Expand-Convolute Codes for Pseudorandom Correlation Generators from LPN

Srinivasan Raghuraman[1], Peter Rindal[2], Titouan Tanguy[3]

[1] Visa Research and MIT
srraghur@visa.com
[2] Visa Research
peterrindal@gmail.com
[3] imec-COSIC, KU Leuven, Belgium
titouan.tanguy@kuleuven.be

Abstract. The recent development of pseudorandom correlation generators (PCG) holds tremendous promise for highly efficient MPC protocols. Among other correlations, PCGs allow for the efficient generation of oblivious transfer (OT) and vector oblivious linear evaluations (VOLE) with sublinear communication and concretely good computational overhead. This type of PCG makes use of a so-called LPN-friendly error-correcting code. That is, for large dimensions the code should have very efficient encoding and have high minimum distance.

We investigate existing LPN-friendly codes and find that several candidates are less secure than was believed. Beginning with the recent *expand-accumulate* codes, we find that for their *aggressive* parameters, aimed at good concrete efficiency, they achieve a smaller [pseudo] minimum distance than conjectured. This decreases the resulting security parameter of the PCG but it remains unclear by how much. We additionally show that the recently proposed and extremely efficient *silver* codes achieve only very small minimum distance and result in concretely efficient attacks on the resulting PCG protocol. As such, silver codes should not be used.

We introduce a new LPN-friendly code which we call *expand-convolute*. These codes have provably high minimum distance and faster encoding time than suitable alternatives, e.g. expand-accumulate. The main contribution of these codes is the introduction of a convolution step that dramatically increases the minimum distance. This in turn allows for a more efficient parameter selection which results in improved concrete performance. In particular, we observe a 3 times improvement in running time for a comparable security level.

1 Introduction

The use of correlated randomness has emerged as the de facto method for efficient multiparty computation (MPC) and other cryptographic protocols. One of the most fundamental examples is the oblivious transfer correlation which drives a large fraction of efficient MPC protocols. Therefore the efficient generation of such correlated randomness is of paramount importance for realizing the

real-world potential of these protocols. The recent development of pseudorandom correlation generators (PCG)[12] has been suggested as the preferred way of generating this correlation due to extremely good communication overhead, typically sublinear, and compelling computational overheads. However, more research is needed to realize the true potential of these techniques. This includes an improved understanding of the underlying security assumptions and better computational efficiency.

A PCG overview. The two party PCG protocols we will focus on have the following structure. During a concretely efficient setup phase, the parties will interactively generate random vectors $\vec{a'}, \vec{b'}, \vec{c'} \in \mathbb{F}^n$ and a scaler $\Delta \in \mathbb{F}$ such that

$$\vec{a'}\Delta + \vec{b'} = \vec{c'}$$

One party will hold $\vec{a'}, \vec{b'}$ while the other holds $\vec{c'}, \Delta$. These values are uniformly distributed with the caveat that $\vec{a'}$ is extremely sparse. The protocol then transforms these $\vec{a'}, \vec{b'}, \vec{c'}$ vectors into unformly random vectors $\vec{a}, \vec{b}, \vec{c} \in \mathbb{F}^k$ that have the same correlation, $\vec{a}\Delta + \vec{b} = \vec{c}$. Typically, k will be large such as 2^{20}. This correlation is known as vector oblivious linear evaluation (VOLE) which is used by several state of art protocols for zero-knowledge and private set intersection. Alternatively, the VOLE correlation can be efficiently converted into k oblivious transfer (OT) correlations which are used by countless protocols.

The core idea of the transformation is to multiply the $\vec{a'}, \vec{b'}, \vec{c'}$ vectors by the transpose of an error correcting generator matrix G. This effectively *compresses* these vectors to obtain $\vec{a} = \vec{a'}G^{\mathsf{T}}, \vec{b} = \vec{b'}G^{\mathsf{T}}, \vec{c} = \vec{c'}G^{\mathsf{T}}$. Given that the error correcting code G has high minimum distance, the sparse vector $\vec{a'}$ will be compressed into a slightly shorter but uniformly random vector $\vec{a}$. Or more formally, $\vec{a}$ is indistinguishable from uniform if the dual LPN assumption holds for G (see Section 2.4).

The main overhead in these protocols is the multiplication with the error correcting code G. Various codes have been proposed with the main challenge being that G must have high minimum distance to argue security while also having efficient multiplication.

1.1 Our Contributions

We propose a new class of error-correcting codes with provably high minimum distance which are optimized for use in very large learning parity with noise (LPN) instances for Pseudorandom Correlation Generators (PCG). We name the codes Expand Convolute after the recent Expand Accumulate codes of Boyle et al. [10] and convolutional codes (see Section 2.5 and [7]) that we build on. Our new codes offer several compelling features. First is that they achieve linear minimum distance (in the code length) which implies that our codes can provably prevent a very large class of attacks when used for LPN and pseudorandom correlation generators. At the same time our codes are very efficient. The encoding/matrix multiply time requires near-linear time and concretely outperforms the prior art [10]. Our code takes Expand Accumulate codes as a starting point

and replaces the accumulation with a convolutional code that we design. Each step of the convolution is mildly less efficient but allows for a drastically bigger rate, e.g. 0.6 as opposed to 0.2. This in turn allows our code to be much more efficient, approaching a $2.8 = 0.6/0.2$ times improvement in the overall running time of the final pseudorandom correlation generator. Alternatively, our construction can decrease the expander parameter by approximately 2 times which similarly improves performance. Finally we introduce a systematic version of our code that achieves a further $1.5\times$ running time improvement at the expense of a small decrease of minimum distance.

When analyzing the concrete parameters of our new code we observe that the aggressive parameters proposed by [10] for their code fall short of their claimed security level. In particular, we are able to empirically find (pseudo) minimum distances of their code that are $30\times$ smaller than conjectured. It does not appear that this results in a practical attack on their construction but certainly decreased their security margin. Due to having a similar structure, one can apply similar techniques to our code. However, as expected we observe that our introduction of a convolution results in a much higher pseudo minimum distance (i.e. the smallest weight codeword that can efficiently be computed).

Our third contribution is demonstrating that the recent Silver codes [14] achieve only sublinear minimum distance and are unsuitable for use in LPN. The compelling feature of Silver was extremely good computational performance. The techniques used to get this efficiency also made proving bounds on the minimum distance of these codes difficult. The authors instead chose to rely on empirical methods for estimating the minimum distance of small codes and extrapolated the distance for the large codes used by PCGs. The authors conjectured their code achieved linear minimum distance with their empirical bounds as supporting evidence. However, we show that the Silver codes only achieve poly log minimum distance which leads to a concretely efficient distinguisher for their PCG construction based on LPN.

Compared to our new codes, the encoding procedure for Silver is several times faster. However, given the sublinear distance of Silver, some added overhead over the silver codes appears necessary. Moreover, our new codes are the fastest known codes that also achieve provable linear minimum distance. In particular, we are twice as fast as [10] for comparable security.

1.2 Technical Overview of Expand Convolute

Both our new expand convolute code and the prior art of expand accumulate [10] describe the generator matrix $G \in \{0,1\}^{k\times n}$ as the multiplication of two other matrices, an expander $B \in \{0,1\}^{k\times n}$ and a convolution $C \in \{0,1\}^{n\times n}$, such that $G = BC$. The expander B is highly sparse with row (and column) hamming weight being $O(\log k)$. Informally, the expander mixes disparate parts of the message together. The convolution C is an upper triangular matrix with a special structure, see Section 2.5. In particular, given $\vec{y} = \vec{x}C$, then for all i, we can write y_i as a linear combination of x_i and $y_{i-1}, \ldots, y_{i-m}$ for some small parameter m. Informally, the convolution can be thought of as performing thorough local

mixing of the last m outputs and x_i. Together, the expander and convolution perform both global and local mixing which give it strong minimum distance properties.

The structure of B, C gives a natural algorithm for computing $\vec{x}G$. For $i = 1, \ldots, n$, we sum the $O(\log k)$ positions of $\vec{x}$ that are indexed by the ith column of B, i.e. $t_i = \vec{B}_{\cdot,i} \cdot \vec{x}$ and use that to update the convolution as $y_i = t_i + \sum_{j=1,\ldots,m} \alpha_{i,j} y_{i-j}$, where $\alpha_{i,j}$ are the random coefficients that define the convolution. Setting $m = 1$ and all of the convolution coefficients $\alpha_{i,1}$ to be 1, we obtain the expand accumulate codes. In contrast, we will define $m \approx 20$ and uniformly sample the convolution coefficients.

The critical security property that we must guarantee is that the code generated by G has high minimum distance. This is equivalent to requiring that low (non-zero) hamming weight codewords $\vec{y} = \vec{x}G$ do not exist. For some iteration i, we refer to the state $\vec{\sigma}_i$ of the convolution as the previous m outputs, i.e., a column vector $\vec{\sigma}_i = (y_{i-1}, \ldots, y_{i-m})^\mathsf{T}$. For the sake of contradiction, let us assume $\vec{x}$ results in a low weight codeword $\vec{y} = \vec{x}BC$, i.e., $\vec{y}$ is mostly zero. Next observe that when the state σ_i is non-zero, the next output $y_i = 1$ with probability $1/2$ over the random choices of $\alpha_{i,j}$, a.k.a C. Therefore, except with negligible probability, for y to be low weight, the state must be zero most of the time.

This brings us to the core idea of our construction. The probability that convolution state can transition to the zero state from non-zero is roughly proportional to 2^{-m}. Observe that while the state is non-zero there is a $1/2$ probability of getting $y_i = 1$ (over the choices of $\alpha_{i,j}$, a.k.a C). Given that $y_i = 1$, *at least* m more iterations are required before the state can *possibly* transition to zero. In particular, if $y_i = 1$ then $y_{i+1} = \ldots = y_{i+m} = 0$ must happen to transition to the zero state $\vec{\sigma}_{i+m+1} = (y_{i+1}, \ldots, y_{i+m}) = \vec{0}$. Given that for $j \in [m]$, $\Pr_C[y_{i+j} = 1 \mid y_i = 1] = 1/2$, we have the probability of transiting to state zero at step $i + m$ given $y_i = 1$ is 2^{-m}, i.e. $\Pr_C[\vec{\sigma}_{i+m+1} = 0 \mid y_i = 1] = 2^{-m}$. We note that formalizing this intuition for all i requires significant nuance due to the fact that these events are not all independent, e.g. when $y_i = 0$.

By setting $m = O(\log k)$, in expectation any given message $\vec{x}$ will not transition to the all zero state and is likely to have high weight. In particular, a larger m results in higher minimum distance. In contrast, the expand accumulate code has $m = 1$ and therefore can easily transition back to the zero state.

While the conceptual idea of making the state larger is simple, the main challenge lies in proving exactly how much increasing m helps. To achieve this we model the process of generating $\vec{y}$ as a walk on a Markov chain, where the randomness is over the choice of B, C. Unfortunately, naively modeling this process results in extremely loose bounds and suggests a larger m is bad. We show a series of transformations between related Markov chains such that a final chain can yield tight bounds on the minimum distance of the code.

Organization. We begin in Section 2 by reviewing background material and the closely related expand accumulate codes of [10]. We then introduce the Silver codes of [14] in Section 3 and discuss why they achieve sublinear minimum

distance. In Section 5 we introduce our new codes. We present two different convolutional codes which we call non-wrapping and wrapping, where the latter achieves better practical performance. Building on these convolutional codes we design our final codes in a manner similar to expand accumulate codes, with the main difference being that our codes achieve a much better rate and or a more efficient expander.

2 Preliminaries

2.1 Notation

$[a, b]$ denotes the set $\{a, ..., b\}$ with $[n]$ being shorthand for $[1, n]$. Let $[a, b]_{\mathbb{R}}$ be the interval a to b over the real values. $(a, b]$ and $[a, b)$ denotes the range excluding the first and last element, respectively. Let $\vec{x} \in \mathcal{R}^n$ be a row vector over a domain $\mathcal{R}$ of length n. x_i denotes the ith element. Alternatively, let $\vec{x} \in \mathcal{R}^V$ for a set V denote a vector of length $|V|$ and is index by elements of V. Let $\bar{y}$ be a column vector. Let $(x_1, ..., x_n)$ denote a row vector. For a set S, let $\vec{x}_S$ denote the subvector index by S, i.e. $(x_{S_1}, ..., x_{S_{|S|}})$. We use upper case to denote matrices, e.g. M. M_i denotes the ith row and $M_{\cdot,i}$ denotes the ith column of M. M^{T} and $\vec{x}^{\mathsf{T}}$ are the transpose of a matrix and vector respectively.

2.2 Background on Markov Chains

Markov chain. A Markov chain is defined by a finite set V of state space and transition matrix $P \in \mathbb{R}^{V \times V}$ s.t. all rows sum to 1 with > 0 entries. For $\vec{u}, \vec{v} \in V$, $P_{u,v}$ is the probability of transitioning from state $\vec{u}$ to $\vec{v}$. P thus naturally describes a random walk:
Let $\vec{\nu} \in \mathbb{R}^V$ denote a distribution over the state space, so ν_v is the probability of sampling state v. Sample a state $x_0 \leftarrow \vec{\nu}$. For $i \in [n]$, sample $x_i \leftarrow P_{x_{i-1}}$.

Stationary distribution. The stationary distribution $\vec{\nu} \in \mathbb{R}^V$ of a Markov chain is a distribution over V such that $\vec{\nu}P = \vec{\nu}$ (e.g. $= \vec{\nu}$ is a left eigenvector for eigenvalue $\lambda_1 = 1$)

Irreducible. A Markov chain is irreducible if any state can be reached from any other state in a finite number of steps. It is well known that if a chain is irreducible, it has a unique stationary distribution.

Reversible. A chain is irreducible reversible if $\forall u, v \in V, \quad \nu_u P_{u,v} = \nu_v P_{v,u}$.

Expander Hoeffding Bound. For a function $f : V \to [0, 1]$, let $\mu = \mathbb{E}_{x \leftarrow \vec{\nu}}[f(x)]$ be the expected value under distribution $\vec{\nu}$. Consider the likelihood that the random variable $S_n = \sum_{i \in [n]} f(x_i)$ deviates a lot from $E_{x \leftarrow \vec{\nu}}[S_n]$, where $(x_1, \ldots, x_n)$ is an n-step random walk on the chain. This distribution is closely related to the second eigenvalue λ_2 of P by the following theorem.

Theorem 1 (Expander Hoeffding Bound). *Let (V, P) denote a finite, irreducible and reversible Markov chain, with stationary distribution $\vec{\pi}$ and second*

largest eigenvalue λ_2*. Let* $f : V \to [0,1]$ *with* $\mu = E_{x \leftarrow \vec{\pi}}[f(x)]$*, and consider* $S_n = \sum_{i \in [n]} f(x_i)$ *with* $x_0 \leftarrow_{\vec{\pi}} V$ *and* $x_1, \ldots, x_n$ *is a random walk which starts from* x_0*. Then for* $\tilde{\lambda}_2 = \max(0, \lambda_2)$*, and any* $\epsilon > 0$*, the following concentration bounds hold:*

$$\Pr[S_n > n\mu + \epsilon] \leq \exp\left(-2\frac{\epsilon^2}{n} \cdot \frac{1-\tilde{\lambda}_2}{1+\tilde{\lambda}_2}\right)$$

$$\Pr[S_n < n\mu - \epsilon] \leq \exp\left(-2\frac{\epsilon^2}{n} \cdot \frac{1-\tilde{\lambda}_2}{1+\tilde{\lambda}_2}\right)$$

Corollary for other starting distributions. We care about the special case where the starting distribution is fixed to be a specific $v \in V$. The bound for this case is (from the *total probability rule*):

$$\Pr[S_n > n\mu + \epsilon] \leq \frac{1}{\pi_v} \exp\left(-2\frac{\epsilon^2}{n} \cdot \frac{1-\tilde{\lambda}_2}{1+\tilde{\lambda}_2}\right)$$

$$\Pr[S_n < n\mu - \epsilon] \leq \frac{1}{\pi_v} \exp\left(-2\frac{\epsilon^2}{n} \cdot \frac{1-\tilde{\lambda}_2}{1+\tilde{\lambda}_2}\right)$$

2.3 Coding Theory

Generator Matrix. Let $k < n \in \mathbb{N}$. Let $G \in \mathcal{R}^{k \times n}$ be a $k \times n$ matrix over the field $\mathcal{R}$, it generates the set of codewords $\mathcal{C} := \{\vec{c} = \vec{x}G \mid \vec{x} \in \mathcal{R}^k\}$ where $\vec{c}$ is a codeword of the linear code $\mathcal{C}$, and $\vec{x}$ is any input k-length vector. G is the *generator matrix* of $\mathcal{C}$. Often it will be useful to consider a family of error correcting codes that can be sampled from by some randomized procedure C.

Standard Form. The *standard form* for a generator matrix is, $G = [I_k|P]$ where I_k is the $k \times k$ identity matrix and P is a $k \times m'$ matrix, where $m' = n - k$. When the generator matrix is in standard form, the code $\mathcal{C}$ is said to be *systematic* and the message can be read directly from the first k positions codeword, i.e. $c = G\vec{x} = [\vec{x}|\vec{p}]$ for some parity information $\vec{p} \in \mathcal{R}^m$. An arbitrary generator matrix G' can be placed in standard form by performing elementary row operations and possibly column swaps.

Parity Check Matrix. The matrix $H \in \mathcal{R}^{m' \times n}$ representing a linear function $\phi : \mathcal{R}^n \to \mathcal{R}^{n-k}$ whose kernel is $\mathcal{C}$ is called a *parity check matrix* of $\mathcal{C}$. Equivalently, H is a matrix whose kernel, a.k.a. null space, is $\mathcal{C}$, i.e. $\mathcal{C} = \{\vec{c} \mid H\vec{c}^{\mathsf{T}} = 0\}$. It can be verified that H is a $m' \times n$ matrix. The code generated by H is called the *dual code* of $\mathcal{C}$, denoted by $\mathcal{C}^{\perp}$.

Minimum Distance. The distance d of a linear code $\mathcal{C}$ can be defined as the minimum number of positions of a codeword $\vec{c} \in \mathcal{C}$ which must be modified to produce another codeword $\vec{c}'$. Equivalently, it is the minimum number of linearly dependent columns of the parity check matrix H. Due to the linearity of $\mathcal{C}$, it is easy to see that d is equal to the minimum weight non-zero codeword.

Computing the minimum distance for an arbitrary linear code $\mathcal{C}$ is known to be NP-complete [22].

Relating the Generator and Parity Check Matrices A generator matrix can be used to construct the parity check matrix for a code (and vice versa). If the generator matrix G is in standard form, $G = [I_k|P]$ then the parity check matrix for $\mathcal{C}$ is $H = [-P^\mathsf{T}|I_{n-k}]$ where P^T is the transpose of the matrix P and I_{n-k} is the $m' \times m'$ identity matrix.

