map $\alpha_i \mapsto i$ is efficiently computable, $\mathcal{C}$ is efficiently syndrome decodable up to $\lfloor \frac{n-k}{2} \rfloor$ errors in $\mathcal{O}\big((n-k)\log^2(n-k)\log q\big)$ finite field operations, and

ii. $\mathcal{C}$ is efficiently syndrome decodable from known support up to $(n-k)$ errors in $\mathcal{O}\big((n-k)\log^2(n-k)\big)$ finite field operations.

Proof. The proof for (i) follows the same blueprint as the decoding algorithm of [LANH16, Section 7], in which the code locators are chosen so that $\alpha_i = \sum_{j=0}^{\ell-1} i_j x^j$ where $i = \sum_{j=0}^{\ell-1} i_j 2^j$ is the binary representation of i.

Although their decoding algorithm is for the dual code of $\mathcal{C}$, it is easily adapted to $\mathcal{C}$ that has the same structure. The first step in their algorithm is the computation of the syndrome, which is the input of our problem. We are given $\boldsymbol{s} = H \cdot (\boldsymbol{c} + \boldsymbol{e}) = H \cdot \boldsymbol{e}$ where $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ satisfies $\mathsf{hw}(\boldsymbol{e}) \leq t \leq \frac{n-k}{2}$. As in [LANH16], the error-locator polynomial $\lambda(x)$ can be computed using a polynomial division and an extended gcd algorithm in LCH basis involving the polynomial associated to $\boldsymbol{s}$, which is of degree less than $n-k$, for a total cost of $\mathcal{O}\big((n-k)\log^2(n-k)\big)$ operations over $\mathbb{F}_q$. The next step is to compute the roots of $\lambda(x)$, which are the code locators α_i for which e_i is non-zero. Instead of relying on FFT to evaluate λ on the n code locators (for a cost linear in n), we use the root finding algorithm described in Corollary 21, using $\mathcal{O}\big((n-k)\log^2(n-k)\log q\big)$ operations in $\mathbb{F}_q$. From the roots, we can obtain easily the set of error locations $I \subset [n]$ of size at most t, since it is assumed that the map $\alpha_i \mapsto i$ is efficiently computable.

From this set, the error values $\{e_i \mid i \in I\}$ can be computed using the variant of Forney's algorithm [For65] proposed in [LANH16, Section 7]. Again, instead of evaluating the required polynomials at all code locators, we evaluate them only on the t meaningful values, i.e. $\{\alpha_i \mid i \in I\}$. According to Corollary 21 (multi-point evaluation) this requires $\mathcal{O}\big((n-k)\log^2(n-k)\big)$ operations in $\mathbb{F}_q$. The overall process is dominated by root finding. This concludes the proof for (i). The proof for (ii) correspond exactly to the last step of (i) when the set I is known. □

Remark 3. Theorem 25*(i)* requires an efficiently computable mapping $\alpha_i \mapsto i$. To define the LCH basis, we can use the canonical basis $(\boldsymbol{e_0}, \ldots, \boldsymbol{e_{\ell-1}})$, and choose $\alpha_i = \sum_{j=0}^{\ell-1} i_j \boldsymbol{e_j}$ where $\sum_{j=0}^{\ell-1} i_j 2^j$ is the binary representation of i. This also provides the best known complexities for operations in the LCH basis.

4 Compression via Linear Codes

In this section, we show that efficiently syndrome decodable linear codes allow for efficient ciphertext compression. By plugging in various syndrome decodable codes, we then obtain ciphertext compression schemes with different trade-offs.