2.4 Syndrome Decoding and Learning Parity with Noise

Our constructions follows the recent line of works [10,14,11,12] to propose a new variant of the learning parity with noise (LPN) assumption (more accurately, a type of syndrome decoding assumption). The learning parity with noise assumption is one of the most fundamental assumptions of cryptography, introduced in the work of [8]; related problems were used even earlier [18]. The LPN assumption over a field $\mathcal{R}$ states, informally, that $(A, A \cdot \vec{s} + \vec{e})$ can not efficiently be distinguished from $(A, \vec{b})$, where A is sampled from some ensemble of matrices, $\vec{s}$ is a uniform secret vector over $\mathcal{R}$, $\vec{e}$ is a *noise vector* sampled from some distribution over sparse $\mathcal{R}$-vectors. $\vec{b}$ is a uniform vector over $\mathcal{R}$. More formally, we define the LPN assumption over $\mathcal{R}$ with dimension m', number of samples n, w.r.t. a code generation algorithm C, and a noise distribution $\mathcal{D}$:

Definition 1 (Primal LPN [14]). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{m',n}(\mathcal{R})\}_{m',n\in\mathbb{N}}$ denote a sequence of efficiently sampleable distributions indexed by a pair of integers (m', n) over a ring $\mathcal{R}$, such that for any $m', n \in \mathbb{N}$, $\mathsf{Im}(\mathcal{D}_{m',n}(\mathcal{R})) \subseteq \mathcal{R}^n$. Let C be a probabilistic code generation algorithm such that $\mathsf{C}(m', n, \mathcal{R})$ outputs a matrix $A \in \mathcal{R}^{n\times m'}$. For dimension $m' = m'(\kappa)$, number of samples (or block length) $n = n(\kappa)$, and ring $\mathcal{R} = \mathcal{R}(\kappa)$, the (primal) $(\mathcal{D}, \mathsf{C}, \mathcal{R})$-$\mathsf{LPN}(m', n)$ assumption states that*

$$\{(A, \vec{b}) \text{ s.t. } A \leftarrow \mathsf{C}(m', n, \mathcal{R}), \vec{e} \leftarrow \mathcal{D}_{m',n}(\mathcal{R}), \vec{s} \leftarrow \mathcal{R}^{m'}, \vec{b} \leftarrow A \cdot \vec{s} + \vec{e}\}$$
$$\overset{c}{\approx} \{(A, \vec{b}) \text{ s.t. } A \leftarrow \mathsf{C}(m', n, \mathcal{R}), \vec{b} \leftarrow \mathcal{R}^n\}.$$

This definition captures not only LPN type assumptions but also assumptions such as LWE or the multivariate quadratic assumption. For our purposes, we restrict the assumptions such that the noise distribution outputs sparse vectors with high probability. The standard LPN is obtained by sampling A as $A \leftarrow \{0,1\}^{n\times m'}$, i.e. a uniformly random binary matrix, and sampling the noise distribution as $\vec{e} \leftarrow \mathsf{Ber}_r^n(\{0,1\})$, i.e. the binary Bernoulli distribution where for all i, $\Pr[e_i = 1] = r$. Other distributions such as regular noise have been proposed [11], where $\vec{e}$ is the concatenation of t unit vectors of length N/t, to achieve improved running times for target applications.

The assumption comes in two equivalent forms, the "primal" as described above, and the "dual". The latter formulation considers the linear error correcting code $\mathcal{C}$ where A is the transpose of the parity check matrix of the linear

code $\mathcal{C}$. Let $H = A^{\mathsf{T}}$ and G be the parity and generator matrix of $\mathcal{C}$, the hardness of distinguishing $H^{\mathsf{T}} \cdot \vec{x} + \vec{e}$ from random is equivalent to the hardness of distinguishing $G \cdot (H^{\mathsf{T}} \cdot \vec{x} + \vec{e}) = G \cdot \vec{e} = \vec{e} \cdot G^{\mathsf{T}}$ from random (since $G^{\mathsf{T}} \cdot H = 0$).

The Linear Test Framework. Recent works[12,11,14,10] have built on the observation that, with a few exceptions, essentially applicable attacks fall within the so called *linear test framework.* These attacks have been based on Gaussian elimination and the BKW algorithm [17] and variants based on covering codes, information set decoding attacks [19,20], statistical decoding attacks [3,16,15], generalized birthday attacks, linearization attacks, attacks based on finding low weight code vectors [23], or on finding correlations with low-degree polynomials [2,9]. The linear test framework essentially states that for each of these distinguishers D, there exists a related adversary $\mathcal{A}$ that takes as input A and outputs a *test vector* $\vec{v}$ such that the distinguishing power of $D(A, \vec{b})$ is no better than $\vec{v} \cdot \vec{b}$ where $\vec{b}$ is the LPN sample.

The linear test framework is useful for two primary reasons. First is that it is expressive enough to capture all relevant attacks on LPN. Secondly, when LPN is instantiated with a code with high minimum distance d, one can prove that the advantage of a linear test distinguisher is negligible. If the weight of $\vec{v}$ is at most d, then the rows of A will be d-wise linearly independent and therefore $\vec{v} \cdot (A\vec{s} + \vec{e})$ will be uniformly random because $\vec{s}$ is uniformly random. Therefore, for the distinguisher to succeed, it must output a test vector $\vec{v}$ with weight greater than d. However, in this case, one can parameterize the noise distribution to be sufficiently high that $\vec{v} \cdot \vec{e}$ is close to uniform.

We note that there are a few notable exceptions to the linear test framework. These exceptions all take advantage of special algebraic structure within the code (e.g. Reed-Solomon) or noise distribution. However, none of these exceptions are directly applicable to our setting[10].

Pseudo Minimum Distance. The original formulation of the linear test framework focuses on the existence of the minimum weight codeword to argue the security of LPN. The recent work [10] made explicit the observation that it is possible for an LPN instance A to be secure per Definition 2 yet have small minimum distance[4]. In particular, the observation is that if codewords with small weight exist but are computationally infeasible to find, then LPN should remain secure. [10] defines *pseudo minimum distance* to be the weight of the smallest weight codeword that can be efficiently computed. Given that computing the minimum weight codeword is known to be NP-Complete, it is very likely that for some codes the difference between pseudo and actual minimum distance could be quite large, possibly even asymptotically different. [10] formalizes this idea ([10], Definition 3.12) and proposes the use of pseudo minimum distance when performing parameter selection.

Minimum Distance vs. Noise Rate. In this discussion, we have focused on the need for high [pseudo] minimum distance. This raises the question of

[4] Or more accurately, have a small enough LPN noise rate such that $\vec{e}$ does not sufficiently intersect a test vector with minimum distance weight with high probability.

exactly how high it needs to be for LPN to be secure. For any given minimum distance d, one needs to set the noise rate of the error vector $\vec{e}$ sufficiently high so that (linear) attacks with weight at least d are sufficiently likely to intersect with it. As such, one can also consider using a code with a smaller distance $d' < d$ and compensate for this by having a higher noise rate, or vice versa. [10] makes extensive use of this tradeoff and considers the use of relatively moderate minimum distance codes with a high noise rate. This has some impact on the communication complexity and running time of the final protocols.

LPN-friendly Codes. Many codes have been conjectured to result in secure LPN instantiations. As mentioned, the original formulation samples A as a uniformly random linear code. Alekhnovich [4] proposed sampling A as a sparse random matrix, i.e. a random LDPC code. Many works have also proposed LPN-friendly codes in the context of the NIST post-quantum completion [5,1,13] which were also used by [11]. It is believed that these codes offer strong security but fall short of the desired running time performance. The recent works of [14] and [10] have both proposed new classes of codes LDPC/Turbo code which aim for improved running times. With the exception of [14], all of these codes have instantiations which yield proofs of security in the linear test framework. However, we show that [14] is not LPN-friendly due to having small minimum distance.

2.5 Convolutional Codes

An important building block for the error correcting codes considered in this work are convolutional codes. These codes differ from the previously mentioned block codes in that they are capable of encoding an unbounded stream of data. A convolutional code maintains an internal state $\sigma_i \in \{0,1\}^m$ for each time step i. The length m of this state is referred to as the state size or memory size. There exists a state transition function ST that takes in the current time step i, current state σ_i and the next message bit x_i. ST outputs the codeword bit c_i and the new state σ_{i+1}. That is, for $i \in [n]$, $\quad (c_i, \sigma_{i+1}) = \mathsf{ST}(i, x_i, \sigma_i)$ where $\sigma_1 = 0$. More generally, one can consider a state transition function that takes in multiple message bits at a time and output multiple bits. In this way, one is able to achieve a non-rate 1 code. However, for our purposes, this definition will suffice.

All convolutional codes we consider will be linear and as such there exists a matrix $M_i \in \{0,1\}^{(m+1)\times(m+1)}$ such that $\quad \mathsf{ST}(i, x_i, \sigma_{i+1}) = (x_i || \vec{\sigma}_i) \cdot M_i$. The generator matrix for a convolutional code (for a fixed block size n) is then an upper triangular matrix $A \in \{0,1\}^{n\times n}$. Moreover, all of the convolutions we consider are *recursive* where $\sigma_i = c_{i-[m]}$ and therefore the convolution can be computed using an *update column vector* $\vec{\alpha}_i \in \{0,1\}^m$ such that $c_i = x_i + \vec{\alpha}_i \vec{c}_{i-[m]}$. That is, each codeword bit is a linear combination of the previous m codeword bits and the corresponding message bit.

The simplest convolutional code we consider is an accumulator. This code has a state size of $m = 1$ and for all i, $\vec{\alpha}_i = (1)$, each output $c_i = x_i + c_{i-1}$.

Looking forward, we will consider more complicated convolutional codes where $m = O(\log n)$ and $\vec{\alpha}_i$ are sampled at random from some distribution.

3 Minimum Distance of Silver Codes

We begin by presenting the negative result that the Silver codes presented in [14] do not achieve linear [pseudo] minimum distance as conjectured. These codes were specifically designed for high efficiency LPN. The core design criteria were high minimum distance and (near) linear time encoding. We demonstrate that the Silver codes have a minimum distance upper bounded by $mw^2 = O(\log^3(n))$, where $m \in \{47, 63\}$ is the memory size of their convolutional code and $w \in \{5, 11\}$ is the number of diagonals in their left matrix. While Silver gives concrete values for m, w, we assume based on their claims that asymptotically they grow logarithmically, which technically means they obtain a $O(n \log n)$ running time. Next, we present the necessary details on the Silver codes which enable our concretely efficient LPN distinguisher along with identifying structural weaknesses of their construction. On the positive side, we identify sever several good design choices in their construction which we will build off of in Section 5.

Silver Codes. Arguably, one reason that the sublinear minimum distance of Silver was not previously observed is due to the somewhat complicated presentation of the code itself. Here we will present a simplified version of the code that has several unnecessary details removed[5].

Silver is a linear systematic code in that the codeword $c \in \{0,1\}^n$ of message $\vec{x} \in \{0,1\}^k$ is composed of $\vec{c} = (\vec{x}||\vec{c'})$. Their encoder can be broken into two phases, *linear sums* L and a *recursive convolution* A with a state size of $m \in \{47, 63\}$, where $\vec{c'} = \vec{x}LA$. $L \in \{0,1\}^{k \times k}$ is an extremely sparse and highly structured matrix that is the addition of w rotations of the identity matrix. The convolution A is an upper triangular matrix with state size $m \approx 16$.

A recursive convolution code offers a more efficient encoding algorithm than naively multiplying by A. In particular, the ith bit of $\vec{y} = \vec{x}A$ is a linear combination of the previous m output bits and x_i, i.e. $y_i = x_i + (y_{i-1}, ..., y_{i-m}) \cdot \vec{\alpha}_i$. For our purposes, we can assume $\vec{\alpha}_i$ is uniformly random. An alternative description of A is with its parity check matrix which essentially consists of a width m diagonal band of zeros and ones defined as a function of the $\vec{\alpha}$ coefficients above.

Sublinear Minimum Distance. The state size m of the convolution immediately places an upper bound on the minimum distance of $I||A$, sometimes referred to as the *free distance* of A in the context of convolutional codes. Consider the following, at some index $x_i = 1$ and for $i' \in [i{+}1, i{+}m]$, set $x_{i'} := (x_{i-1}, ..., x_{i-m}) \cdot \vec{\alpha}$. At all other positions let $\vec{x}$ be zero. Such an $\vec{x}$ will result in the state of the convolution return to all zeros after m iterations. Here we refer to the state of the convolution as the last m outputs which in turn define the next output. As such, the minimum distance of $I||A$ is upper bounded by $m + 1$.

[5] These removals will not influence the minimum distance.

For Silver, it is not as simple to get the state of the convolution to return to zeros. Let us first focus on the code formed by the generator matrix $LA \in \{0,1\}^{k \times n}$. Let $q_1, ..., q_w \in [k]$ be the rotations of the identity matrix that form:

$$L = \sum_{j \in [w]} \mathsf{shift}(I_k, q_j).$$

As such $\vec{z} = \vec{x}L$ has the structure that $z_i = \sum_{j \in [w]} x_{i+q_j}$. For simplicity let $q_1 = 0$ and assume that the distances between any pair of q_j is greater than mw. Ostensibly, the logic behind Silver is that one is not able to find a value for $\vec{x}$ such that state of the convolution will quickly return to zero whenever it turns on. Of course one could always choose $x_i, ..., x_{i+m}$ such that the state of convolution at iteration i quickly goes to zero but the other $w-1$ locations that x_i is mapped to will remain at a non-zero state with high probability.

Unfortunately, this need not be the case. Let $x_i = 1$ and consider $x_{i+1}, ...x_{i+mw}$. Let the rest of $\vec{x}$ be zero. Due to the structure of the convolution the state of the convolution at iterations $i + mw + \{q_1, ..., q_w\}$ will be a linear function F of $x_{i+1}, ...x_{i+mw}$. Moreover, the total size of the convolution states at these w iterations will be mw. As such, assuming F is bijective then one can solve for $x_{i+1}, ..., x_{i+mw}$ such that $F(x_{i+1}, ..., x_{i+mw}) = 0^{mw}$.

Therefore, one can choose $\vec{x}$ such that after mw iterations the convolution will return to the zero state at all of the w iterations that x_i was mapped to. This will contribute at most mw^2 ones to the codeword while the rest is all zeros. Since the overall code is of the form $(\vec{x}||\vec{x}LA)$, the minimum distance will be at most $mw + mw^2$ where the first term is due to $\vec{x}$ itself. Therefore one can conclude the [pseudo] minimum distance is $O(\log^3(n))$. See Section D for additional comments on Silver.

4 Expand Accumulate Codes

Expand-accumulate codes (EA) were recently introduced by Boyle et al. [10]. Among other compelling features, EA offers concretely good encoding time and provable minimum distance under certain instantiations. To achieve better concrete performance [10] suggests the use of "aggressive" parameters which are not compatible with their proof of minimum distance but are plausibly secure. We show that these parameters fall short of their conjectured pseudo minimum distance by as much as $30\times$ for a code of size $k = 2^{20}$. It is unclear to us if this decrease results is a practical LPN distinguisher but, nonetheless, it is of concern and decreases their security margin. More generally, we show that for an EA code to have linear [pseudo] minimum distance, these parameters must grow with k. EA codes can be described using

- A sparse expander $B \in \{0,1\}^{k \times n}$, each row having approximate w weight.
- An accumulator matrix $A \in \{0,1\}^{n \times n}$.

Encoding is performed as $\vec{c} = (\vec{x}B)A$, i.e. $G = BA$. First the message $\vec{x}$ is expanded by B, i.e. $\vec{y} = \vec{x}B$. Concretely, each position y_i will be a sum of

approximately $w(k/n) \approx w/2$ entries in $\vec{x}$. Accumulating as $\vec{c} = \vec{y}A$ correspond to taking the prefix sum of each entry, i.e. $c_i = \sum_{j \in [i]} y_j$.

4.1 On the minimum distance of EA

[10] proposes several different distributions for B and error vector $\vec{e}$. [10] proved that their code for certain parameter regimes has high minimum distance, where B is sampled using a Bernoulli distribution. In particular, let $\mathsf{EA}(k, n, p_{\mathsf{w}}) = BA$ refer to the process of sampling $B \in \{0,1\}^{k \times n}$ where $B_{i,j} \leftarrow \mathsf{Ber}_{p_{\mathsf{w}}}$ with $p_{\mathsf{w}} = w/n$ for $w = \Omega(\log n)$. As such, the rows of B have expected weight w with $\Pr[B_{i,j} = 1] = p_{\mathsf{w}}$. Concretely, $w \approx 2.5 \ln n$ was proposed. Below we will review the proof of high minimum distance for this case. [10] also suggests more aggressive parameters with each row of B being randomly sampled with a small fixed hamming weight, e.g. $w = 7$. Focusing on the Bernoulli distribution, the sequence $(y_1, \ldots, y_n) = \vec{x}(BA)$ can be modeled as a n-step random walk (over the randomness of B) on a Markov chain with state space $\{0,1\}$. That is, at step i of the walk, the current state will be y_i.

In particular, one can visualize this process by focusing on the state of the accumulator. Instead of first computing $\vec{x}B$ and then accumulating the result, consider the scenario where the accumulator state at iteration i is updated with $\vec{x}\vec{B}_{\cdot i}$. Whenever this update is one, the single bit state of the accumulator will flip and equivalently we will have $y_i = y_{i-1} \oplus 1$. This formulation naturally lends itself to being modeled as a Markov chain with the accumulator state being the state space, i.e. $\{0,1\}$, and transition probability from $0 \to 1$ or $1 \to 0$ being $p_r = \Pr[\vec{x}\vec{B}_{\cdot i}]$. Additionally, recall that the column $\vec{B}_{\cdot i}$ is sampled as $\mathsf{Ber}^k_{p_{\mathsf{w}}}$ and therefore p_r is a function of $r = \mathsf{HW}(\vec{x}), p_{\mathsf{w}}$ when $\vec{B}_{\cdot i}$ is viewed as a random variable. Looking forward, we can bound the value of p_r using the Piling-up lemma (2). The proof perform a case analysis of $r = \mathsf{HW}(\vec{x})$ for $r = 1, r < \Omega(\frac{\log n}{n})$ and otherwise. Analyzing the behavior of the chain for each case, via its *spectral gap* (e.g. convergence rate) and then using an *expander Hoeffding bound* (concentration bound for Markov chains), it can be shown that the random walk $(y_1, \ldots, y_N)$ is unlikely to spend too much time on state 0, or equivalently, $\mathsf{HW}(\vec{x}G)$ is unlikely to be too small. In particular, [10] proved the following.

Lemma 1 (Core EA Lemma). *Let $k, n \in \mathbb{N}$, $k \leq n$. Fix $p_{\mathsf{w}} \in (0, \frac{1}{2})$, $\delta > 0$, $\beta = \frac{1}{2} - \delta$. Let $r \in n$, $x \in \{0,1\}^k$ s.t. $\mathsf{HW}(x) = r$, define $\xi_r = (1 - 2p_{\mathsf{w}})^r$, then:*

$$\Pr\left[\mathsf{HW}(\vec{x}G) < \delta n \,|\, G \leftarrow EA(k, n, p_{\mathsf{w}})\right] \leq 2 \exp\left(-2n\beta^2 \cdot \left(\tfrac{1-\xi_r}{1+\xi_r}\right)\right)$$

The final proof of minimum distance can then be achieved by taking a union bound over all $\vec{x} \in \{0,1\}^k$. For $w = \Omega(\log(n))$, EA shows that with probability at least $1 - \frac{1}{poly(n)}$ the code has minimum distance $\Omega(n)$, Concretely, EA suggest the use of $w \approx 2.5 \ln n$ which translates to $w = 36$ for $k = 2^{20}$. Alternatively, setting $w = \Omega(\log^2(n))$ results in linear minimum distance except with negligible probability $1 - O(n^{-\log(n)})$.