$\mathsf{OblivCompress}(\mathsf{ek}, \mathbf{ct})$	$\mathsf{Decompress}(\mathsf{dk}, \mathbf{ct})$
$f(\boldsymbol{c}) := H \cdot \boldsymbol{c}$	$\boldsymbol{s} \leftarrow \mathsf{E.Dec}(\mathsf{dk}, \mathbf{ct})$
$\mathbf{ct}' \leftarrow \mathsf{E.Eval}(\mathsf{ek}, f, \mathbf{ct})$	$\boldsymbol{m} \leftarrow \mathcal{C}.\mathsf{Decode}(\boldsymbol{s})$
return $\mathbf{ct}'$	**return** $\boldsymbol{m}$

Fig. 1. Oblivious ciphertext compression from linear codes. E is a homomorphic encryption scheme and $\mathcal{C}$ is a linear code with parity-check matrix H.

Theorem 26. *Let $n, t, k, q, \lambda \in \mathbb{N}$, let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$ that is efficiently syndrome decodable up to t errors and let $H \in \mathbb{F}^{(n-k)\times n}$ be a parity-check matrix for the code $\mathcal{C}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$ and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Then the construction in Figure 1 is an oblivious ciphertext compression scheme for E for message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = 1 - \frac{k}{n}$.*

Proof. Let $\boldsymbol{m} \in \mathcal{P}_t$ be an arbitrary plaintext vector. We start by observing that compression only requires an additively homomorphic encryption scheme, since computing $\mathbf{ct}'$ only requires homomorphic additions and multiplications by constants. Next, we observe that

$$
\begin{aligned}
& \Pr\left[\mathsf{sparse}(\boldsymbol{m}) \neq \boldsymbol{m}' : \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}' \leftarrow \mathsf{OblivCompress}(\mathsf{ek}, \mathbf{ct}) \\ \boldsymbol{m}' \leftarrow \mathsf{Decompress}(\mathsf{dk}, \mathbf{ct}') \end{array}\right] \\
= \; & \Pr\left[\mathsf{sparse}(\boldsymbol{m}) \neq \boldsymbol{m}' : \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}' \leftarrow \mathsf{Eval}(\mathsf{ek}, f, \mathbf{ct}) \\ \boldsymbol{s} \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathbf{ct}') \\ \boldsymbol{m}' \leftarrow \mathsf{Decode}(\boldsymbol{s}) \end{array}\right] \\
\leq \; & \Pr\left[\mathsf{sparse}(\boldsymbol{m}) \neq \boldsymbol{m}' : \boldsymbol{m}' \leftarrow \mathsf{Decode}(f(\boldsymbol{m}))\right] + \mathsf{negl}(\lambda) \\
= \; & \mathsf{negl}(\lambda)
\end{aligned}
$$

where the second to last inequality follows from the correctness of the encryption scheme and the last one follows from the fact that the code is syndrome decodable up to t errors and the fact that $\boldsymbol{m} \in \mathcal{P}_t$. Finally, we observe that $|\mathbf{ct}'| = \xi \cdot (n - k)$. Therefore $|\mathbf{ct}'| \leq \delta \cdot \xi \cdot n$ for the compression rate $\delta \geq 1 - \frac{k}{n}$. □

Remark 4. The oblivious compression algorithm of Figure 1 can be interpreted as (homomorphic) syndrome computation. The input is a sparse vector, viewed as an error vector added to the all-zeros codeword. The result is the syndrome associated to this error vector.

To better understand what this theorem says, let us consider a code $\mathcal{C}$ which is maximum distance separable and can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors. For any linear code that corrects up to t errors, it must hold that $d \geq 2t + 1$ and thus

$$
n - k + 1 = d \implies n - k + 1 \geq 2t + 1 \iff k \leq n - 2t,
$$

which means that the best (i.e. smallest) compression rate we can achieve with this approach is

$$
\delta = 1 - \frac{k}{n} \geq 1 - \frac{n - 2t}{n} = \frac{2t}{n}.
$$

Ignoring the encryption scheme, to just encode a plain vector $\boldsymbol{v} \in \mathbb{F}^n$ with hamming weight at most t, we need at least $\approx t \cdot (\lg|\mathbb{F}| + \lg n)$ bits, since we need to write down the t non-zero values and their locations.

But our coding-theoretic approach requires $2t \cdot \lg|\mathbb{F}|$ bits. When $\lg n \approx \lg|\mathbb{F}|$, our approach is concretely optimal, but when $\lg|\mathbb{F}| >> \lg n$, then our approach is not. We leave it open to construct an oblivious compression scheme for encryption schemes with large plaintext spaces, which achieves the concretely best possible compression rate.

Plugging a concrete efficiently syndrome decodable linear code into our blueprint from Theorem 26, we obtain the following corollaries.

Corollary 27. *Let $n, t, q, \lambda \in \mathbb{N}$, such that $q > n$ and $n > 2t > 0$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$ and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. There exists an oblivious ciphertext compression scheme for E for message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = 2t/n$. The computational complexity of compression is $\mathcal{O}\big(\frac{n}{t}\mathsf{M}(t)\log(t)\big)$ homomorphic ciphertext operations and decompression requires $\mathcal{O}(\mathsf{M}(t)\log t \log q)$ finite field operations.*

Proof. Let $\mathcal{C}$ be a normalized GRS code over $\mathbb{F}_q$ as defined in Definition 15 with parameters n and $k = n-2t$. Choose code locators $\alpha_1, \ldots, \alpha_n$ such that $\alpha_i \mapsto i$ is efficiently computable. The oblivious compression consists in computing $H \cdot \boldsymbol{c}$ homomorphically where H is the parity-check matrix of $\mathcal{C}$. By Lemma 16 with $n - k = 2t$, this requires $\mathcal{O}\big(\frac{n}{t}\mathsf{M}(t)\log(t)\big)$ linear operations in the finite field. The decompression reduces to syndrome decoding. By Theorem 18 with $n - k = 2t$, the code $\mathcal{C}$ is efficiently syndrome decodable up to $\lfloor\frac{n-k}{2}\rfloor = t$ errors, in $\mathcal{O}(\mathsf{M}(t)\log(t)\log(q))$ finite field operations since the map $\alpha_i \mapsto i$ is efficiently computable. The statement then follows from plugging the code $\mathcal{C}$ into Theorem 26. □

The complexity statement in Corollary 27 is expressed generically in terms of the complexity $\mathsf{M}(t)$ of the multiplication of two polynomials of degree less than t. It can be specialized and sharpened depending on the message space $\mathcal{M}$. The following corollary makes the precise statement for different cases.

Corollary 28. *Let $\mathcal{M} = \mathbb{F}_q$, and* OblivCompress *be the oblivious ciphertext compression scheme given in Corollary 27.*

(i) If $q = 2^\ell$, OblivCompress *requires $\mathcal{O}(n \log t)$ ciphertext operations for the compression, and $\mathcal{O}\big(t \log^2 t \cdot \log q\big)$ operations over $\mathbb{F}_{2^\ell}$ for the decompression.*

(ii) If $q \equiv 1 \bmod 2^\ell$ and $2^{\ell-1} \leq 4t < 2^\ell$, OblivCompress *requires $\mathcal{O}\big(n \log^2 t\big)$ ciphertext operations for the compression, and $\mathcal{O}\big(t \log^2 t \cdot \log q\big)$ operations over $\mathbb{F}_q$ for the decompression.*

(iii) Without any assumption on $\mathbb{F}_q$, OblivCompress *requires $\mathcal{O}\big(n \log^2 t \log\log t\big)$ ciphertext operations for the compression, and $\mathcal{O}\big(t \log^2 t \cdot \log\log t \log q\big)$ operations over $\mathbb{F}_q$ for the decompression.*