In further pursuit of better concrete efficiency, [10] proposes a set of aggressive parameters where w is set to be relatively low. In particular, for code sizes $k \in \{2^{20}, 2^{25}, 2^{30}\}$, [10] proposes the use of $w \in \{7, 9, 11\}$. For each of these 9 combinations, [10] conjecture concrete lower bounds on the pseudo minimum distance of the code (see Figure 9 [10]). The pseudo minimum distances being conjectured are significantly higher than what can be obtained from their proof of minimum distance. EA reconcile this difference by noting that their proof likely has some slack and with the notion of *pseudo minimum distance* of a code, see Section 2.4.

Building on the notion of pseudo minimum distance and intuition from their proof, [10] goes on to conjecture that if they have small pseudo minimum distance, then with high probability these will correspond to weight 1 messages, i.e. a row of $G = EA$ will have small weight. They then sample G 100 times and for $n \in \{2^9, 2^{10}, ..., 2^{15}\}$ and record the minimum weight row. From these they extrapolate that the minimum distance goes as δn where $\delta = 0.4633e^{-0.101 \log_2(n)}$ for $w = 7$. Moreover, they give an informal conjecture that finding a lower weight codeword is hard. Next we show that this is false. In particular, we give an improved analysis that shows how to find higher weight messages that result in lower weight codewords. For example, with a similar number of samples of G we were able to find codewords with a δ 30× smaller than conjectured.

4.2 Smaller Weight EA Codewords

We present a more general analysis that will in fact also apply to our construction. This analysis borrows ideas developed for proving that certain Turbocodes have sublinear distance [6]. Let m be the state size of the convolution, e.g. $m = 1$ for EA. The intuition of the analysis is that when w is small, one can find a small number of small regions over $[n]$ such that many rows of B are non-zero exclusively in the regions. The state space of the accumulator at the *end* of these w regions has a combined size of 2^{mw}. Therefore, if one can find mw rows of B that are non-zero exclusively in the regions, then there must exist as assignment to the corresponding message bits such that the state of the accumulator is zero at the end of these regions. Let x be zero outside this assignment and therefore the overall weight of the codeword can at most be the total size of these small regions. For the special case of an accumulator with $m = 1$, it suffices to find two rows of B that map to the same w regions. Next, we describe the process of finding these regions and show that it implies a sublinear minimum distance when w is a constant and m is sublinear. For $i \in [k], j \in [w]$, let $\sigma_j(i) \in [0, n)$ denote the j'th position that x_i was mapped to by B_i when using zero indexing. That is $B_{i,\sigma_j(i)+1} = 1$ and otherwise is zero. For some parameter $\beta \ll n$, divide the range $[n]$ up into β evenly sized regions, where each region has size $\mu = n/\beta$. For each row B_i, let us define its *signature* as $S_i = \{\sigma_1(i)/\mu, ..., \sigma_w(i)/\mu\}$. Let the domain of S_i be $\mathbb{S}$ and note that $|\mathbb{S}| \leq \beta^w$ since $\sigma_j(i) \in [0, \beta - 1]$. By the pigeon hole principle, there must exist a $U \subset [k], T \in \mathbb{S}$ such that $\forall u \in U, S_u = T$ and $|U| \geq k/\beta^w$. Wlog, let us assume $|T| = w$. Let $t_1, ..., t_w \in [n]$ index the end of these w regions in T. The state space of the convolution at these steps

has a combined size of 2^{mw}. There are $2^{|U|}$ assignments to the message bits x_u for $u \in U$ and therefore if $|U| > mw$, then there must exist two distinct input strings $x, y \in \{0,1\}^k$ which are zero outside U such that their states at steps $t_1, ..., t_w$ are equal. $x + y$ is then encoded into a codeword with weight at most $\mu w = nw/\beta$. Therefore if we set

$$\beta = \left(\frac{k}{wm}\right)^{1/w} \implies |U| \geq \frac{k}{\left((\frac{k}{wm})^{1/w} - 1\right)^w} \geq \frac{k}{\left(\frac{k}{wm}\right)} = wm,$$

$|U|$ will be sufficiently large and therefore x, y must exist s.t. $\mathsf{HW}(x + y)$ is sublinear. Moreover, for EA, A is an accumulator where all transitions are the same and having $m = 1$, the size of U need only be 2. Concretely, for $k = 2^{20}, w = 7$ we can set $\beta = 7.5$ and obtain a minimum distance upper bound of δn where $\delta = 0.93$. Increasing k to 2^{30} and setting $\beta = 18.6$, we obtain $\delta = 0.38$. These upper bounds on δ are significantly larger than those suggested by the EA empirical analysis. However, these bounds hold with probability 1 and therefore are not tight on average.

To get more accurate average case bounds, we implement the analysis and find that δ decreases significantly. We extend the basic analysis above to consider signatures S_u that are almost equal to T. For example, if S_u is equal to T except in one position it is off by 1. By setting $\beta = 10$, we were able to find that most codes with $k = 2^{20}, w = 7$ have minimum distance at most $\delta = 0.01$ and as small as 0.002 over a few hundred trials. That is a reduction in minimum distance of $6\times$ to $30\times$ over what was reported by [10]. We observed similar trends for other parameters.

Looking forward, one should note that the effectiveness of this search falls off dramatically as we increase the state size m. Even for moderate such as $m = 25, w = 7$, the theoretical bound on δ passes 1 and therefore is not meaningful.

5 Expand Convolute Codes

In this section, we introduce expand-convolute codes, which are defined by the product $G = BC$ for a sparse expanding matrix B and a convolution matrix C. We conjecture that the LPN problem is hard to solve for this matrix ensemble and provide theoretical evidence for this conjecture by demonstrating that it resists attacks in the linear test framework.

5.1 Defining Expand Convolute Codes

Expand Matrix. For a ring $\mathcal{R}$ and parameters $w, k, n \in \mathbb{N}$ with $w \ll k \leq n$, an expanding matrix $B \in \mathcal{R}^{k \times n}$ is an $k \times n$ matrix over $\mathcal{R}$ with each row having (approximately) w non-zero entries. In our provable instantiation, every entry of B is sampled as $B_{i,j} \leftarrow \mathsf{Ber}_{p_\mathsf{w}}(\mathcal{R})$, where $p_\mathsf{w} = \frac{w}{n}$. Fixed row weight w is also conjectured to have similar minimum distance [10].

Convolute Matrix. For a ring $\mathcal{R}$ and parameters $m, n \in \mathbb{N}$ with $m \leq n$, a convolutional code generator matrix $C \in \mathcal{R}^{n \times n}$ is an $n \times n$ upper-triangular matrix over $\mathcal{R}$ with state size m. That is, C has 1's on its diagonal and entries below the diagonal being some linear combination of the following m columns. Concretely, for any $i \in [n]$, there exist $\alpha_{i,1}, \ldots, \alpha_{i,m} \in \mathcal{R}$ such that

$$C_{\cdot,i} = \overleftarrow{\mathsf{id}}_{i,n} + \sum_{j \in [m]} \alpha_{i,j} C_{\cdot,i+j}$$

where $\overleftarrow{\mathsf{id}}_{i,n}$ is the $n \times 1$ column vector over $\mathcal{R}$ whose i'th entry is 1 and all other entries are 0. In our provable instantiation, C is to sampled such that each $\alpha_{i,j} \leftarrow \mathsf{Ber}_{p_c}(\mathcal{R})$, for $p_c = \frac{1}{2}$. The expand convolute generator matrix is defined as $G = BC$. We denote this sampling procedure by $\mathsf{ECGen}(w, m, k, n, p_c, \mathcal{R})$.

5.2 The EC-LPN Assumption

In this work, we provide a new (dual) LPN-type assumption connected to expand convolute codes which we term EC-LPN. It is obtained by specializing Definition 1 to the case where the code generation algorithm samples $G \leftarrow \mathsf{ECGen}$.

Definition 2 (EC-LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_n(\mathcal{R})\}_{n \in \mathbb{N}}$ denote a family of efficiently sampleable distributions over a ring $\mathcal{R}$, such that for any $n \in \mathbb{N}$, $\mathsf{Im}(\mathcal{D}_n(\mathcal{R})) \subseteq \mathcal{R}^n$. For a dimension $k = k(\kappa)$, number of samples $n = n(\kappa)$, expansion weight $w = w(\kappa) \in [n]$, state size $m = m(\kappa) \in [n]$, convolving density $p_c = p_c(\kappa) \in [0, 1]$ and ring $\mathcal{R} = \mathcal{R}(\kappa)$, the $(\mathcal{D}, \mathcal{R})$-EC-LPN$(w, m, k, n, p_c)$ assumption states that*

$$\{(G, \overleftarrow{b}) \text{ s.t. } G \leftarrow \mathit{ECGen}(w, m, k, n, p_c, \mathcal{R}), \overleftarrow{e} \leftarrow \mathcal{D}_n(\mathcal{R}), \overleftarrow{b} \leftarrow G\overleftarrow{e}\}$$
$$\overset{c}{\approx} \{(G, \overleftarrow{b}) \text{ s.t. } G \leftarrow \mathit{ECGen}(w, m, k, n, p_c, \mathcal{R}), \overleftarrow{b} \leftarrow \mathcal{R}^k\}.$$

In order to provide evidence for the EC-LPN assumption, we will show that it is secure against linear tests, at least when $\mathcal{R} = \mathbb{F}_2$, $w, m = \Theta(\log n)$ and $p_c = \frac{1}{2}$. To do this, it suffices to show that the minimum distance of the expand convolute code is large (with high probability). To that end, we try to bound the probability that a message vector is mapped to a codeword of low weight using the roadmap below:

1. We consider a fixed message $\vec{x} \in \mathbb{F}_2^k$ and imagine revealing the coordinates of the random vector $\vec{x}G$ one at a time, where $G \leftarrow \mathsf{ECGen}$, and observe that this can be viewed as a random walk on a Markov chain (Section 5.2.1).
2. The state space of this Markov chain exactly corresponds to the various possible values of the internal state σ of the code. In general, this would be of size 2^m. However, when $p_c = \frac{1}{2}$, it turns out that the state space shrinks to be of size $m + 1$, which is easier to analyze (Section 5.2.2).
3. However, this Markov chain is not reversible and hence we are not in a position to apply the Hoeffding bound to characterize it (Section 5.2.3).

4. We instead resort to analyzing a much simpler reversible chain (Section 5.2.4 and Section 5.2.6), and comparing the behavior of the two chains via a coupling argument (Section 5.2.5).

We formalize all these details ahead, the security analysis in Section 5.2.7.

5.2.1 Understanding the Markov chain For parameters $w, m, k, n \in \mathbb{N}$ with $w, m, k \leq n$ and $0 \leq p_c \leq 1$, let $G = BC$ be $G \leftarrow \mathsf{ECGen}(w, m, k, n, p_c, \mathbb{F}_2)$. Consider a fixed message $\vec{x} \in \mathbb{F}_2^k$. Define $R = \frac{k}{n}$ to be the rate of the code. Consider the codeword $\vec{y} \in \mathbb{F}_2^n$ given by $\vec{y} = \vec{x}G$. We see that the bits of y can be computed iteratively (in reverse) as follows:

$$\begin{aligned} y_1 &= \vec{x}B_{\cdot,1} \\ y_i &= \vec{x}B_{\cdot,i} + \sum_{j \in [m]} \alpha_{i,j} y_{i+j} \qquad\qquad ,\forall i \in [2, n] \end{aligned}$$

We can view this as a recursive process with an internal state σ_i at step i given by $\vec{\sigma}_i = (y_{i-1}, \ldots, y_{i-m})$. In step 1, let $y_1 = \vec{x}B_{\cdot,1}$ and add y_1 to $\vec{\sigma}_2$ as $\vec{\sigma}_2 = (0, \ldots, 0, y_1)$. In step $i \in [2, n]$, let $y_i = \vec{x}B_{\cdot,i} + \vec{\alpha}_i \cdot \vec{\sigma}_i$. Now, note that

$$\Pr[y_i = 1 | \vec{\sigma}_i] = \Pr_{B_{\cdot,i}}[\vec{x}B_{\cdot,i} = 0] \cdot \Pr_{\vec{\alpha}_i}[\vec{\alpha}_i \vec{\sigma}_i = 1] + \Pr_{B_{\cdot,i}}[\vec{x}B_{\cdot,i} = 1] \cdot \Pr_{\vec{\alpha}_i}[\vec{\alpha}_i \sigma_i = 0]$$

The above follows from a few observations. Firstly, B and C are sampled independently in the sampling of G, which means that $B_{\cdot,i}$ and $\vec{\alpha}_i$ are independent. Secondly, $\vec{\sigma}_i$ is independent of both $B_{\cdot,i}$ and $\vec{\alpha}_i$. To estimate the probabilities above, we recall the piling-up lemma below.

Lemma 2 (Piling-up Lemma). *For any $0 < q < \frac{1}{2}$ and $t \in \mathbb{N}$, given t random variables $(X_1, \ldots, X_t)$ that are i.i.d. according to Ber_q, it holds that*

$$Pr\left[\bigoplus_{i=1}^{t} X_i = 0\right] = \frac{1}{2} + \frac{(1-2q)^t}{2}$$

Let $r = \mathsf{HW}(\vec{x})$. By Lemma 2, $\Pr_{B_{\cdot,i}}[\vec{x}B_{\cdot,i} = 0] = \frac{1}{2} + \frac{(1-2p_{\mathsf{w}})^r}{2}$ where $p_{\mathsf{w}} = \frac{w}{n}$. By Lemma 2, we also have that $\Pr_{\vec{\alpha}_i}[\vec{\alpha}_i \vec{\sigma}_i = 0] = \frac{1}{2}$ as $p_c = \frac{1}{2}$. We note, however, that the last equation above only holds as long as $\vec{\sigma}_i \neq (0, \ldots, 0)$. Indeed, $\Pr_{\vec{\alpha}_i}[\vec{\alpha}_i \sigma_i = 0] = 1$ if $\sigma_i = (0, \ldots, 0)$. Thus, $\Pr[y_i = 1 | \vec{\sigma}_i \neq 0] = \frac{1}{2}$ and $\Pr[y_i = 1 | \vec{\sigma}_i = 0] = \frac{1}{2} - \frac{(1-2p_{\mathsf{w}})^r}{2}$. We define $p_r := \frac{1}{2} - \frac{(1-2p_{\mathsf{w}})^r}{2}$

We now begin to view the recursive encoding process described before as a walk on a Markov chain whose state space is the set of all possible values of the internal state σ. We begin at $\vec{\sigma}_1 = (0, \ldots, 0)$. The random walk of n steps then moves as $\vec{\sigma}_2, \ldots, \vec{\sigma}_n$. Since $\vec{\sigma}_i$ are binary vectors of size m, the size of the state space is 2^m. At any step $i \in [n]$, we move from $\vec{\sigma}_{i-1}$ to $\vec{\sigma}_i$. Note that given $\vec{\sigma}_{i-1}$, $\vec{\sigma}_i$ can be only one of two different values, depending on whether y_i is 0 or 1.

We denote these two values by $\vec{\sigma}_i^{(0)}$ and $\vec{\sigma}_i^{(1)}$. From our above calculation,

$$\Pr\left[\vec{\sigma}_{i-1} \to \vec{\sigma}_i^{(1)} | \vec{\sigma}_{i-1} \neq 0\right] = \frac{1}{2}$$
$$\Pr\left[\vec{\sigma}_{i-1} \to \vec{\sigma}_i^{(1)} | \vec{\sigma}_{i-1} = 0\right] = p_r$$

5.2.2 Shrinking the Markov chain It turns out that the above Markov chain can be shrunk to have only $m+1$ states as opposed to 2^m. This can be done by combining states in the original chain. We note that this possibility arises specifically because we have considered the case of $p_c = \frac{1}{2}$ which induces a lot of symmetry in the transitions.

We can group states based on the suffix of the m bits representing the state. In particular, all states whose last bit is 1 can be combined. All states whose last two bits are 10 can be combined. All states whose last three bits are 100 can be combined. And so on, until the last remaining state is all zeros 0^m state. We will name these states $1, 0_1, 0_2, \ldots, 0_{m-1}, 0_m$. Note that there are $m+1$ of them. This chain is described in Figure 1. From our calculations above, we see that we have only three types of transitions:

$$\Pr\left[0_m \to 1\right] = p_r, \qquad \Pr\left[0_m \to 0_m\right] = 1 - p_r$$

and all other transitions are with probability $\frac{1}{2}$.

To see the correctness, consider the following. Suppose that at a certain point in the random walk, we have internal state σ whose last $\ell+1$ bits are 10^ℓ and hence we are in the shrunk state 0_ℓ. After one more step in the walk, the internal state will be either $10^{\ell+1}$ or $10^\ell 1$. This means that we will either be in the shrunk state $0_{\ell+1}$ or 1. The transition probabilities of both of these are $\frac{1}{2}$, as long as $\ell < m$. If $\ell = m$, after one more step in the walk, the internal state will be either 0^m or $0^{m-1}1$. This means that we will either be in the shrunk state 0_m or 1. The transition probabilities of these are $1 - p_r$ and p_r respectively.

In other words, the Markov chain does not need to keep tract of the actual state of the convolution (with its 2^m possible states) but instead just if there has been a 1 transition in the last m transitions. Looking forward we will effectively shrink the chain to two via a Markov chain coupling.

5.2.3 Random Walks and Minimum Distance Recall that our overall goal was to show that the minimum distance of G is large (with high probability) and that we are attempting to show this by arguing that for any message $\vec{x}$, the weight of $\vec{y} = \vec{x}G$ is large (with high probability). It is easy to see that the weight of $\vec{y}$ is the number of steps in the n step random walk in the Markov chain we have in Figure 1 that we are in the state 1 (this is because the last bit of the state corresponds to the codeword bit). One strategy to estimate the amount of time we spend in a state in a random walk on a Markov chain is to estimate the stationary distribution of the chain and then use a concentration bound of some sort, for instance, the expander Hoeffding bound (Theorem 1). The Markov chain

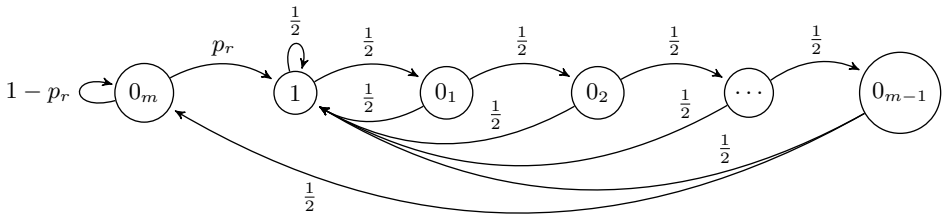

Fig. 1. The $m+1$ state Markov chain representing the encoding process.

we have in Figure 1 is unfortunately not reversible, and bounds for irreversible Markov chains are unfortunately not tight enough for our purposes. Therefore, we take a different approach.