Proof. When no assumption on q is possible, as in *(iii)*, we do have $\mathsf{M}(t) = \mathcal{O}(t \log t \cdot \log\log t)$ using the generic fast polynomial multiplication algorithm of Cantor and Kaltofen [CK91]. If we further assume that $q \equiv 1 \bmod 2^\ell$ and $2^{\ell-1} \leq 4t < 2^\ell$, we can employ the *Fast Fourier Transform* with a 2^ℓth primitive root of unity, yielding a complexity of $\mathsf{M}(t) = \mathcal{O}(t \log t)$. One may remark that $2^\ell > 4t$ is necessary since the syndrome size is $n - k = 2t$. Plugging these values of $\mathsf{M}(t)$ in Corollary 27 proves *(ii)* and *(iii)*. The proof for the decompression in *(i)* corresponds exactly to Theorem 25 where we use a normalized GRS code over the LCH basis. The cost for the compression is given by Lemma 23, remarking that the mapping over an extension binary field described in Remark 3, ensures that the code locators lie in an affine subspace of $\mathbb{F}_q$, whence the cost of $\mathcal{O}(n \log t)$ homomorphic operations. □

4.1 Oblivious Compression with Auxiliary Input

In the oblivious ciphertext compression with auxiliary input setting of Bienstock et al. [BPSY23], we can further strengthen the approach from above to still obtain an optimal compression rate. The main idea here is to use the blueprint from Theorem 26, but with a linear code which has improved syndrome decoding capabilities, when additionally given the locations of the errors.

$\mathsf{OblivCompress}(\mathsf{ek}, \mathbf{ct})$	$\mathsf{Decompress}(\mathsf{dk}, \mathbf{ct}, I)$
$f(\boldsymbol{c}) := H \cdot \boldsymbol{c}$	$\boldsymbol{s} \leftarrow \mathsf{E.Dec}(\mathsf{dk}, \mathbf{ct})$
$\mathbf{ct}' \leftarrow \mathsf{E.Eval}(\mathsf{ek}, f, \mathbf{ct})$	$\boldsymbol{m} \leftarrow \mathcal{C}.\mathsf{Decode}(\boldsymbol{s}, I)$
return $\mathbf{ct}'$	**return** $\boldsymbol{m}$

Fig. 2. Auxiliary-input oblivious ciphertext compression scheme. E is a homomorphic encryption scheme and $\mathcal{C}$ is a linear code with parity-check matrix H.

Theorem 29. *Let $n, t, k, q, \lambda \in \mathbb{N}$, let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$ that is efficiently syndrome decodable from known support up to t errors and let $H \in \mathbb{F}^{(n-k)\times n}$ be a parity-check matrix for the code $\mathcal{C}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$. Let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Then the construction in Figure 2 is an auxiliary-input oblivious ciphertext compression scheme for E with message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = 1 - \frac{k}{n}$.*

Proof. The proof is completely analogous to that of Theorem 26 with the only difference being that we rely on a linear code that is efficiently syndrome decodable from known support. □

Instantiating the above theorem with the linear code from Theorem 18, we obtain an auxiliary-input oblivious compression scheme with optimal compression rate. The proof of the next corollary is, *mutatis mutandis*, the same as the proof of Corollary 27. We use a normalized GRS code over $\mathbb{F}_q$ that is efficiently syndrome decodable from known support up to t errors, with parameters n and $k = n - t$. In this case, there is no need for the map $\alpha_i \mapsto i$ to be efficiently computable. This means that one can choose code locators in geometric progression and benefit from the improved complexity in Lemma 16.

Corollary 30. *Let $n, t, q, \lambda \in \mathbb{N}$, such that $q > n$ and $n > t > 0$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$ and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. There exists an auxiliary-input oblivious ciphertext compression scheme for E for message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = t/n$. The compression requires $\frac{n}{t}\mathsf{M}(t) + \mathcal{O}(n)$ homomorphic ciphertext operations and the decompression $\mathcal{O}(\mathsf{M}(t) \log t)$ finite field operations.*

Similarly to Corollary 28 for the regular oblivious compression scheme, we can make the precise the complexity statements with an auxiliary input.

Corollary 31. *Let $\mathcal{M} = \mathbb{F}_q$, and $\mathsf{OblivCompressAux}$ be the auxiliary-input oblivious ciphertext compression scheme given in Corollary 30.*

(i) If $q = 2^\ell$, or $q \equiv 1 \bmod 2^\ell$ and $2^{\ell-1} \leq 2t < 2^\ell$, $\mathsf{OblivCompressAux}$ requires $\mathcal{O}(n \log t)$ homomorphic operations for the compression, and $\mathcal{O}\big(t \log^2 t\big)$ operations over $\mathbb{F}_q$ for the decompression.

(ii) Without any assumption on $\mathbb{F}_q$, $\mathsf{OblivCompressAux}$ requires $\mathcal{O}(n \log t \log\log t)$ homomorphic operations for the compression, and $\mathcal{O}\big(t \log^2 t \log\log t\big)$ operations over $\mathbb{F}_q$ for the decompression.

Proof. This is a direct consequence of Corollary 30 using the exact complexities for $\mathsf{M}(t)$. For *(ii)*, $\mathsf{M}(t) = \mathcal{O}(t \log t \log\log t)$ using the generic fast polynomial algorithm of Cantor and Kaltofen [CK91]. For *(i)*, in both cases, $\mathbb{F}_q$ has an FFT algorithm of complexity $\mathcal{O}(t \log t)$ (using additive FFT of [LCH14] over binary extension field, and using the classic Cooley-Tukey FFT otherwise), yielding $\mathsf{M}(t) = \mathcal{O}(t \log t)$. □

5 Decompression

Let us now see how linear codes allow for efficient oblivious ciphertext decompression. Recall that in this setting the roles of compressor and decompressor are switched around. The compressor is given the sparse representation of a plaintext vector $\boldsymbol{m}$ of bounded hamming weight and is asked to produce a succinct

$\mathsf{Compress}(\mathsf{ek}, \boldsymbol{p})$	$\mathsf{OblivDecompress}(\mathsf{ek}, \mathbf{ct})$
1 : **parse** $\boldsymbol{p}$ **as** $\{(p_i, i) \mid p_i \neq 0\}$	1 : $f(\boldsymbol{u}) := \boldsymbol{u} \cdot H$
2 : $I := \{i \in [n] \mid p_i \neq 0\}$	2 : $\mathbf{ct}' \leftarrow \mathsf{E.Eval}(\mathsf{ek}, f, \mathbf{ct})$
3 : $\boldsymbol{m} = \mathcal{C}.\mathsf{Solve}(\boldsymbol{p}, I)$	3 : **return** $\mathbf{ct}'$
4 : $\mathbf{ct} \leftarrow \mathsf{E.Enc}(\mathsf{ek}, \boldsymbol{m})$	
5 : **return** $\mathbf{ct}$	

Fig. 3. Oblivious ciphertext decompression from linear codes. E is a homomorphic encryption scheme and $\mathcal{C}$ is a linear code with parity-check matrix H and $\mathsf{Solve}(\boldsymbol{p}, I)$ finds $\boldsymbol{m}$, such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ satisfies $c_i = p_i$ for all $i \in I$.

representation $\mathbf{ct}$ of this vector. The decompressor is given $\mathbf{ct}$ and is asked to obliviously decompress it into a dense encrypted vector $\mathbf{ct'}$, such that ct'_i is an encryption of m_i for each non-zero m_i. In terms of privacy, we want $\mathbf{ct}$ to hide the locations of the non-zero entries of $\boldsymbol{m}$ from the decompressor.