5.2.4 A Reversible Markov chain We will study a different, related, but reversible Markov chain described in Figure 2. This chain has only two states, namely a 0 state and a ? state, and the following transition probabilities:

$$\Pr[0 \to 0] = 1 - p_r, \qquad \Pr[0 \to ?] = p_r$$

$$\Pr[? \to 0] = 2^{-(m+\theta_m)}, \qquad \Pr[? \to ?] = 1 - 2^{-(m+\theta_m)}$$

At a high level, the 0 state in this reversible chain is meant to capture the 0_m in our previous chain, and the ? state in the reversible chain is meant to *emulate* all the other states of the other chain. Unlike our previous shrinking, this isn't a transformation that preserves functionality. Indeed, the chains in Figures 1 and 2 are different. However, posit that they are very closed related for the following reason. Firstly note that the states 0_m and 0 in the two chains are identical in a sense. Furthermore, in the chain in Figure 1, the shortest path that takes you from 1 back to 0_m is a of length m, each transition having a probability of $\frac{1}{2}$, while in the chain in Figure 2, the transition probability from ? to 0 is $2^{-(m+\theta_m)}$ for some $\theta_m > 0$. Thus, in this reversible Markov chain, leaving the 0 state is about as hard as in our other chain, and hence we hope that we can relate the time spent in states 1 and ? on random walks on the two chains. If we succeed in doing so, then we can use the expander Hoeffding bound to find the time spent in ? in the reversible chain and use that to argue about the time spent in 1 in the other chain, which is what we set out to do.

One final note before we move ahead is regarding the ?. Firstly, the reason for using ? instead of 1 is because the ? emulates states $1, 0_1, \ldots, 0_{m-1}$ in the other chain, and we are actually only interested in the time spent in 1 in the other chain. Looking ahead, we will come up with a way of estimating what fraction of time spent in ? will account for the time spent in 1.

5.2.5 Relating the two chains : A Coupling *Recall that our strategy is to relate the behavior of the irreversible chain in Figure 1 with that of the reversible*

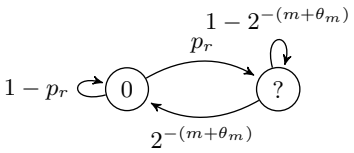

Fig. 2. The 2-state reversible chain which we compare to.

chain in Figure 2, which is much easier to study.

We will consider random walks on both of the Markov chains. In the case of the irreversible chain, we count the number of time steps spent in state 1. In the case of the reversible chain, every time we are in the ? state, we flip a coin and with probability $\frac{1}{2}$, we count it. That is, in expectation, we count approximately half the number of time steps spent in state ?. In this section, we will relate these counts by proving the theorem below.

Theorem 2. *Let n denote the length of the random walks performed on the chains in Figures 1 and 2, where $m \geq \log n + 2$. Starting from state 0_m of the irreversible chain (Figure 1), let X_i be the indicator of being in state 1 at step i. Starting from state 0 of the reversible chain (Figure 2), let Y_i be the indicator of being in state ? at step i and then uniformly mapping ? to $\{0,1\}$ (with probability $\frac{1}{2}$). Fix $\delta \in [0,1]$ and $\hat{k} > 0$. Then, there exists $\theta_m \in [0,1)$ such that*

$$\Pr\left[\sum_{i\in[n]} X_i \leq \delta n - \hat{k}(m-1)\right] \leq \frac{1}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right]$$

where $\tilde{\delta}_r = \frac{\hat{k}}{n \cdot 2^{-(m+\theta_m)} \cdot p_r}$.

Conjecture: Theorem 2 holds for all $m \geq 2$.

In order to prove Theorem 2, we will induce a coupling on the random walks on the two Markov chains (Lemma 7). In order to perform the coupling, we walk through a series of lemmas that estimate the required probabilities. We begin with bounding the probability of having a run of length i without returning to the zero state (0 or 0_m)

Lemma 3 (Reversible Run of ?'s). *Starting from state ? in the reversible chain (Figure 2), let q_i to be the probability of going to state 0 for the first time exactly at step $i+1$. We have $q_i = \left(1 - 2^{-(m+\theta_m)}\right)^i \cdot 2^{-(m+\theta_m)}$*

Proof. Starting at state ?, the only such possible path stays in the ? for i steps and transitions from state ? to state 0 in the last step. □

Lemma 4 (Irreversible Run of 1's). *Starting from state* 1 *in the irreversible chain (Figure 1), let* p_i *to be the probability of going to state* 0_m *for the first time exactly at step* $i + m$*. Then for* $i \in [n]$*,* $m \geq \log n + 2$ *and* $\theta_m = 0$*,*

$$p_i < q_i$$

Conjecture: For all $m \geq 2$, there exists $\theta_m \in (0, 1)$ such that for all $i \geq 1$,

$$\sum_{j \in [i]} p_j < \sum_{j \in [i]} q_j$$

Proof. See Appendix B.1. □

We are now going to define our coupling. To aid in the description of the coupling, we first describe two processes (Processes 1 and 2) that will faithfully perform random walks on the two chains (Lemmas 5 and 6). In particular, we will first re-characterise the process of performing a random walk to sample the walk as several sub-walks (see Appendix C). First we will define the function S_0 that will sample the size of the sub-walk with state transitions $0_m \to \ldots \to 0_m \to 1$. Then we will define the function S_1 that will sample the size of the sub-walk with the form $1 \to \ldots \to 0_{m-1} \to 0_m$, that is, start at state 1 and end once state 0_m is reached. Such walks might visit state $1, 0_1, \ldots, 0_{m-1}$ many times but 0_m only once. Constructing the overall walk can then be achieved by alternating calls to S_0 and S_1 (along with sampling a specific sub-walks that is consistent with S_1).

Process 1 (Irreversible Chain)**.** *For each* $i \in \mathbb{N}$*, let* $\tau_i = (1 - p_r)^{i-1} p_r$ *and* $\eta_i = \sum_{j=1}^{i-1} \tau_j$*. So, we have* $\eta_1 = 0$*,* $\eta_2 = \tau_1$*,* $\eta_3 = \tau_1 + \tau_2$*,* $\ldots$*,*

$$\sum_{i=1}^{\infty} \tau_i = p_r \sum_{i=0}^{\infty} (1 - p_r)^i = \frac{p_r}{1 - (1 - p_r)} = 1$$

That is, $\eta_i \to 1$ *as* $i \to \infty$*. Also,* η_i *is trivially monotonically increasing. Define the monotonically increasing step function* $S_0 : [0, 1]_{\mathbb{R}} \to \mathbb{N}$ *where* $S_0(0) = 1$ *and for* $p \in (0, 1]_{\mathbb{R}}$*,* $S_0(p) = k$ *such that* $p \in (\eta_k, \eta_{k+1}]$*.*

Define $\zeta_1 = 0$ *and* $\zeta_i = \sum_{j=1}^{i-1} p_j$ *for* $i \in \mathbb{N}$*. We also have*

$$\sum_{j=1}^{\infty} p_i = 1$$

That is, $\zeta_i \to 1$ *as* $i \to \infty$*. Also,* ζ_i *is trivially monotonically increasing. Define the monotonically increasing step function* $S_1 : [0, 1]_{\mathbb{R}} \to \mathbb{N}$ *where* $S_1(0) = m$ *and* $S_1(p) = k + m - 1$ *such that* $p \in (\zeta_k, \zeta_{k+1}]$*.*

Let $\dot{X}$ *be the random variable denoting the output of Algorithm 1 (see Appendix B.5) which on input* S_0, S_1 *samples a walk by alternating calls to* S_0, S_1 *and constructing sub-walks of that length.*

Lemma 5 (Random walk on irreversible Markov chain). *Let $\tilde{X}$ be the random variable described by Process 1. Then $\tilde{X}$ follows the same distribution as a random walk on the irreversible chain (Figure 1).*

See Appendix B.2 for a proof of Lemma 5.

Process 2 (Reversible Chain)**.** *Let $S'_0 = S_0$. Define $\iota_1 = 0$ and $\iota_i = \sum_{j=0}^{i-1} q_j$ for $i \in \mathbb{N}$. We also have $\sum_{j=1}^{\infty} q_j = 1$ That is, $\iota_i \to 1$ as $i \to \infty$. Also, ι_i is trivially monotonically increasing. Define the monotonically increasing step function $S'_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$ where $S'_1(0) = 1$ and $S_1(p) = k$ such that $p \in (\iota_k, \iota_{k+1}]$. Let $\tilde{Y}$ be the random variable denoting the output of Algorithm 2 (see Appendix B.5) on input S'_0, S'_1.*

Lemma 6 (Walk on reversible chain). *Let $\tilde{Y}$ be the random variable described by Process 2. Then $\tilde{Y}$ follows the same distribution as a random walk on the reversible chain (Figure 2), where the ?'s are set to 1 or 0 with probability $\frac{1}{2}$.*

Proof. Similar to the proof of Lemma 5. □

We are now ready to define our coupling. We will essentially couple Process 1 and Process 2 that perform random walks on the irreversible and reversible chains, respectively, by having them use the same random coins, which we recall are $(\mathsf{p}_0, \ldots, \mathsf{p}_*), (\mathsf{q}_0, \ldots, \mathsf{q}_*)$. We will use this coupled way of producing random walks on the two chains to compare the hamming weights of the random walks produced by the two processes (Lemma 7). Since $S_0 = S'_0$, most of the work in the proof of Lemma 7 is with regard to relating S_1 and S'_1. In Figure 3 we give an illustration of how S_1 and S'_1 partition the $[0,1]$ interval, annotated with all the possible sub-walks associated with each of the segments. For example, in the segment $p \in (\iota_1, \iota_2]$, $S'_1(p) = 1$, and hence a sub-walk on the reversible chain would correspond to a single ?, while in the segment $p \in (\zeta_1, \zeta_2]$, $S_1(p) = m$, and hence a sub-walk on the irreversible chain would correspond to 10^{m-1}, i.e., the walk $1 \to 0_1 \to \ldots 0_{m-1} \to 0_m$.

Recall that $\iota_i = \sum_{j=1}^{i-1} q_j$ and $\zeta_i = \sum_{j=1}^{i-1} p_j$. From Lemma 4, for all $i \in [n]$, $m \geq \log n + 2$ and $\theta_m = 0$, $p_i < q_i$, and hence $\iota_i < \zeta_i$ for all $2 \leq i \leq n$ (from our conjecture, for all $m \geq 2$, there exists $\theta_m \in (0,1)$ such that for all $i \geq 1$, $\sum_{j \in [i]} p_j < \sum_{j \in [i]} q_j$, i.e., $\iota_i < \zeta_i$ for all $i \geq 2$). The dotted lines materialize the Lemma 4 (and our conjecture). This picture can be used to aid in our conclusion that $\forall p \in [0,1]$, $S_1(p) - (m-1) \geq S'_1(p)$ (where the $m-1$ corresponds to the additional 0^{m-1} that strings annotating the partition of S_1 have), although we will provide a formal proof of this in Lemma 7. This conclusion will allow us to prove that with high probability, the hamming weights of the random walks produced by the two processes cannot be that different, in particular, that the hamming weight of the random walk produced by Process 1, the process of interest, cannot be much smaller than the hamming weight of the random walk produced by Process 2, which we will lower bound in Section 5.2.6.

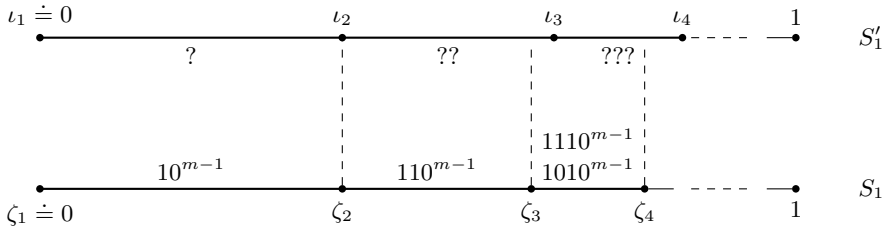

Fig. 3. Example of partitioning $[0, 1]$ by S_1 and S'_1 *(not at scale).*

Lemma 7 (Coupling). *Consider the random variable Z which outputs $(Z_{\tilde{X}}, Z_{\tilde{Y}})$ by following Processes 1 and 2 run on the same randomness $\mathsf{p}_0, \ldots, \mathsf{p}_*$ and $\mathsf{q}_0, \ldots, \mathsf{q}_*$. Fix $\hat{k} > 0$. Define $\tilde{\delta}_r = \frac{\hat{k}}{n \cdot 2^{-m} \cdot p_r}$. Then, we have with probability at least $1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)$:*

$$HW(Z_{\tilde{X}}) \geq HW(Z_{\tilde{Y}}) - \hat{k}(m-1)$$

Proof. The proof of this comes in two parts, first we show that the non-zero sub-walks (of length S_1, S'_1) will have the property that the reversible chain (S'_1) will be sufficiently shorter. As such, produce fewer ones. Secondly, because the reversible chain's non-zero sub-walks are shorter it will need to perform more sub-walks to reach a walk length of n. We show that this is unlikely to produce more than $k(m-1)$ additional ones.

Note that the states 0_m and 0 are identical, and S_0 and S'_0 are partitioned similarly. Since we use the same randomness for both, the draws on S_0 and S'_0 will give the same number of 0s in both $Z_{\tilde{X}}$ and $Z_{\tilde{Y}}$. We also use the same randomness to determine the number of 1s inside the convolution (portion of the walk spent outside the 0_m state) in the irreversible chain and to decide on the real value of ? in the reversible chain. In particular, both processes sample the $\vec{t}$ vectors using randomness $\mathsf{q}_1, \ldots, \mathsf{q}_*$. Therefore, if we concatenate all such $\vec{t}$, they will produce the same string with the exception that the irreversible chain sometimes overwrites the state as a one. However, this only increases the inequality and therefore safe to ignore.

We show that for the same p, we always have $S_1(p) \geq S'_1(p) + (m-1)$. We have $|S_1(0)| \geq |S'_1(0)| + (m-1)$ by definition. We have $S_1(p) = k + (m-1)$ for k such that $p \in (\zeta_k, \zeta_{k+1}]$ and $S'_1(p) = k$ for k such that $p \in (\iota_k, \iota_{k+1}]$. Consider $p \in (\zeta_k, \zeta_{k+1}]$ for some $k \in \mathbb{N}$. The claim is true as long as $p \leq \iota_{k+1}$. We thus show that $\zeta_{k+1} \leq \iota_{k+1}$. We have

$$\zeta_{k+1} = \sum_{j=0}^{k-1} p_j < \sum_{j=0}^{k-1} q_j = \iota_{k+1}$$

as we can bound the sum term by term provably when $m \geq \log n + 2$ from Lemma 4, and bound the sums conjecturally when $m \geq 2$. We thus have that $\forall p, S_1(p) \geq S'_1(p) + (m-1)$.

However, because a draw on S_1 outputs a sequence with a trailing $m-1$ 0's, we may have more draws on S_1' than on S_1. Once the $Z_{\tilde{X}}$ process has ended, it remains to draw for $Z_{\tilde{Y}}$ a number of steps at most equal to the number of times we have reached state 1 for the first time starting from state 1 and going through 0 exactly once. Let us denote by E the Bernoulli random variable for which a success is looping through the simplified chain, we thus have $Pr[E] = \frac{p_r}{2^{m+\theta_m}}$. We now consider $\tilde{E}$ to be the binomial random variable which consists of n independent trials of E. We have $\mathbb{E}[\tilde{E}] = \mu = n2^{-(m+\theta_m)}p_r$. A Chernoff bound gives us that $\Pr\left[\tilde{E} \geq (1+\delta)\mu\right] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$. So by setting $\tilde{\delta}_r = \frac{\hat{k}}{\mu}$, we have

$$Pr\left[\tilde{E} \geq \mu + \hat{k}\right] \leq \exp\left(-\frac{\tilde{\delta}_r\hat{k}}{2+\tilde{\delta}_r}\right)$$

□

We are now ready to prove Theorem 2. Due to lack of space, we defer the proof to Appendix B.3.

5.2.6 Analysis of the reversible chain We now formally analyze the behavior of a random walk on the reversible chain and the conversion of ?'s to 1's in order to bound $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right]$. This gives rather tedious expressions that are hard to get a handle on analytically (but they do give significantly better parameters with empirical estimation). And so, we end this section by analyzing another reversible chain which gives bounds that are much more amenable to analytical manipulation.

Bounds for empirical estimation. We are trying to bound $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right]$ from before, namely, the probability that a random walk on the reversible chain does not produce enough 1's. We bound this by first looking at the number of time steps spent in the ? state and then bounding how many of them get converted to 1's with probability $\frac{1}{2}$. It is easy to check that the 2-state chain is irreducible. One can easily estimate the stationary distribution $\vec{\pi}_r$ and the second largest eigenvalue λ_r as

$$\vec{\pi}_r = \left(\frac{2^{-(m+\theta_m)}}{p_r + 2^{-(m+\theta_m)}}, \frac{p_r}{p_r + 2^{-(m+\theta_m)}}\right)$$

$$\lambda_r = 1 - p_r - 2^{-(m+\theta_m)}$$

We can also then check the detailed balance conditions to ascertain that the chain is also reversible.

Time spent in the ? state. To bound the time spent in the ? state, we use the corollary of the expander Hoeffding bound for non-uniform starting distributions. Note that the reversible chain is also finite and irreducible. Therefore, the expander Hoeffding bound is applicable. If we consider f such that $f(0) = 0$ and $f(?) = 1$, $N_?$ counts the number of time steps the random walk spends in the ?

state. In this case, $\mu_r = \vec{\pi}_{r,?}$ and $\tilde{\lambda}_2 = \lambda_r$. We consider a random walk of length n that starts at the $x_0 = 0$ state. From the corollary of the expander Hoeffding bound, we have that for any $\epsilon > 0$,

$$\Pr[N_? < n\mu_r - \epsilon] \leq \left(1 + 2^{m+\theta_m} p_r\right) \exp\left(-2\frac{\epsilon^2}{n} \cdot \frac{1-\lambda_r}{1+\lambda_r}\right)$$

Let $\gamma \in [0, \mu_r)_{\mathbb{R}}$ be a parameter. Setting $\epsilon = (\mu_r - \gamma)n$, we have $\Pr[N_? \leq \gamma n] \leq \rho_{\gamma,r}$, where

$$\rho_{\gamma,r} = \left(1 + 2^{m+\theta_m} p_r\right) \exp\left(-2n\left(\frac{p_r}{p_r + 2^{-(m+\theta_m)}} - \gamma\right)^2 \cdot \frac{p_r + 2^{-(m+\theta_m)}}{2 - p_r - 2^{-(m+\theta_m)}}\right)$$

Converting ?'s to 1's. Now, note that we convert the ?'s to 1's independently with probability $\frac{1}{2}$. Consider the experiment of tossing T independent coins and counting the number of heads ct. This process can be modeled as a Markov chain as described in Figure 4. It is easy to check that this chain is irreducible. One can easily estimate the stationary distribution $\vec{\pi}$ and the second largest eigenvalue λ as

$$\vec{\pi} = \left(\frac{1}{2}, \frac{1}{2}\right)$$

$$\lambda = 0$$

We can also then check the detailed balance conditions to ascertain that the chain is also reversible. To bound the number of heads ct obtained while tossing T independent coins, we bound the time spent in the 1 state of the chain in Figure 4 using the expander Hoeffding bound for the uniform starting distribution, which is its stationary distribution. Note that the reversible chain is also finite and irreducible. Therefore, the expander Hoeffding bound is applicable. If we consider f such that $f(0) = 0$ and $f(?) = 1$, S_T counts the number of time steps the random walk spends in the 1 state. In this case, $\mu_r = \vec{\pi}_{r,?}$ and $\tilde{\lambda}_2 = \lambda_r$. We consider a random walk of length T that starts at a state x_0 sampled according to the uniform distribution, which is the stationary distribution of the chain. From the expander Hoeffding bound, we have that for any $\epsilon > 0$,

$$\Pr\left[S_T < \frac{T}{2} - \epsilon\right] \leq \exp\left(-2\frac{\epsilon^2}{T}\right)$$

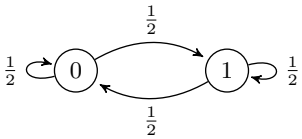

Fig. 4. The 2-state reversible chain that models coin flips.