Theorem 32. *Let $n, t, k, q, \lambda \in \mathbb{N}$, let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$ that is efficiently syndrome decodable up to t errors and let $H \in \mathbb{F}^{(n-k)\times n}$ be a parity-check matrix for the code $\mathcal{C}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$ and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. Then the construction in Figure 3 is an oblivious ciphertext decompression scheme for E for message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = 1 - \frac{k}{n}$.*

Proof. Let $\boldsymbol{p} \in \mathcal{P}_t$, and let $I = \{i \in [n] \mid p_i \neq 0\}$. By definition of $\mathcal{C}$ and $\mathcal{P}_t$, it is clear that $|I| \leq t \leq d-1$. We can then apply Corollary 14 to efficiently compute a vector $\boldsymbol{m} \in \mathbb{F}_q^{n-k}$ such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ and $c_i = p_i$ for all $i \in I$. We remark that no further hypothesis can be made of the values of c_j for all $j \notin I$. The encoding $\mathbf{ct}$ of $\boldsymbol{m}$ with the encryption scheme E satisfies $|\mathbf{ct}| = \xi \cdot (n-k)$. We next observe that decompression only requires an additive homomorphic encryption scheme since the function $f : \boldsymbol{u} \mapsto \boldsymbol{u} \cdot H$ only requires additions and multiplications by constants. Furthermore, by the definition of f, we know that $\boldsymbol{c} \leftarrow f(\boldsymbol{m})$ satisfies $c_i = p_i$ for all $i \in I$. Since c is a dense vector of dimension n, we have $|\mathbf{ct}'| \leq \xi \cdot n$ and then $\delta = |\mathbf{ct}|/|\mathbf{ct}'| = (n-k)/n = 1 - \frac{k}{n}$. The following observation follows almost straightforwardly from the correctness of E:

$$
\begin{aligned}
& \Pr\left[\exists\, i \in [n]\ (p_i \neq 0 \wedge p_i \neq p'_i) \colon \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{ct} \leftarrow \mathsf{Compress}(\mathsf{ek}, \mathsf{sparse}(\boldsymbol{p})) \\ \mathbf{ct}' \leftarrow \mathsf{OblivDecompress}(\mathsf{ek}, \mathbf{ct}) \\ \boldsymbol{p}' \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathbf{ct}') \end{array}\right] \\
= \ & \Pr\left[\exists\, i \in [n]\ (p_i \neq 0 \wedge p_i \neq p'_i) \colon \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \boldsymbol{m} \leftarrow \mathcal{C}.\mathsf{Solve}(\boldsymbol{p}, I) \\ \mathbf{ct} \leftarrow \mathsf{E.Enc}(\mathsf{ek}, \boldsymbol{m}) \\ \mathbf{ct}' \leftarrow \mathsf{E.Eval}(\mathsf{ek}, f, \mathbf{ct}) \\ \boldsymbol{p}' \leftarrow \mathsf{Dec}(\mathsf{dk}, \mathbf{ct}') \end{array}\right] \\
\leq \ & \Pr\left[\exists\, i \in [n]\ (p_i \neq 0 \wedge p_i \neq p'_i) \colon \boldsymbol{p}' \leftarrow f(\boldsymbol{m})\right] + \mathsf{negl}(\lambda) \\
= \ & \mathsf{negl}(\lambda).
\end{aligned}
$$

We finally need that our decompression meets IND-CPA security. Since the ciphertext decompression is done through a non-encrypted linear function, it is enough to assume that E is IND-CPA secure. □

Remark 5. The construction of Figure 3 can be interpreted in terms of erasure correction in the dual code $\mathcal{C}^\perp$ of $\mathcal{C}$. The generator matrix of $\mathcal{C}^\perp$ is the parity-check matrix H of $\mathcal{C}$. The compression algorithm is

given a sparse vector $\boldsymbol{p}$, viewed as a codeword (of the dual code) where some entries have been erased. It corrects these erasures, that is, it computes a codeword $\boldsymbol{c}$ that agrees with $\boldsymbol{p}$ on its non-zero entries, and compresses $\boldsymbol{c}$ by computing the corresponding message $\boldsymbol{m}$. The oblivious decompression is given the message and homomorphically computes the associated codeword.

Plugging concrete linear codes, such as the *normalized Generalized Reed-Solomon* codes of Section 2, in the framework of Theorem 32, we obtain easily the following corollary.

Corollary 33. *Let $n, t, q, \lambda \in \mathbb{N}$, such that $q > n$ and $n > t > 0$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an additively homomorphic encryption scheme with message space $\mathcal{M} = \mathbb{F}_q$ and ciphertext length $\xi = \xi(\lambda)$ and let $\mathcal{P}_t = \{\boldsymbol{v} \in \mathcal{M}^n \mid \mathsf{hw}(\boldsymbol{v}) \leq t\}$. There exists an oblivious ciphertext decompression scheme for E for message space $\mathcal{P}_t$ with error $\epsilon = \mathsf{negl}(\lambda)$ and compression rate $\delta = t/n$. The decompression requires $\frac{n}{t}\mathsf{M}(t) + \mathcal{O}(n)$ homomorphic ciphertext operations and the compression $\mathcal{O}(\mathsf{M}(t)\log t)$ finite field operations.*

Proof. Let $\mathcal{C}$ be a normalized GRS code as in Definition 15, with parameters n and $k = n - t$ and code locators $\alpha_i = \gamma^i$, $1 \leq i \leq n$, for some $\gamma \in \mathbb{F}_q$ of order at least n. The compression computes a message $\boldsymbol{m}$ such that $\boldsymbol{c} = \boldsymbol{m} \cdot H$ agrees with the input $\boldsymbol{p}$ on its non-zero entries, where H is the parity-check matrix of $\mathcal{C}$. By Lemma 17 with $n-k = t$, this requires $\mathcal{O}(\mathsf{M}(t)\log(t))$ finite field operations. The oblivious decompression computes homomorphically $\boldsymbol{c} \cdot H$. Since the code locators are in geometric progression, Lemma 16 with $n - k = t$ shows that this requires $\frac{n}{t}\mathsf{M}(t) + O(n)$ homomorphic operations. The statement follows from plugging the code $\mathcal{C}$ into Theorem 32. □

We can directly refine the complexity statements for our oblivious decompression scheme depending on the message space according to the cost of $\mathsf{M}(t)$, as for Corollary 31.

Corollary 34. *Let $\mathcal{M} = \mathbb{F}_q$, and* OblivDecompress *be the given oblivious ciphertext decompression scheme given in Corollary 33.*

(i) If we assume that $q = 2^\ell$, or that $q \equiv 1 \bmod 2^\ell$ and $2^{\ell-1} \leq 2t < 2^\ell$, OblivDecompress *requires $\mathcal{O}(n \log t)$ ciphertext operations for the decompression, and $\mathcal{O}\big(t \log^2 t\big)$ operations over $\mathbb{F}_q$ for the compression.*

(ii) If no assumption is made on $\mathbb{F}_q$, OblivDecompress *requires $\mathcal{O}(n \log t \log\log t)$ ciphertext operations for the decompression, and $\mathcal{O}\big(t \log^2 t \log\log t\big)$ operations over $\mathbb{F}_q$ for the compression.*

References

ACLS18. Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*, pages 962–979, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press. doi:10.1109/SP.2018.00062. 1, 1.3

AFS18. Adi Akavia, Dan Feldman, and Hayim Shaul. Secure search on encrypted data via multi-ring sketch. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 985–1001, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243810. 1, 1.3

AGHL19. Adi Akavia, Craig Gentry, Shai Halevi, and Max Leibovich. Setup-free secure search on encrypted data: Faster and post-processing free. *Proc. Priv. Enhancing Technol.*, 2019(3):87–107, 2019. 1.3

Ber70. Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970. doi:10.1090/S0025-5718-1970-0276200-X. 3.1, 3.2

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_12. 1.3

BIM00. Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 55–73, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-44598-6_4. 1.3

BLS03. A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 37–44, New York, NY, USA, 2003. ACM. doi:10.1145/860854.860870. 2, 3.1, 3.1, 3.2, 3.2

BPSY23. Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Batch PIR and labeled PSI with oblivious ciphertext compression. In Joseph A. Calandrino and Carmela Troncoso, editors, *33nd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, pages 301–318. USENIX Association, 2023. 1, 1, 1.2, 1, 1.2, 1.2, 1.3, 1.3, 2.2, 2.2, 4.1