Let $\beta \in [0, \frac{1}{2})_{\mathbb{R}}$ be a parameter. Let N_1 denote the number of time steps spent in the 1 state. Setting $\epsilon = (\frac{1}{2} - \beta)T$, we have

$$\Pr[N_1 \leq \beta T] \leq \exp\left(-2T\left(\frac{1}{2} - \beta\right)^2\right) = \chi_{\beta,T}$$

Putting it all together. Thus far, we have estimated separately the probabilities that our random walk on the reversible chain spends too little time in the ? state, and that too few ? are converted into 1's. In theory, we could just add up these failure probabilities and wrap things up. However, this plays against us, particularly for concrete parameters. This is because in order to make the first failure probability, $\rho_{\gamma,r}$ small, we would have to keep γ small. Indeed, the lower γ is, the better our concentration bound works and the lower $\rho_{\gamma,r}$ will be. But, this is antithetical to what we would like in the case of trying to make the second probability, namely, the probability of converting too few ?'s to 1's, small. Indeed, $\Pr[N_1 \leq \beta T]$ will be smaller for larger T. This push and pull in the two failure probabilities makes it difficult to achieve good concrete parameters. To get around this issue, we note that simply adding the probabilities is in a sense performing a worst-case of worst-case analysis. Rather, we must perform a case-by-case analysis. For instance, for large γ, $\rho_{\gamma,r}$ will not be very small, but $\chi_{\beta,\gamma n}$ will be tiny. Similarly, on the other end, if γ is small, $\chi_{\beta,\gamma n}$ will not be very small, but $\rho_{\gamma,r}$ will be tiny. The intuition of the upcoming analysis is that we must leverage this tradeoff in order to estimate $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right]$ tightly.

We divide $[0,1]$ into K bins of size $\frac{1}{K}$. We then use the law of total probability over the space of the K bins as follows. We have

$$\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] \leq \sum_{i\in[K]} \Pr\left[N_? \in \left[\frac{(i-1)n}{K}, \frac{in}{K}\right]\right] \cdot \Pr\left[N_1 \leq \left(\frac{\delta K}{i-1}\right)\frac{(i-1)n}{K}\right]$$

where $\Pr\left[N_1 \leq \left(\frac{\delta K}{i-1}\right)\frac{(i-1)n}{K}\right] \leq \chi_{\frac{\delta K}{i-1}, \frac{(i-1)n}{K}}$.

Now, we need to estimate $\Pr\left[N_? \in \left[\frac{(i-1)n}{K}, \frac{in}{K}\right]\right]$. A trivial bound would be that

$$\Pr\left[N_? \in \left[\frac{(i-1)n}{K}, \frac{in}{K}\right]\right] \leq \Pr\left[N_? \leq \frac{in}{K}\right] = \rho_{\frac{i}{K},r}$$

This would give us

$$\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] \leq \sum_{i\in[K]} \rho_{\frac{i}{K},r}\chi_{\frac{\delta K}{i-1}, \frac{(i-1)n}{K}} = \epsilon_{\delta,r}$$

which would work, but we would like to do even better.

Due to lack of space, we defer this analysis to Appendix B.4.

Analytical bounds. In order to estimate $\sum_{i\in[n]} Y_i$ analytically, we consider a different version of the reversible chain, one that implicitly converts ?'s to 0's and 1's independently with probability $\frac{1}{2}$. We present this chain in Figure 5. It is easy to see that this 3-state chain is identical to the 2-state chain in Figure 2 where we convert ?'s to 0's and 1's independently with probability $\frac{1}{2}$. Indeed, the 0 states are identical, and we have split up the ? state into two states, namely, $0_?$ and $1_?$ that share the probability masses of the ? state. Indeed, now $\sum_{i\in[n]} Y_i$ is equal to the time spent in the $1_?$ in this new 3-state chain, and we will estimate that using the expander Hoeffding bound.

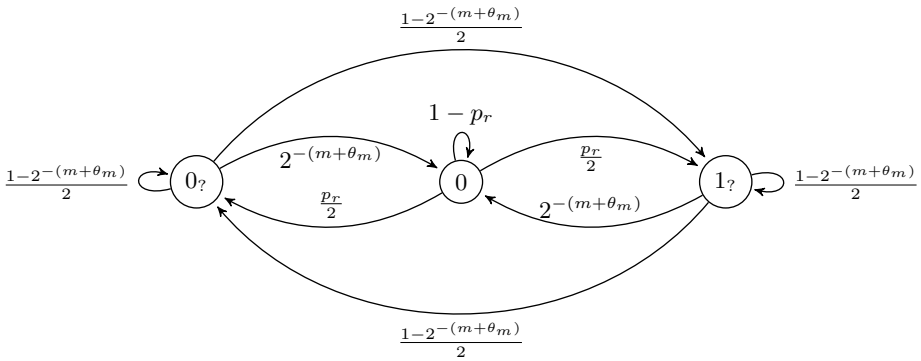

Fig. 5. The 3-state reversible chain which we compare to.

It is easy to check that this 3-state chain is irreducible. One can easily estimate the stationary distribution $\vec{\pi}_r$ and the second largest eigenvalue λ_r as

$$\vec{\pi}_r = \left(\frac{2^{-(m+\theta_m)}}{p_r + 2^{-(m+\theta_m)}}, \frac{1}{2}\cdot\frac{p_r}{p_r + 2^{-(m+\theta_m)}}, \frac{1}{2}\cdot\frac{p_r}{p_r + 2^{-(m+\theta_m)}}\right)$$

$$\lambda_r = 1 - p_r - 2^{-(m+\theta_m)}$$

We can also then check the detailed balance conditions to ascertain that the chain is also reversible. To bound the time spent in the $1_?$ state, we use the corollary of the expander Hoeffding bound for non-uniform starting distributions. Note that the reversible chain is also finite and irreducible. Therefore, the expander Hoeffding bound is applicable. If we consider f such that $f(0) = f(0_?) = 0$ and $f(1_?) = 1$, S_n counts the number of time steps the random walk spends in the $1_?$ state. In this case, $\mu_r = \vec{\pi}_{r,1_?}$ and $\tilde{\lambda}_2 = \lambda_r$. We consider a random walk of length n that starts at the $x_0 = 0$ state. From the corollary of the expander Hoeffding bound, we have that for any $\epsilon > 0$,

$$\Pr[S_n < n\mu_r - \epsilon] \leq \left(1 + 2^{m+\theta_m} p_r\right) \exp\left(-2\frac{\epsilon^2}{n}\cdot\frac{1-\lambda_r}{1+\lambda_r}\right)$$

Setting $\epsilon = (\mu_r - \delta)n$, we have $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] \leq \epsilon_{\delta,r}$, where

$$\epsilon_{\delta,r} = \left(1 + 2^{m+\theta_m} p_r\right) \exp\left(-2n\left(\frac{1}{2} \cdot \frac{p_r}{p_r + 2^{-(m+\theta_m)}} - \delta\right)^2 \cdot \frac{p_r + 2^{-(m+\theta_m)}}{2 - p_r - 2^{-(m+\theta_m)}}\right)$$

5.2.7 Completing the Security Analysis In the previous section, we determined $\epsilon_{\delta,r}$ such that $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] \leq \epsilon_{\delta,r}$. Notice that all the analysis up until now has been for a fixed hamming weight $r \in [k]$ (which we occasionally suppress in notion for readability). To complete the analysis, we must perform a union bound over all messages of all possible hamming weights. This way, we bound the probability that any message is mapped to a codeword of low hamming weight. This is because $\sum_{i\in[n]} X_i$ is exactly $\mathsf{HW}(\vec{y}) = \mathsf{HW}(\vec{x}G)$, and through Theorem 2, we have upper bounded $\Pr\left[\sum_{i\in[n]} X_i \leq \delta n - \hat{k}(m-1)\right]$ with $\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right]$ (within a multiplicative factor). Since the code is linear, we are effectively bounding the probability that the code has low minimum distance.

Let $\mathsf{d}(G)$ denote the minimum distance of the code G where $G \leftarrow \mathsf{ECGen}$. Using Theorem 2, we have

$$\Pr[\mathsf{d}(G) < \delta n - \hat{k}(m-1)] \leq \sum_{r\in[k]} \binom{k}{r} \frac{1}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \epsilon_{\delta,r}$$

where $\tilde{\delta}_r = \frac{\hat{k}}{n \cdot 2^{-(m+\theta_m)} \cdot p_r}$. Using our analytically friendly $\epsilon_{\delta,r}$ from before, we prove the following theorem.

Theorem 3. *Let $w, m, k, n \in \mathbb{N}$ with $w, m, k \leq n$. Define $R = \frac{k}{n}$. Fix $\delta \in [0,1]$ and $\hat{k} > 0$. We assume that the following hold: $w = C \ln n$ for some $C > 2$; $m = C_m \log n$ for some $C_m > 1$; $R \leq \frac{2}{e}$, $C\left(\frac{20}{41} - \delta\right)^2 > 2$ and $R < \frac{1}{\ln 2} \cdot \frac{e-1}{e+1}\left(\frac{20}{41} - \delta\right)^2$; $\hat{k} \geq n^{1-C_m}$ and $\hat{k} \geq 2\ln 2$.*

Then, for all sufficiently large n,

$$\Pr\left[\mathsf{d}(G) < \delta n - \hat{k}(m-1) : G \leftarrow \mathsf{ECGen}\left(w, m, k, n, \frac{1}{2}, \mathbb{F}_2\right)\right] \leq 2Rn^{-C\left(\frac{20}{41}-\delta\right)^2 + C_m + 3}$$

Proof. See Appendix B.5. □

For computing our parameters, we use our tighter analysis from the previous section and empirically estimate the failure probabilities which appear in Table 1. It is evident that we compare rather favorably with [10] boasting much smaller failure probabilites for their choice as parameters, as well as for the enhanced choice of rate $R = \frac{1}{2}$. In the following sections we give additional ways of further improving our results.

	δ	[10] ($R=\frac{1}{5}$)	Ours ($R=\frac{1}{5}$)	Ours ($R=\frac{1}{2}$)
$C=3$	0.005	$2.77\cdot10^{-4}$	$9.39\cdot10^{-12}$	$2.56\cdot10^{-10}$
	0.02	$1.08\cdot10^{-3}$	$3.31\cdot10^{-10}$	$7.32\cdot10^{-9}$
	0.05	$1.44\cdot10^{-2}$	$5.25\cdot10^{-8}$	$8.61\cdot10^{-7}$
$C=2.5$	0.005	$1.23\cdot10^{-2}$	$1.32\cdot10^{-8}$	$2.32\cdot10^{-7}$
	0.02	$3.84\cdot10^{-2}$	$2.57\cdot10^{-7}$	$3.79\cdot10^{-6}$
	0.05	$3.58\cdot10^{-1}$	$1.75\cdot10^{-5}$	$2.01\cdot10^{-4}$
$C=2.3$	0.005	$5.67\cdot10^{-2}$	$2.37\cdot10^{-7}$	$3.49\cdot10^{-6}$
	0.02	$1.65\cdot10^{-1}$	$3.65\cdot10^{-6}$	$4.57\cdot10^{-5}$
	0.05	1	$1.77\cdot10^{-4}$	$1.76\cdot10^{-3}$
$C=1.5$	0.005	1	$2.26\cdot10^{-2}$	$1.648\cdot10^{-1}$
	0.02	1	$1.34\cdot10^{-1}$	$8.81\cdot10^{-1}$
	0.05	1	1	1

Table 1. Comparing failure probabilities for $k=2^{20}$ between our non-wrapping construction with $m=25$ and [10]. For both constructions we have expected row weight $w=C\ln n$ and we assume we start at state 0 (see Section 5.2.8). For [10] we use δ and for ours $\tilde{\delta}$ such that $\tilde{\delta}n-\hat{k}(m-1)=\delta n$.

	δ	[10] ($R=\frac{1}{5}$)	Ours ($R=\frac{1}{5}$)	Ours ($R=\frac{1}{2}$)	Ours ($R=\frac{4}{7}$)
$C=3$	0.005	$1.39\cdot10^{-4}$	$6.28\cdot10^{-14}$	$7.30\cdot10^{-13}$	$1.06\cdot10^{-12}$
	0.02	$5.39\cdot10^{-4}$	$2.22\cdot10^{-12}$	$2.09\cdot10^{-11}$	$1.29\cdot10^{-11}$
	0.05	$7.21\cdot10^{-3}$	$3.51\cdot10^{-10}$	$2.46\cdot10^{-9}$	$1.53\cdot10^{-9}$
$C=2.5$	0.005	$6.16\cdot10^{-3}$	$1.06\cdot10^{-10}$	$7.93\cdot10^{-10}$	$1.08\cdot10^{-9}$
	0.02	$1.92\cdot10^{-2}$	$2.06\cdot10^{-9}$	$1.30\cdot10^{-8}$	$8.68\cdot10^{-9}$
	0.05	$1.79\cdot10^{-1}$	$1.40\cdot10^{-7}$	$6.89\cdot10^{-7}$	$4.64\cdot10^{-7}$
$C=2.3$	0.005	$2.84\cdot10^{-2}$	$2.07\cdot10^{-9}$	$1.30\cdot10^{-8}$	$1.72\cdot10^{-8}$
	0.02	$8.25\cdot10^{-2}$	$3.17\cdot10^{-8}$	$1.70\cdot10^{-7}$	$1.17\cdot10^{-7}$
	0.05	$7.81\cdot10^{-1}$	$1.54\cdot10^{-6}$	$6.56\cdot10^{-6}$	$4.56\cdot10^{-6}$
$C=1.5$	0.005	1	$3.00\cdot10^{-4}$	$9.35\cdot10^{-4}$	$1.11\cdot10^{-3}$
	0.02	1	$1.78\cdot10^{-3}$	$5.01\cdot10^{-3}$	$3.91\cdot10^{-3}$
	0.05	1	$2.26\cdot10^{-2}$	$5.54\cdot10^{-2}$	$4.32\cdot10^{-2}$

Table 2. Comparing failure probabilities for $k=2^{20}$ between our non-wrapping construction with $m=25$ and [10]. For both constructions we have expected row weight $w=C\ln n$ and we assume we start in the stationary (see Section 5.2.8). For [10] we use δ and for ours $\tilde{\delta}$ such that $\tilde{\delta}n-\hat{k}(m-1)=\delta n$.

5.2.8 Initializing the chain with the stationary distribution The corollary of the Hoeffding bound that we apply introduces a noticable significant multiplicative factor in the failure probability when the starting distribution is not the stationary distribution. When defining $G = BC$ where B is a sparse expander, the initial state of the accumulator is the zero state, which can be far from the stationary distribution due to having small mass on the zero state. One can include this discrepancy in the failure bound (Table 1). Alternatively, one can directly sample the initial state from the stationary distribution to improve the failure bounds (Table 2). Consider the related code $G' = B'C'$ where $B' = R||B$ and $R \in \{0,1\}^{k \times m}$ is the matrix that output the accumulator state after a sufficient number of iterations. In particular, let us assume we have an upper bound q on the number of steps it takes for a random walk on the Markov chain to converge from the zero state to the stationary distribution. We can then sample an expander $\hat{B} \in \{0,1\}^{k \times q}$ and a convolution $\hat{C} \in \{0,1\}^{q \times q}$ and define $R = \hat{B} \cdot \hat{C} \cdot (0^{q-m \times m} /\!\!/ I_m)$ That is, consider the process of performing q iterations of expand accumulate and outputting the state. For the convolution we use, the state is simply the last m outputs which can be obtained by multiplying the whole output by $(0^{q-m \times m} /\!\!/ I_m)$, where I_m is the identity matrix of size m. For irreducible, reversible Markov chains, it is known that the mixing time $\tau(\epsilon) \leq \frac{1}{1-\lambda_2} \ln\left(\frac{1}{\pi_* \epsilon}\right)$ for $\epsilon > 0$, where λ_2 is the second largest eigenvalue of the chain. The mixing time estimates the number of steps needed to get ϵ-close to the stationary distribution. The chains that we have analyzed using the Hoeffding bound are all irreducible, reversible, and have $\lambda_2 = 1 - p_r - 2^{-(m+\theta_m)}$ and $\pi_* = \frac{1}{1+2^m p_r}$. To obtain a failure probability of $2^{-\lambda}$, it suffices to perform $q = \frac{m+\lambda}{p_r + 2^{-(m+\theta_m)}}$ additional iterations. Alternatively, we conjecture that sampling R as uniform should produce a similar distribution. These results appear in Table 2[6], where we can now achieve $C = 1.5$ and $R = \frac{4}{7} > \frac{1}{2}$.

5.3 Optimizations

We additionally present two optimizations to our code that are proven using natural extensions of our analysis. The first is referred to as wrapping where the last bit of the convolution is always set to one. This makes the convolution less likely to transition to the zero state. Secondly we show that our construction can naturally be made systematic with essentially no loss in minimum distance. This in turn improves our running by 1.5 times. See Appendix A.

6 Implementation

We implement expand accumulate codes [10] (henceforth EA) and our new expand convolute codes and report on their running time performance. The code can be found in the libOTe repository. Running times were obtained on an Intel

[6] The $R = \frac{4}{7}$ case was analyzed with better precision, which explains why the failure probability is sometimes smaller than for smaller rate.

$i7$ laptop with 8GB of RAM. It is our intention to open-source the code after publication. Our conclusion is that our code is twice as fast as EA for comparable security. We choose to sample the expander B with fixed row weight w. We define the rate R of the code as $R = k/n$ and consider R for our code. The unique parameter for our code is the convolution state size m. We only implement the wrapping code where the last convolution state transition bit is always 1. We report our running times of the dual LPN encoding algorithms (i.e. $\overline{y} = G\overline{e}$) in Table 3 for $k = 2^{20}$ outputs. The running time of the associated PCG algorithms for generating OT and VOLE correlations [14,11] scales proportionally to the encoding times due to encoding being the primary overhead. For EA to provably have an acceptable probability of being linear minimum distance (see Table 1) they require an approximate expander weight of $w = 40$. In contrast, our code for the same distance δ, expander weight w, and state size $m = 16$ achieves a 10^{10} times smaller failure probability. Moreover, our code requires 548ms while EA requires 740ms. For a comparable failure probability, we can decrease w to $w = 21$ and obtain a resulting running time of 275ms, a reduction of 2.7×.

EA also proposes aggressive parameters where they set the expander weight w to be in the range of 7 to 11. For these parameters, they empirically estimate the (pseudo) minimums distance and conjecture that it should be hard to find smaller weight codewords. We showed that this is not the case (see 4.2) by finding moderately smaller codewords. However, this appears to only mildly decrease their security guarantees. For example, EA with $w = 7$, we were able to find a minimum distance $n\delta$ for δ as small as $\delta = 0.002$ compared to their conjectured $\delta = 0.06$. When applying the same techniques to our most aggressive parameters of $w = 5, m = 9$, we were able to find a distance as small as $\delta = 0.03$. However, with little runtime overhead, we suggest increasing the state size to $m = 25$ which increases the empirical minimum distance found to $\delta = 0.1$. Moreover, the running time of this code is 74ms compared to the empirically worse minimum distance code of $w = 7$ EA with 143ms. In light of these results and Table 2 & 4, we make the following suggestions for our *wrapping* convolution with $k < 2^{26}$: aggressive parameters $w = 5, m = 25$; moderate parameters $w = 21, m = 21$; conservative parameters $w = 41, m = 25$. Note that for $k = 2^{20}$, the effective non-wrapping state size m is approximately 15 larger. Finally, we also consider the systematic version of our code (Section A.2) which gives a further 1.5× speedup with almost no change to minimum distance.

References

1. Aguilar-Melchor, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. IEEE TIT (2018)
2. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in AC^0 o MOD_2. pp. 251–260 (2014). https://doi.org/10.1145/2554797.2554821
3. Al Jabri, A.: A statistical decoding algorithm for general linear block codes (2001)
4. Alekhnovich, M.: More on average case vs approximation complexity. pp. 298–307 (2003). https://doi.org/10.1109/SFCS.2003.1238204

Code	[11]	[10]	Ours non-sys			Ours sys		
m	-	1	9	17	25	9	17	25
R \ w	1/2	1/5	1/2			1/2		
5		–	62	67	74	49	52	54
7		143	82	90	95	62	64	68
11	2131	192	140	145	153	99	101	106
21		357	265	275*	283	175	190	203
41		740*	505	534	548	334	340	346

Table 3. The running time of our code in milliseconds compared to related art for $k = 2^{20}$. The codes have rate $R = k/n$, wrapping state size m, and expander weight w (not applicable to [11]). We give the running times for both non-systematic (non-sys) and systematic (sys) version of our codes.

5. Aragon, N., Barreto, P.L., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.P., Zemor, G.: Bike: Bit flipping key encapsulation (2017)
6. Bazzi, L., Mahdian, M., Spielman, D.A.: The minimum distance of turbo-like codes. IEEE Trans. Inf. Theory **55**(1), 6–15 (2009)
7. Berrou, C., Glavieux, A.: Near optimum error correcting coding and decoding: turbo-codes. IEEE Trans. Commun. **44**(10), 1261–1271 (1996)
8. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. pp. 278–291 (1994). https://doi.org/10.1007/3-540-48329-2_24
9. Bogdanov, A., Rosen, A.: Pseudorandom functions: Three decades later. Cryptology ePrint Archive, Report 2017/652 (2017), https://eprint.iacr.org/2017/652
10. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated pseudorandomness from expand-accumulate codes. In: CRYPTO 2022
11. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation (2019)
12. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. Crypto (2019)
13. Carlos, Aguilar, M., Isae, S., Aragon, N., Loic, Olivier, B., Jurjen, B., Jean-Christophe, D., Philippe, G., Jerome, Edoardo, P., Jean-Marc, R., Pascal, V., Gilles, Z.: Hamming quasi-cyclic (hqc) (2020)
14. Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: CRYPTO (2021)
15. Debris-Alazard, T., Tillich, J.P.: Statistical decoding (2017)
16. Fossorier, M.P., Kobara, K., Imai, H.: Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem (2006)
17. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem (2005)
18. McEliece, R.J.: A public-key cryptosystem based on algebraic (1978)
19. Prange, E.: The use of information sets in decoding cyclic codes (1962)
20. Stern, J.: A method for finding codewords of small weight (1988)
21. Tillich, J., Zémor, G.: On the minimum distance of structured LDPC codes with two variable nodes of degree 2 per parity-check equation (2006)

22. Vardy, A.: The intractability of computing the minimum distance of a code (1997)
23. Zichron, L.: Locally computable arithmetic pseudorandom generators (2017)

SUPPLEMENTARY MATERIAL

Disclaimer Case studies, comparisons, statistics, research and recommendations are provided "AS IS" and intended for informational purposes only and should not be relied upon for operational, marketing, legal, technical, tax, financial or other advice. Visa Inc. neither makes any warranty or representation as to the completeness or accuracy of the information within this document, nor assumes any liability or responsibility that may result from reliance on such information. The Information contained herein is not intended as investment or legal advice, and readers are encouraged to seek the advice of a competent professional where such advice is required.

These materials and best practice recommendations are provided for informational purposes only and should not be relied upon for marketing, legal, regulatory or other advice. Recommended marketing materials should be independently evaluated in light of your specific business needs and any applicable laws and regulations. Visa is not responsible for your use of the marketing materials, best practice recommendations, or other information, including errors of any kind, contained in this document.

A Optimizations

A.1 Wrapping Convolutions

In Section E we present a technique that we refer to as wrapping which allows the convolutions to behave as if it had a larger state size. Recall that at each iteration, the convolution outputs the next bit as $y_i = \vec{x}B_{\cdot,i} + \vec{\sigma}_i\vec{\alpha}_i$. The next state σ_{i+1} is then defined as $(y_i||\sigma_{i,1}, ..., \sigma_{i,m-1})$. Essentially, for the code to have low minimum distance it is necessary for the state to spend a lot of time as 0^m. With high probability this implies that the state must transition from a non-zero state to the zero state and as such at some step i, state $\vec{\sigma}_i = (0, ..., 0, 1)$ transitions to $\vec{\sigma}_{i+1} = 0^m$. For non-wrapping chains this final transition happens with probability $1/2$.

Consider defining α_i to always having the last bit defined as 1. As such, for the critical transition from $\vec{\sigma}_i = (0, ..., 0, 1)$ to $\vec{\sigma}_{i+1} = 0^m$, it is necessary for $\vec{x}B_{\cdot,i} = 1$. However, this only happens with probability p_r, which for small r is much less than $1/2$. As r increases we converge back to the previous case of $1/2$ transition probability. When analyzing a wrapping chain of state size $\tilde{m}$, we thus map for each $r \in [1, k]$ the corresponding non-wrapping chain of state size $\tilde{m} + \lfloor -\log p_r \rfloor - 1$.

However, a close analysis of the proof shows that the primary contributors to low minimum distance are low weight messages. As such, by defining the last bit of α_i to be 1, we are able to achieve a more robust accumulator. Or equivalently, we are able to have a smaller state size which leads to more efficient implementation. Table 4 shows the failure probabilities for a state size $m = 5$ wrapping chain assuming the conjecture of Lemma 4 holds.

Empirical analysis, supported by Tables 5,6,7 in Appendix E, seems to show that, in our parameter range, a wrapping accumulator with m sates gives similar failure probabilities than a non-wrapping accumulator with approximately $m - \log_2 p_1$ states. For example, a non-wrapping chain of size $m = 25$ is equivalent to a wrapping chain with $m = 9$. We note that beyond a certain threshold, increasing the state size of the wrapping chain may be detrimental if we do not assume the chain starts with stationary distribution (see Sec.5.2.8). However, we believe this to be an artifact of the proof.

systematic?		no	no	yes	yes
	δ \ R	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{5}$	$\frac{1}{2}$
$C = 3$	0.005	$4.31 \cdot 10^{-11}$	$2.99 \cdot 10^{-10}$	$9.63 \cdot 10^{-11}$	$2.5 \cdot 10^{-9}$
	0.02	$1.50 \cdot 10^{-9}$	$8.46 \cdot 10^{-9}$	$9.63 \cdot 10^{-11}$	$2.5 \cdot 10^{-9}$
	0.05	$2.26 \cdot 10^{-7}$	$9.54 \cdot 10^{-7}$	$2.29 \cdot 10^{-7}$	$3.0 \cdot 10^{-6}$
$C = 2.5$	0.005	$1.34 \cdot 10^{-8}$	$2.59 \cdot 10^{-7}$	$4.64 \cdot 10^{-8}$	$7.0 \cdot 10^{-7}$
	0.02	$2.59 \cdot 10^{-7}$	$4.19 \cdot 10^{-6}$	$4.64 \cdot 10^{-8}$	$7.0 \cdot 10^{-7}$
	0.05	$1.72 \cdot 10^{-5}$	$2.13 \cdot 10^{-4}$	$3.01 \cdot 10^{-5}$	$2.5 \cdot 10^{-4}$
$C = 2.3$	0.005	$2.38 \cdot 10^{-7}$	$3.82 \cdot 10^{-6}$	$5.45 \cdot 10^{-7}$	$6.6 \cdot 10^{6}$
	0.02	$3.64 \cdot 10^{-6}$	$4.94 \cdot 10^{-5}$	$5.45 \cdot 10^{-7}$	$6.6 \cdot 10^{6}$
	0.05	$1.73 \cdot 10^{-4}$	$1.83 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$	$1.5 \cdot 10^{-3}$

Table 4. Failure probabilities for $k = 2^{20}$ for our systematic and non-systematic constructions with wrapping and $m = 5$ and overall rate R. We have expected row weight $w = C \ln n$ and we assume we start at state 0 (see Section 5.2.8). We use $\tilde{\delta}$ such that $\tilde{\delta} n - \hat{k}(m-1) = \delta n$.

A.2 Systematic

Our code can also be instantiated in the systematic setting where the codeword for $x \in \mathcal{R}^k$ is instantiated as $c = x|xG$. In this setting the rate of G can be greater than 1 due to the codeword additionally including the message. In particular, we consider the code where $G = BC$, B is an $k \times k$ expander and convolution C. G has rate 1 which results in an overall rate 0.5. The core observation for trivially extending our minimum distance analysis to this setting is that the union bound need not be performed for all $r = 1, \ldots, k$, but instead we only $r = 1, \ldots, t$ where $t = \delta k$ is the target weight of the code, and $\delta < 1$, e.g. $\delta = 0.05$. Moreover, when performing the union bound for r, the codeword already has at least r ones in it and therefore r fewer ones must be produced by xG. As we will see in the next section, our systematic codes are approximately 1.5× faster in terms of running time. Moverover, the minimum distance behavior is essentially the same between systematic and non-systematic codes as can be seen in Table 4.

B Deferred Proofs and Analyses

B.1 Irreversible Run of 1's: Proof of Lemma 4

Proof. We first want to prove that for all $i \leq n$, $p_i \leq \frac{1}{2} \cdot 2^{-m}$. We know that for all i, a random walk that visits state 0_m for the first time at step $i+m$ has to transition $1 \to 0_1 \to \ldots \to 0_m$, which happens with probability 2^{-m}. What remains to be computed is the probability that a random walk of i steps starting at state 1 ends at state 1 without ever visiting 0_m. This happens with at most probability $\frac{1}{2}$. We call the walks that end at state 1 without ever visiting 0_m as valid paths, and the other paths as invalid paths. Note that all valid paths are equally likely to be drawn, in particular, with a probability of 2^{-i}. Half of all possible walks starting at state 1 end at state 1 in walks of length less than m, but once we start considering walks of length more than m steps, some of these walks are considered invalid since they visit 0_m. Because all valid paths are equally likely to be drawn, we have that with probability at most $\frac{1}{2}$, the walk is valid (has never visited 0_m) and ends at state 1. We thus have $p_i \leq \frac{1}{2} \cdot 2^{-m}$.

Now, note that for all $t \geq 2$, we have $\left(1 - \frac{1}{t}\right)^t > \frac{1}{2e}$. Furthermore, for all $n \in \mathbb{N}$ and all $m \geq \log n + 2$, we have

$$\left(1 - \frac{1}{2^m}\right)^n > (2e)^{-\frac{n}{2^m}} \geq (2e)^{-0.25} > \frac{1}{2}$$

This means that for all $n \in \mathbb{N}$, $m \geq \log n + 2$ and $i \in [n]$, $\left(1 - \frac{1}{2^m}\right)^i > \frac{1}{2}$. We can thus conclude by observing that for $m \geq \log n + 2$, $\forall i \in [n]$, $q_i = (1 - 2^{-m})^i \cdot 2^{-m} \geq \frac{1}{2} \cdot 2^{-m}$.

We would now like to talk about all $m \geq 2$ as opposed to just $m \geq \log n + 2$. Let $N_{i,m}$ denote the number of valid walks of length i. Then,

$$p_i = N_{i,m} \cdot 2^{-i} \cdot 2^{-m}$$

We will show that $N_{i,m} = F_i^{(m)}$, the ith term in the Fibonacci sequence of order m. Recall that the Fibonacci sequence of order m, $F^{(m)}$, is defined as

$$F_i^{(m)} = \begin{cases} 0 & i < 0 \\ 1 & i = 0 \\ \sum_{j=i-1}^{i-m} F_j^{(m)} & i > 0 \end{cases}$$

To help understand why $N_{i,m} = F_i^{(m)}$, we use the help of an example. Figure 6 is an illustration that shows the tree of all possible walks of length $i = 5$ on the chain for the case of $m = 4$. Note that the invalid paths are marked in red. There are 15 valid paths, that can be partitioned into the following groups:

- 8 valid paths of length 4 which then take the $1 \to 1$ transition.
- 4 valid paths of length 3 which then takes the $1 \to 0_1 \to 1$ transitions.

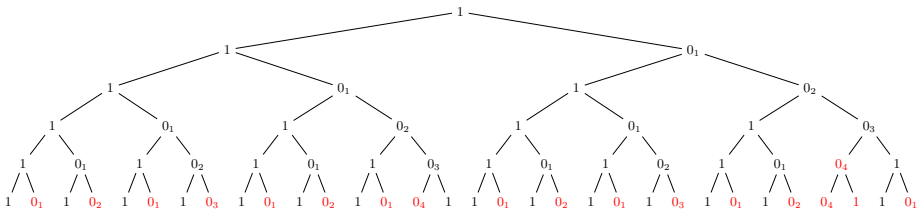

Fig. 6. All possible paths of 5 steps from state 1 for $m = 4$

– 2 valid paths of length 2 which then takes the $1 \to 0_1 \to 1 \to 1$ transitions.
– 1 valid path of length 1 which then takes the $1 \to 0_1 \to 1 \to 1 \to 1$ transitions.

Thus, the number of valid paths of length 5 for $m = 4$ satisfies

$$N_{5,4} = N_{4,4} + N_{3,4} + N_{2,4} + N_{1,4}$$

One can easily verify the base cases of this recursion as $N_{1,4} = 1$, $N_{1,4} = 1 + N_{1,4} = 2$, $N_{3,4} = 1 + N_{1,4} + N_{2,4} = 4$ and $N_{4,4} = 1 + N_{1,4} + N_{2,4} + N_{3,4} = 8$ from Figure 6. Given this, it is clear that $N_{5,4} = F_5^{(4)}$. The above argument readily generalizes to show that $N_{i,m} = F_i^{(m)}$. It is known that

$$F_i^{(m)} = \left\lfloor \frac{r_m^i(r_m - 1)}{(m+1)r_m - 2m} \right\rceil = \left\lfloor \frac{r_m^i(r_m - 1)}{(m+1)r_m - 2m} + \frac{1}{2} \right\rfloor \leq \frac{r_m^i(r_m - 1)}{(m+1)r_m - 2m} + \frac{1}{2}$$

where r_m is the real root of $x + x^{-m} - 2$ that is closest to 2.

Claim. *For all* $m \geq 2$, $2\left(1 - \frac{1}{2^m}\right) < r_m < 2\left(1 - \frac{1}{2^{m+1}}\right)$.

Proof. Using elementary calculus, one can easily verify that the real function $f(x) = x + x^{-m}$ has a local minimum at $x = m^{\frac{1}{m+1}} \in \left(1, \frac{3}{2}\right)$, for all $m \geq 2$. Furthermore, for $x > m^{\frac{1}{m+1}}$, $f(x)$ is increasing. Note that $f(2) > 2$, and it can also be easily verified that $f\left(m^{\frac{1}{m+1}}\right) < 2$ for all $m \geq 2$. Therefore, $r_m \in \left(m^{\frac{1}{m+1}}, 2\right)$. Note that $2\left(1 - \frac{1}{2^{m+1}}\right) \in \left(m^{\frac{1}{m+1}}, 2\right)$ for all $m \geq 2$. Now,

$$\begin{aligned} f\left(2\left(1 - \frac{1}{2^{m+1}}\right)\right) &= 2\left(1 - \frac{1}{2^{m+1}}\right) + \left(2\left(1 - \frac{1}{2^{m+1}}\right)\right)^{-m} \\ &= 2 + \left(2\left(1 - \frac{1}{2^{m+1}}\right)\right)^{-m} - 2^{-m} \\ &> 2 \end{aligned}$$

as $2\left(1-\frac{1}{2^{m+1}}\right) < 2$. Therefore, $r_m < 2\left(1-\frac{1}{2^{m+1}}\right)$. Similarly, note that $2\left(1-\frac{1}{2^m}\right) \in \left(m^{\frac{1}{m+1}}, 2\right)$ for all $m \geq 2$. Now,

$$\begin{aligned} f\left(2\left(1-\frac{1}{2^m}\right)\right) &= 2\left(1-\frac{1}{2^m}\right) + \left(2\left(1-\frac{1}{2^m}\right)\right)^{-m} \\ &= 2 + \left(2\left(1-\frac{1}{2^m}\right)\right)^{-m} - 2^{-(m-1)} \\ &< 2 \end{aligned}$$

as $\left(1-\frac{1}{2^m}\right)^m > \frac{1}{2}$ for all $m \geq 2$. Therefore, $r_m > 2\left(1-\frac{1}{2^m}\right)$. □

From our claim, $r_m = 2\left(1-2^{-(m+\psi_m)}\right)$ for some $\psi_m \in (0,1)$. It can be verified that the ratio $\frac{r_m-1}{(m+1)r_m-2m}$ decreases as m increases, rapidly approaching $\frac{1}{2}$ from above, as r_m approaches 2. Therefore, crudely, we expect $F_i^{(m)} \approx \frac{1}{2}r_m^i$ and therefore $p_i \approx 2^{-(m+1)}\left(1-2^{-(m+\psi_m)}\right)^i$. It then seems plausible to consider setting $\theta_m \approx \psi_m$ to have $p_i < q_i$ for all $i \geq 1$ as $2^{-(m+1)} < 2^{-(m+\psi_m)}$ for $\psi_m < 1$. However, this does turn out to be a crude approximation. For example, in the case of $m = 4$ and $m = 5$, we do have $p_4 > q_4$ and $p_5 > q_5$ for $\theta_4 = \psi_4$ and $\theta_5 = \psi_5$. While the margin of error in estimating r_4, r_5 and ψ_4, ψ_5 might be responsible for this empirical observation, it does question whether we can achieve $p_i < q_i$ for all $i \geq 1$.

However, even in cases such as the above, we do observe that for all $i \geq 1$, $\sum_{j\in[i]} p_j < \sum_{j\in[i]} q_j$ when $\theta_m = \psi_m$. From empirical observations, it seems like $i = m$ might be the only problematic value of i where p_i might perhaps be larger than q_i, but for all other values of i, it appears that $p_i < q_i$ and this more than compensates for the case of $i = m$ and satisfies $\sum_{j\in[i]} p_j < \sum_{j\in[i]} q_j$ when $\theta_m = \psi_m$. We provide evidence of this in Figure 7. We thus conjecture that for each $m \geq 2$, there exists $\theta_m = \psi_m \in (0,1)$ such that for all $i \geq 1$, $\sum_{j\in[i]} p_j < \sum_{j\in[i]} q_j$ where $r_m = 2\left(1-2^{-(m+\psi_m)}\right)$ is real root of $x + x^{-m} - 2$ closest to 2.