BS05. Alin Bostan and Éric Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, 2005. doi:10.1016/j.jco.2004.09.009. 3.1, 3.1

CDG+21. Seung Geol Choi, Dana Dachman-Soled, S. Dov Gordon, Linsheng Liu, and Arkady Yerukhimovich. Compressed oblivious encoding for homomorphically encrypted search. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2277–2291, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press. doi:10.1145/3460120.3484792. 1, 1, 1.2, 1.2, 1.3

CGKS95. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. doi:10.1109/SFCS.1995.492461. 1.3

CK91. David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991. 4, 4.1

CKK16. Jung Hee Cheon, Miran Kim, and Myungsun Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Transactions on Information Forensics and Security*, 11(1):188–199, January 2016. doi:10.1109/TIFS.2015.2483486. 1

Cox21. Nicholas Coxon. Fast transforms over finite fields of characteristic two. *Journal of Symbolic Computation*, 104:824–854, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0747717120301127, doi:10.1016/j.jsc.2020.10.002. 3.2, 20

FLOS24. Nils Fleischhacker, Kasper Green Larsen, Maciej Obremski, and Mark Simkin. Invertible Bloom lookup tables with less memory and randomness. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 54:1–54:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 1, 1.2, 1.2

FLS23. Nils Fleischhacker, Kasper Green Larsen, and Mark Simkin. How to compress encrypted data. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 551–577, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30545-0_19. 1, 1.2, 1.2, 2.2, 2

For65. G. Forney. On decoding BCH codes. *IEEE Transactions on Information Theory*, 11(4):549–557, 1965. doi:10.1109/TIT.1965.1053825. 3.2

Gen09. Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. 1.2

GI14. Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_35. 1.3

LANH16. Sian-Jheng Lin, Tareq Y. Al-Naffouri, and Yunghsiang S. Han. FFT algorithm for binary extension finite fields and its application to Reed–Solomon codes. *IEEE Transactions on Information Theory*, 62(10):5343–5358, 2016. doi:10.1109/TIT.2016.2600417. 3.2, 20, 3.2, 3.2

LCH14. Sian-Jheng Lin, Wei-Ho Chung, and Yunghsiang S. Han. Novel polynomial basis and its application to Reed-Solomon erasure codes. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, page 316–325, USA, 2014. IEEE Computer Society. doi:10.1109/FOCS.2014.41. 3.2, 19, 20, 3.2, 22, 4.1

LT22. Zeyu Liu and Eran Tromer. Oblivious message retrieval. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 753–783, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15802-5_26. 1, 1, 1.2, 1.2, 1.3

LTW24. Zeyu Liu, Eran Tromer, and Yunhao Wang. PerfOMR: Oblivious message retrieval with reduced communication and computation. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3169–3186, Philadelphia, PA, August 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/liu-zeyu. 1, 1, 1.2, 1.3

Rab80. M. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, May 1980. doi:10.1137/0209024. 3.1, 3.2

RAD78. Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, Richard J. Lipton, David P. Dobkin, and Anita K. Jones, editors, *Foundations of secure computation*, pages 169–179. Academic Press, 1978. 1.2

Rot06. Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, Cambridge, UK ; New York, 2006. 13, 15, 3.1

vzGG13. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013. 3.1, 3.1, 3.1, 3.2

WYXZ20. Rui Wen, Yu Yu, Xiang Xie, and Yang Zhang. LEAF: A faster secure search algorithm via localization, extraction, and reconstruction. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1219–1232, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3417237. 1.3

YSK+13. Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Secure pattern matching using somewhat homomorphic encryption. In Ari Juel and Bryan Parno, editors, *CCSW 2013: The ACM Cloud Computing Security Workshop*, pages 65–76, Berlin, Germany, November 8, 2013. ACM Press. doi:10.1145/2517488.2517497. 1