□

B.2 Random walk on irreversible Markov chain: Proof of Lemma 5

Proof. At a high level, a random walk on the irreversible chain can be described completely as: *"The walk left* 0 *after* i_1 *steps, then went back to* 0 *after* j_1 *steps, ...,"* where the number of steps i_k, j_k are drawn from the correct distribution. Moreover, the non-zero sub-walk is uniformly sampled conditioned on not getting m zeros in a row (except at the end) as is required. By construction, our process samples the number of steps for each of these *sub-walks* exactly as per the distribution induced by the transition probabilities in the irreversible chain.

In more detail, let us understand more closely the values η_i and ζ_i. Recall that $\tau_i = (1-p_r)^{i-1}p_r$. In the irreversible chain (Figure 1), this corresponds exactly to the probability of the event of starting at the state 0_m, staying there for the next

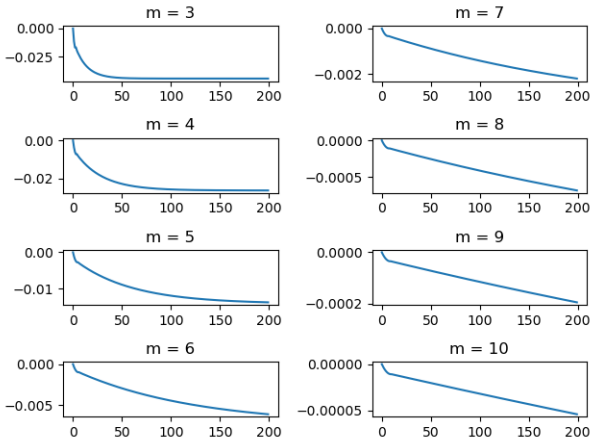

Fig. 7. Plots of the difference of partial sum $\sum_{j\in[i]}(p_j - q_j)$ for $1 \leq i \leq 200$ and $3 \leq m \leq 10$. The horizontal axes of the plots are i and the vertical axes are $\sum_{j\in[i]}(p_j - q_j)$.

$i-1$ steps, and exiting on *exactly* step i. The value η_i was defined as $\eta_i = \sum_{j=1}^{i-1} \tau_j$, and hence η_i is the probability that of the event of starting at the state 0_m and exiting *before* i steps. The claims regarding η_i being monotonically increasing and $\eta_i \to 1$ as $i \to \infty$ now follow intuitively as well. The monotonically increasing step function S_0 can now be seen as $S_0(p)$ is the unique k such that p is more than the probability of exiting before k steps but is at most the probability of exiting before $k+1$ steps. Thus, if we sample $p \leftarrow [0,1]$, $\underbrace{0^m \to 0^m \to \ldots \to 0^m \to 1}_{S_0(p) \text{ steps}}$ is a uniformly random sample of a random walk in the irreversible chain starting at 0_m and ending as soon as we reach 1. This is how we will use S_0 to sample sub-walks on the irreversible chain starting at 0_m and ending when we reach 1.

We can look at ζ_i in a similar manner. Recall that in the irreversible chain (Figure 1), p_i corresponds exactly to the probability of the event of starting at the state 1 and reaching the state 0_m for the first time on *exactly* step $i+m$. The value ζ_i was defined as $\zeta_i = \sum_{j=1}^{i-1} p_j$, and hence ζ_i is the probability that of the event of starting at the state 1 and reaching the state 0_m for the first time *before* $i+m$ steps. The claims regarding ζ_i being monotonically increasing and $\zeta_i \to 1$ as $i \to \infty$ now follow intuitively as well. The monotonically increasing step function S_1 can now be seen as $S_1(p) = k+m-1$ for the unique k such that p is more than the probability of reaching the state 0_m for the first time before $k+m$ steps but is at most the probability of reaching the state 0_m for

the first time before $k+m+1$ steps. Thus, if we sample $p \leftarrow [0,1]$,

$$\underbrace{\underbrace{\underbrace{1 \to \ldots \ldots \ldots}_{S_1(p)-m-1 \text{ steps}}}_{S_1(p)-m \text{ steps}} \to 1 \to 0_1 \to 0_2 \to \ldots 0_{m-1} \to 0_m}_{S_1(p) \text{ steps}}$$

is a uniformly random sample of a random walk in the irreversible chain starting at 1 and ending as soon as we reach 0_m. This is how we will use S_1 to sample sub-walks on the irreversible chain starting at 1 and ending when we reach 0_m. As shown above, the last m steps must be $1 \to 0_1 \to 0_2 \to \ldots \to 0_{m-1} \to 0_m$, and the one step before that, if it exists, must bring us back to 1. Thus, we still have to describe how to sample the sub-walk of $S_1(p) - m - 1$ steps, but that is just a simple Bernoulli distribution with parameter $\frac{1}{2}$, as transitions from each of the states $1, 0_1, \ldots, 0_{m-1}$ have probability $\frac{1}{2}$. Care must be taken, however, to ensure that we don't take the $0_{m-1} \to 0_m$ transition since that would end the sub-walk prematurely, but this is easy to do using bookkeeping.

Finally, we would like to show that $\tilde{X}$, the random variable described by Process 1, follows the same distribution as a random walk on the irreversible chain. $\tilde{X}$ is the random variable denoting the output of Algorithm 1 (see Appendix B.5) which samples a walk by alternating calls to S_0, S_1 and constructing sub-walks of that length. Specifically, Algorithm 1 samples a random walk $\vec{x}$ of length n by sampling sub-walks starting at 0_m and 1. Note that $\vec{x}$ is a binary vector and represents being in state 1 using a 1, and being in any of the states $0_1, \ldots, 0_m$ using a 0. S_0 is used to sample a sub-walk starting at state 0_m and ending at state 1, where the sub-walk spends some amount of time in state 0_m and then exits 0_m to end in 1. This is done exactly as we described earlier using $p = \mathsf{p}_i$ and appending $0^{S_0(\mathsf{p}_i)}$ to our random walk $\vec{x}$. S_1 is used to sample a sub-walk starting at state 1 and ending as soon as it reaches the state 0_m. This is done exactly as we described earlier using $p = \mathsf{p}_{i+1}$. $\vec{t}$ represents the part of the walk that precedes the last m steps $\to 0_1 \to 0_2 \to \ldots 0_{m-1} \to 0_m$. $\vec{t}$ begins with a 1 and then uses $\mathsf{Ber}_{\frac{1}{2}}$ transitions for $S_1(\mathsf{p}_{i+1}) - m - 1$ steps, using the randomness q.. A couple of corner cases need to be handled:

- If we've had $m-1$ 0s, i.e., we've been considering the sub-walk $0_1 \to \ldots \to 0_{m-1}$, we must choose the $0_{m-1} \to 1$ transition to avoid ending the walk prematurely, and hence append 1 to $\vec{t}$.
- If $S_1(\mathsf{p}_{i+1}) = m$, then the sub-walk must be $1 \to 0_1 \to \ldots \to 0_{m-1} \to 0_m$, and so we don't leave state 1 and return to it. In all other cases, we do, and therefore, $\vec{t}$ has an additional 1 appended to its end.

That completes the description of $\vec{t}$. The last m steps are accounted for by finally appending 0^{m-1}[7] to $\vec{x}$. and appending 0^{p_i} to our random walk $\vec{x}$. Thus, after these two sub-walks, we have appended $0^{S_0(\mathsf{p}_i)} \| \vec{t} \| 0^{m-1}$ to $\vec{x}$ and at this point,

[7] We append 0^{m-1} and note 0^m since the very last 0 for the state 0_m is already accounted for by $0^{S_0(\mathsf{p}_i)}$.

the walk has returned back to 0_m for the first time after leaving it. We do this repeatedly until we get a walk of length n (if $|\vec{x}|$ exceeds n, we simply trim it down to length n by considering its prefix, which maintains the distribution).

This completes the proof that $\tilde{X} = \vec{x}$ follows the same distribution as a random walk on the irreversible chain. □

B.3 Coupling: Proof of Theorem 2

Proof. (Theorem 2) For clarity, we abuse notation and always consider the Hamming weight of the string that the random variables output. We have

$$\begin{aligned}
\Pr\left[\sum_{i\in[n]} X_i \le \delta n - \hat{k}(m-1)\right] &= \Pr\left[\tilde{X} \le \delta n - \hat{k}(m-1)\right] \\
&= \Pr\left[Z_{\tilde{X}} \le \delta n - \hat{k}(m-1)\right] \\
&= \frac{\Pr\left[Z_{\tilde{X}} \le \delta n - \hat{k}(m-1) | Z_{\tilde{Y}} \le \delta n\right] \cdot \Pr\left[Z_{\tilde{Y}} \le \delta n\right]}{\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1)\right]}
\end{aligned}$$

where the last equality is from Bayes' rule. Due to the total probability law,

$$\begin{aligned}
&\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1)\right] \\
=&\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1), Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right] \cdot \Pr\left[Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right] \\
&+\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1), Z_{\tilde{X}} < Z_{\tilde{Y}} - \hat{k}(m-1)\right] \cdot \Pr\left[Z_{\tilde{X}} < Z_{\tilde{Y}} - \hat{k}(m-1)\right] \\
\ge&\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1), Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right] \cdot \Pr\left[Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right]
\end{aligned}$$

Note that

$$Z_{\tilde{X}} \le \delta n - \hat{k}(m-1) \wedge Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1) \implies Z_{\tilde{Y}} \le \delta n$$

Therefore,

$$\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1), Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right] = 1$$

From Lemma 7,

$$\Pr\left[Z_{\tilde{X}} \ge Z_{\tilde{Y}} - \hat{k}(m-1)\right] \ge 1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)$$

Therefore,

$$\Pr\left[Z_{\tilde{Y}} \le \delta n | Z_{\tilde{X}} \le \delta n - \hat{k}(m-1)\right] \ge 1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)$$

Substituting back, we have

$$\begin{aligned}\Pr\left[\sum_{i\in[n]} X_i \le \delta n - \hat{k}(m-1)\right] &\le \frac{\Pr\left[Z_{\tilde{X}} \le \delta n - \hat{k}(m-1) | Z_{\tilde{Y}} \le \delta n\right] \cdot \Pr\left[Z_{\tilde{Y}} \le \delta n\right]}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \\ &\le \frac{\Pr\left[Z_{\tilde{Y}} \le \delta n\right]}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \\ &= \frac{1}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right]\end{aligned}$$

□

B.4 Tight Analysis of the Reversible Chain

Notice that $\rho_{\frac{i}{K},r}$ increases as i increases from 1 to K, while $\chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}}$ decreases as i increases from 1 to K. Define

$$\upsilon_i = \Pr\left[N_? \in \left[\frac{(i-1)n}{K}, \frac{in}{K}\right]\right]$$

for $i \in [K]$. We have

$$\begin{aligned}\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] &\le \upsilon_i \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} \\ &= \sum_{i\in[K-1]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + \upsilon_K \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}}\end{aligned}$$

If $\upsilon_K \le \rho_{1,r} - \rho_{\frac{K-1}{K},r}$, then

$$\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] \le \sum_{i\in[K-1]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}}$$

If not, let $\psi_K = \upsilon_K - \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) > 0$. Then

$$\begin{aligned}
\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] &\le \sum_{i\in[K-1]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}} \\
&\qquad + \psi_K \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}} \\
&\le \sum_{i\in[K-1]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}} \\
&\qquad + \psi_K \cdot \chi_{\frac{\delta K}{K-2},\frac{(K-2)n}{K}} \\
&= \sum_{i\in[K-2]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + (\upsilon_{K-1} + \psi_K)\, \chi_{\frac{\delta K}{K-2},\frac{(K-2)n}{K}} \\
&\qquad + \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}}
\end{aligned}$$

If $\upsilon_{K-1} + \psi_K \le \rho_{\frac{K-1}{K},r} - \rho_{\frac{K-2}{K},r}$, then

$$\begin{aligned}
\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] &\le \sum_{i\in[K-2]} \upsilon_i \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} + \left(\rho_{\frac{K-1}{K},r} - \rho_{\frac{K-2}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-2},\frac{(K-2)n}{K}} \\
&\qquad + \left(\rho_{1,r} - \rho_{\frac{K-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{K-1},\frac{(K-1)n}{K}}
\end{aligned}$$

If not, we can define $\psi_{K-1} = \upsilon_{K-1} + \psi_K - \left(\rho_{\frac{K-1}{K},r} - \rho_{\frac{K-2}{K},r}\right) > 0$ and proceed as before. Proceeding like this until we cover every term in the summation, we see that

$$\begin{aligned}
\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] &\le \psi_1 + \sum_{i\in[K]} \left(\rho_{\frac{i}{K},r} - \rho_{\frac{i-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}} \\
&\approx \sum_{i\in[K]} \left(\rho_{\frac{i}{K},r} - \rho_{\frac{i-1}{K},r}\right) \cdot \chi_{\frac{\delta K}{i-1},\frac{(i-1)n}{K}}
\end{aligned}$$

as

$$\begin{aligned}
\psi_1 &= \sum_{i\in[K]} \upsilon_i - (\rho_{1,r} - \rho_{0,r}) \\
&\approx 1 - (1 - 0) = 0
\end{aligned}$$

The only slack in the above analysis is with regards to boundary conditions. Cleaning that, we can upper bound $\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right]$ as follows. Define

$$\beta_{i,r} = \begin{cases} 1 & \text{if } \frac{p_r}{p_r + 2^{-(m+\theta_m)}} < \frac{i}{K} \\ \rho_{\frac{i}{K},r} & \text{otherwise} \end{cases}$$

and

$$\omega_i = \begin{cases} 1 & \text{if } \frac{\delta K}{i-1} > \frac{1}{2} \\ \chi_{\frac{\delta K}{i-1}, \frac{(i-1)n}{K}} & \text{otherwise} \end{cases}$$

Then,

$$\Pr\left[\sum_{i\in[n]} Y_i \le \delta n\right] \le \sum_{i\in[K]} (\beta_{i,r} - \beta_{i-1})\,\omega_i = \epsilon_{\delta,r}$$

B.5 Linear Minimum Distance: Proof of Theorem 3

Proof. Let $\mathsf{d}(G)$ denote the minimum distance of the code G where $G \leftarrow \mathsf{ECGen}$. Using Theorem 2, we have

$$\Pr[\mathsf{d}(G) < \delta n - \hat{k}(m-1)] \le \sum_{r\in[k]} \binom{k}{r} \frac{1}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \epsilon_{\delta,r}$$

where $\tilde{\delta}_r = \frac{\hat{k}}{n\cdot 2^{-(m+\theta_m)}\cdot p_r}$.

First, consider $\tilde{\delta}_r = \frac{\hat{k}}{n\cdot 2^{-(m+\theta_m)}\cdot p_r}$. We would like to upper bound $\frac{1}{1-\exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)}$. To do so, we need to lower bound $\tilde{\delta}_r$. The best upper bound we have on p_r is $\frac{1}{2}$. Therefore, we set $m = \Theta(\log n)$. Let $m = C_m \log n$ for some $C_m > 1$. Then,

$$\tilde{\delta}_r \ge \frac{2\hat{k}}{\frac{n}{n^{C_m}}} = 2\hat{k}n^{C_m - 1}$$

We set $\hat{k} \ge n^{1-C_m}$. Then, $\tilde{\delta}_r \ge 2$. This implies that

$$\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r} = \frac{\hat{k}}{1 + \frac{2}{\tilde{\delta}_r}} \ge \frac{\hat{k}}{2}$$

Then,

$$\frac{1}{1 - \exp\left(-\frac{\tilde{\delta}_r \hat{k}}{2+\tilde{\delta}_r}\right)} \le \frac{1}{1 - e^{-\frac{\hat{k}}{2}}} \le 2$$

as long as $\hat{k} \ge 2\ln 2$. Therefore,

$$\Pr[\mathsf{d}(G) < \delta n - \hat{k}(m-1)] \le 2 \sum_{r\in[k]} \binom{k}{r} \epsilon_{\delta,r}$$

Note that $\hat{k}(m-1) = o(n)$ as long as $\hat{k} = o\left(\frac{n}{\log n}\right)$, and in this case, we will have a meaningful guarantee of linear minimum distance as $\delta n - \hat{k}(m-1) = \Theta(n)$.

We now consider the terms $\binom{k}{r}\epsilon_{\delta,r}$ for various values of r and bound the term in each case. We will set $w = \frac{C \ln n}{n}$ for the remainder of the analysis, for some $C \geq 2$. Recall that

$$\epsilon_{\delta,r} = \left(1 + 2^{m+\theta_m} p_r\right) \exp\left(-2n \left(\frac{1}{2} \cdot \frac{p_r}{p_r + 2^{-(m+\theta_m)}} - \delta\right)^2 \cdot \frac{p_r + 2^{-(m+\theta_m)}}{2 - p_r - 2^{-(m+\theta_m)}}\right)$$

Case $r = 1$. Here, $p_r = p_{\mathsf{w}} = \frac{w}{n}$ and $\frac{p_r + 2^{-(m+\theta_m)}}{2 - p_r - 2^{-(m+\theta_m)}} \geq \frac{p_r}{2} = \frac{C \ln n}{2n}$. Next,

$$\frac{p_r}{p_r + 2^{-(m+\theta_m)}} = 1 - \frac{1}{n^{C_m - 1} C \log n + 1} \geq 1 - \frac{1}{C \log n + 1} \geq 1 - \frac{1}{41}$$

for $n \geq 2^{20}$, as $C_m > 1$ and $C \geq 2$. Therefore, for $r = 1$,

$$\begin{aligned}\binom{k}{r}\epsilon_{\delta,r} &\leq k\left(1 + 2Cn^{C_m - 1} \ln n\right) \exp\left(-C\left(\frac{20}{41} - \delta\right)^2 \ln n\right) \\ &= R\left(1 + 2Cn^{C_m - 1} \ln n\right) n^{-C\left(\frac{20}{41} - \delta\right)^2 + 1}\end{aligned}$$

where $R = \frac{k}{n}$ is the rate of the code.

Case $2 \leq r \leq \frac{n}{2C \ln n}$. As before, $\frac{p_r + 2^{-(m+\theta_m)}}{2 - p_r - 2^{-(m+\theta_m)}} \geq \frac{p_r}{2}$. In this case,

$$p_r = \frac{1}{2} - \frac{\left(1 - \frac{2C \ln n}{n}\right)^r}{2} \geq \frac{1}{2} - \frac{\exp\left(-r\frac{2C \ln n}{n}\right)}{2} \geq \frac{1}{2} - \frac{1 - \frac{rC \ln n}{n}}{2} = \frac{rC \ln n}{2n}$$

as $e^{-z} \leq 1 - \frac{z}{2}$ for all $z \in [0, 1]$. We also have $p_r \leq \frac{1}{2}$. Finally, $\binom{k}{r} \leq \left(\frac{ek}{r}\right)^r = \left(\frac{eRn}{r}\right)^r$. Therefore,

$$\binom{k}{r}\epsilon_{\delta,r} \leq \left(\frac{eRn}{r}\right)^r \left(1 + \frac{n^{C_m}}{2}\right) \exp\left(-C\left(\frac{20}{41} - \delta\right)^2 \frac{r \ln n}{2}\right)$$

By taking logarithms, it is easy to see that the above bound reduces as r increases and therefore attains its maximum at $r - 2$. Thus, for $2 \leq r \leq \frac{n}{2C \ln n}$,

$$\binom{k}{r}\epsilon_{\delta,r} \leq \left(1 + \frac{n^{C_m}}{2}\right) n^{-C\left(\frac{20}{41} - \delta\right)^2 + 2}$$

where we have assumed that $R \leq \frac{2}{e}$.

Case $r \geq \frac{n}{2C \ln n}$. In this case,

$$p_r = \frac{1}{2} - \frac{\left(1 - \frac{2C \ln n}{n}\right)^r}{2} \geq \frac{1}{2} - \frac{\exp\left(-r\frac{2C \ln n}{n}\right)}{2} \geq \frac{1}{2} - \frac{\frac{1}{e}}{2} = \frac{1}{2}\left(1 - \frac{1}{e}\right)$$

So, $\frac{p_r+2^{-(m+\theta_m)}}{2-p_r-2^{-(m+\theta_m)}} \geq \frac{p_r}{2-p_r} \geq \frac{e-1}{e+1}$. We also have $p_r \leq \frac{1}{2}$. Finally, $\binom{k}{r} \leq 2^k = 2^{Rn}$. Therefore,

$$\binom{k}{r}\epsilon_{\delta,r} \leq 2^{Rn}\left(1+\frac{n^{C_m}}{2}\right)\exp\left(-\frac{e-1}{e+1}\left(\frac{20}{41}-\delta\right)^2 n\right)$$

Observe that if $C\left(\frac{20}{41}-\delta\right)^2 > 2$ and $R < \frac{1}{\ln 2}\cdot\frac{e-1}{e+1}\left(\frac{20}{41}-\delta\right)^2$, then the upper bound above is smaller than the bound in the previous case for sufficiently large n. Thus, for $r \geq \frac{n}{2C\ln n}$,

$$\binom{k}{r}\epsilon_{\delta,r} \leq \left(1+\frac{n^{C_m}}{2}\right) n^{-C\left(\frac{20}{41}-\delta\right)^2+2}$$

for sufficiently large n, where we have assumed that $C\left(\frac{20}{41}-\delta\right)^2 > 2$ and $R < \frac{1}{\ln 2}\cdot\frac{e-1}{e+1}\left(\frac{20}{41}-\delta\right)^2$.

Therefore,

$$\begin{aligned}\Pr[\mathsf{d}(G) < \delta n - \hat{k}(m-1)] &\leq 2\sum_{r\in[k]}\binom{k}{r}\epsilon_{\delta,r} \\ &\leq 2k\left(1+\frac{n^{C_m}}{2}\right) n^{-C\left(\frac{20}{41}-\delta\right)^2+2} \\ &\leq 2Rn\cdot n^{C_m}\cdot n^{-C\left(\frac{20}{41}-\delta\right)^2+2}\end{aligned}$$

which completes the proof. □

C Algorithms

We present the algorithms referenced in Section 5.2.5 here (due to lack of space).

D Extended Discussion of Silver

D.1 Silver's Analysis of Minimum Distance

One could ask how such an oversight was made. One one hand, the authors of Silver clearly stated that the design has heuristic security and required more research to build confidence. It is our understanding that the authors focused on the characterization of minimum distance as the minimum summation columns in the parity check matrix. In this context it is more difficult to observe the existence of this linear system. In contrast, our attack is easy to observe once the basic task is to quickly drive the convolution state to zero. We first observed this attack only when you consider scaling n be to exponential. In such a setting this system becomes more apparent.

Algorithm 1 Randomized process with same distribution as the irreversible chain (Figure 1). S_0 samples the length of the sub-walk at state 0_m while S_1 samples the length outside state 0_m.

Input: $S_0, S_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$. **Output**: $\vec{x} \in \{0,1\}^n$.
Randomness: $(\mathsf{p}_0, \ldots, \mathsf{p}_*), (\mathsf{q}_0, \ldots, \mathsf{q}_*) \leftarrow [0,1]^*_{\mathbb{R}}$
$\vec{x} \leftarrow \{0,1\}^0; i, j := 0;$
while $|\vec{x}| \leq n$ **do**
  $\vec{t} \leftarrow 1;$
  while $|\vec{t}| < S_1(\mathsf{p}_{i+1}) - m - 1$ **do**
    if $t_{|\vec{t}|-[m-1]} = 0^{m-1}$ **then**
      $\vec{t} := \vec{t}||1;$
    else
      $\vec{t} := \vec{t}||\mathsf{Ber}_{\frac{1}{2}}(q_{j+|\vec{t}|});$
    end if
  end while
  if $S_1(\mathsf{p}_{i+1}) \neq m$ **then**
    $\vec{t} := \vec{t}||1;$
  end if
  $\vec{x} := \vec{x}||0^{S_0(\mathsf{p}_i)}||\vec{t}||0^{m-1};$
  $i := i + 2; j := j + |\vec{t}|$
end while
Return $\vec{x}_{[n]}$

Algorithm 2 Randomized process with same distribution as the reversible chain.

Input: $S'_0, S'_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$. **Output**: $\vec{x} \in \{0,1\}^n$.
Randomness: $(\mathsf{p}_0, \ldots, \mathsf{p}_*), (\mathsf{q}_0, \ldots, \mathsf{q}_*) \leftarrow [0,1]^*_{\mathbb{R}}$
$\vec{x} \leftarrow \{0,1\}^0; i, j := 0;$
while $|\vec{x}| \leq n$ **do**
  $\vec{t} \leftarrow \{0,1\}^0;$
  while $|\vec{t}| < S'_1(\mathsf{p}_{i+1})$ **do**
    $\vec{t} := \vec{t}||\mathsf{Ber}_{\frac{1}{2}}(q_{j+|\vec{t}|});$
  end while
  $\vec{x} := \vec{x}||0^{S'_0(\mathsf{p}_i)}||\vec{t};$
  $i := i + 2; j := j + |\vec{t}|;$
end while
Return $\vec{x}_{[n]};$

To compensate for their lack of theoretical bounds Silver employed extensive use of empirical methods to estimate the minimum distance for small codes. These methods themselves still appear effective at estimating the minimum distance in light of our attack. However, these methods have a running time exponential in the length of the code. As a result Silver was only able to evaluate the codes of size up to $n = 800$ and observed minimum distance up to 140. However, our attacks show that the minimum distance of these codes stop growing

at approximately[8] 705 or 4,158 depending on the variant. This is well beyond what their empirical methods would be able to detect and exactly the central fear in their methodology. As such, the methodology of measuring the minimum distance for small codes and extrapolating appears too fragile to be considered as strong evidence for linear minimum distance.

In light of this one might conclude that the approach taken by Silver was miss guided. However, we believe this overlooks several innovative ideas put forth. First is the idea of using automated tools to help guide the design of certain aspects of the code. This had two impacts on the Silver code. First is that they changed the accumulator A that was used in the [21] codes (the inspiration for Silver) to be a convolution. The change was motivated and guided by their empirical methods and only in retrospect it was identified as a type of convolutional code which have previously been studied[6]. Looking forward we will show that this change can, if properly designed, significantly improve the performance/minimum distance of the code.

D.2 Structural Weaknesses

Another important question is if there exist alterations to Silver that would restore the conjectured (or provable) linear minimum distance. Unfortunately, this appears to be very unlikely without very significant changes to the structure of the code. In particular, one perspective on Silver is that it performs *only* local operations. This was an implicit design goal of Silver as cache efficiency was identified as a critical metric for achieving high performance.

Recall that the input message is first shifted and added to itself, i.e. $y = xL$. While these shifts are somewhat non-local, the overall structure of x is preserved. The convolution can then be computed as a small sliding window of size s over y which is essentially a purely local operation. As such, the convolution has what we call *non-locality* distance of m.

To stress this point, an alternative method of computing Silver would be to as

$$\mathsf{shift}(\vec{x}, q_1)A + ... + \mathsf{shift}(\vec{x}, q_w)A$$

Each term has distance m and therefore composing them can only increase the *non-locality* distance by w. It stands to reason that any code that can be computed with $w = O(\log n)$ passes over the input with each pass keeping at most $w = O(\log n)$ internal state can not have linear minimum distance.

Indeed, [6] considers a related and stronger class of codes known as parallel turbo codes. These codes can be characterized as $(\vec{x}, \vec{x}\pi_1 A, ..., \vec{x}\pi_w A)$ where π_i are random permutations. For a fixed message size k, it is not hard to see that such codes should be strictly stronger than Silver due to the permutations essentially mixing $\vec{x}$ much more than shifting it can. However, even these stronger codes are known not to achieve linear minimum distance when w is relatively small, i.e. $w \in \{5, 11\}$ as Silver specifies. The proof of this follows a similar

[8] Computed as $\frac{sw^2+sw}{2}$.

structure as our analysis of Silver. The primary difference is that the addition of the permutation complicates the analysis in that each x_i is distributed to w locations in a random manner. Nevertheless, even with such permutations one can show that there will exist x_i which are all mapped to a small number of regions and from there the same analysis as Silver applies.

E Relating Wrapping and Non-wrapping Chains

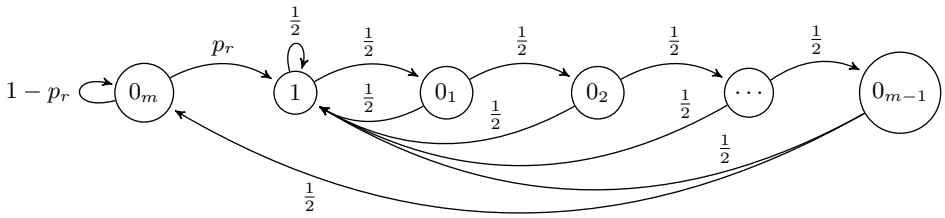

Fig. 8. The $(m+1)$-state non-wrapping Markov chain representing the encoding process.

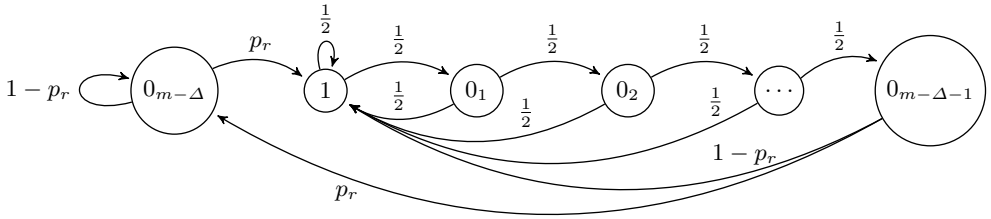

Fig. 9. The $(m - \Delta + 1)$-state wrapping chain representing the encoding process.

The wrapping chain has a lower probability of returning from the end of the chain back to the very first 0 state, namely, p_r instead of $\frac{1}{2}$. Intuitively, it should be possible for us to leverage this to remove some of the 0s from the tail of the long chain. For instance, if $p_r = \frac{1}{4}$, then we should be able to get rid of the state 0_{m-1} in the non-wrapping chain. Ahead, we prove exactly this fact formally.

Theorem 4. *Let n denote the length of the random walks performed on the chains in Figures 8 and 9. Starting from state 0_m of the non-wrapping chain (Figure 8), let X_i be the indicator of being in state 1 at step i. Starting from*

state $0_{m-\Delta-1}$ *of the wrapping chain (Figure 9), let* Y_i *be the indicator of being in state* 1 *at step* i*. If* $p_r \leq \frac{1}{2^{\Delta+1}}$*, then for all* $\delta \in [0,1]$*, we have:*

$$\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] \leq \Pr\left[\sum_{i\in[n]} X_i \leq \delta n\right]$$

Lemma 8. *Denote* $\tilde{X}$ *the random variable described by Process 3, then* $\tilde{X}$ *follows the same distribution as a walk on the non-wrapping chain.*

Proof. At a high level, a random walk on the non-wrapping chain can be described completely as: *"The walk left* 0 *after* i_1 *steps, then went back to* 1 *after* j_1 *steps,* 1 *after* j_2 *steps, . . . "* where the number of steps i_k, j_k are drawn from the correct distribution. Moreover, the non-zero sub-walk is uniformly sampled conditioned on not getting m zeros in a row (except at the end) as is required. By construction, our process samples the number of steps for each of these *sub-walks* exactly as per the distribution induced by the transition probabilities in the non-wrapping chain.

The proof of this lemma is similar to the proof of Lemma 5 (see Section B.2). □

Process 3 (Non-wrapping Chain)**.** *For each* $i \in \mathbb{N}$*, let* $\tau_i = (1-p_r)^{i-1}p_r$ *and* $\eta_i = \sum_{j=1}^{i-1} \tau_i$*. So, we have* $\eta_1 = 0$*,* $\eta_2 = \tau_1$*,* $\eta_3 = \tau_1 + \tau_2$*, . . .,*

$$\sum_{i=1}^{\infty} \tau_i = p_r \sum_{i=0}^{\infty} (1-p_r)^i = \frac{p_r}{1-(1-p_r)} = 1$$

That is, $\eta_i \to 1$ *as* $i \to \infty$*. Also,* η_i *is trivially monotonically increasing. Define the monotonically increasing step function* $S_0 : [0,1]_{\mathbb{R}} \to \mathbb{N}$ *where* $S_0(0) = 1$ *and for* $p \in (0,1]_{\mathbb{R}}$*,* $S_0(p) = k$ *such that* $p \in (\eta_k, \eta_{k+1}]$*.*

Define $\zeta_1 = 0$ *and* $\zeta_{i+1} = \sum_{j=1}^{i} \frac{1}{2^j}$ *for* $1 \leq i \leq m$ *and* $\zeta_{m+2} = 1$*. We also have*

$$\zeta_{m+1} = \sum_{j=1}^{m} \frac{1}{2^j} = 1 - \frac{1}{2^m}$$

That is, $\zeta_{m+2} - \zeta_{m+1} = \frac{1}{2^m}$*. Also,* ζ_i *is trivially monotonically increasing. Define the monotonically increasing step function* $S_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$ *where* $S_1(0) = 0$ *and* $S_1(p) = k-1$ *such that* $p \in (\zeta_k, \zeta_{k+1}]$ *for* $1 \leq k \leq m+1$*.*

Let $\tilde{X}$ *be the random variable denoting the output of Algorithm 3 which on input* S_0, S_1 *samples a walk by alternating calls to* S_0, S_1 *and constructing sub-walks of that length.*

Lemma 9. *Denote* $\tilde{Y}$ *the random variable described by Process 4, then* $\tilde{Y}$ *follows the same distribution as a walk on the wrapping chain.*

Proof. Same as Lemma 8. □

Process 4. *Let* $S'_0 = S_0$. *Define* $\iota_1 = 0$ *and* $\iota_{i+1} = \sum_{j=1}^{i} \frac{1}{2^j}$ *for* $1 \leq i \leq m - \Delta - 1$, $\iota_{m-\Delta+1} = \iota_{m-\Delta} + \frac{1-p_r}{2^{m-\Delta-1}}$, *and* $\iota_{m-\Delta+2} = 1$. *We also have*

$$\iota_{m-\Delta} = \sum_{j=1}^{m-\Delta-1} \frac{1}{2^j} = 1 - \frac{1}{2^{m-\Delta-1}}$$

That is, $\iota_{m-\Delta+2} - \iota_{m-\Delta+1} = \frac{p_r}{2^{m-\Delta-1}}$. *Also,* ι_i *is trivially monotonically increasing. Define the monotonically increasing step function* $S'_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$ *where* $S'_1(0) = 0$ *and* $S_1(p) = k - 1$ *such that* $p \in (\iota_k, \iota_{k+1}]$ *for* $1 \leq k \leq m - \Delta + 1$. *Let* $\tilde{Y}$ *be the random variable denoting the output of Algorithm 4 on input* S'_0, S'_1.

Algorithm 3 Randomized process with same distribution as the non-wrapping chain (Figure 8). S_0 samples the length of the sub-walk at state 0_m while S_1 samples the length outside state 0_m.

Input: $S_0, S_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$. **Output**: $\vec{x} \in \{0,1\}^n$.
Randomness: $(\mathsf{p}_0, \ldots, \mathsf{p}_*) \leftarrow [0,1]^*_{\mathbb{R}}$
$\vec{x} \leftarrow \{0,1\}^0; i := 0;$
convolving := false
while $|\vec{x}| \leq n$ **do**
  if convolving **then**
    if $S_1(\mathsf{p}_i) = m$ **then**
      $\vec{x} := \vec{x} || 10^{m-1}$
      convolving = false
    else
      $\vec{x} := \vec{x} || 10^{S_1(\mathsf{p}_i)}$
    end if
  else
    $\vec{x} := \vec{x} || 0^{S_0(\mathsf{p}_i)}$
    convolving = true
  end if
  $i := i + 1$
end while
Return $\vec{x}_{[n]}$

Lemma 10. *Now consider random variable* Z *which outputs* $(Z_{\tilde{X}}, Z_{\tilde{Y}})$ *by following the two previous processes ran on the same randomness* $\mathsf{p}_0, \ldots, \mathsf{p}_*$. *If* $p_r \leq \frac{1}{2^{\Delta+1}}$, *we have:*

$$HW(Z_{\tilde{Y}}) \geq HW(Z_{\tilde{X}})$$

Proof. Note that the states 0_m and 0 are identical, and S_0 and S'_0 are partitioned similarly. Since we use the same randomness for both, the draws on S_0 and S'_0 will give the same number of 0s in both $Z_{\tilde{X}}$ and $Z_{\tilde{Y}}$. We also use the same randomness to determine the number of 1s inside the convolution (portion of

Algorithm 4 Randomized process with same distribution as the wrapping chain (Figure 9). S'_0 samples the length of the sub-walk at state $0_{m-\Delta}$ while S'_1 samples the length outside state $0_{m-\Delta}$.

Input: $S'_0, S'_1 : [0,1]_{\mathbb{R}} \to \mathbb{N}$. **Output**: $\vec{x} \in \{0,1\}^n$.
Randomness: $(\mathsf{p}_0, \ldots, \mathsf{p}_*) \leftarrow [0,1]^*_{\mathbb{R}}$

$\vec{x} \leftarrow \{0,1\}^0; i := 0;$
convolving := false
while $|\vec{x}| \leq n$ **do**
  if convolving **then**
    if $S'_1(\mathsf{p}_i) = m - \Delta$ **then**
      $\vec{x} := \vec{x}||10^{m-\Delta-1}$
      convolving = false
    else
      $\vec{x} := \vec{x}||10^{S'_1(\mathsf{p}_i)}$
    end if
  else
    $\vec{x} := \vec{x}||0^{S'_0(\mathsf{p}_i)}$
    convolving = true
  end if
  $i := i + 1$
end while
Return $\vec{x}_{[n]}$

the walk spent outside the $0_m/0_{m-\Delta}$ state) in the non-wrapping chain and in the wrapping chain.

We show that for the same p, we always have $S_1(p) \geq S'_1(p)$ and that the wrapping chain never goes back to the $0_{m-\Delta}$ state sooner than the non-wrapping one goes back to the 0_m state. We have $S_1(p) = S'_1(p)$ for $p \in \left[0, 1 - \frac{1}{2^{m-\Delta-1}}\right]$ by definition. For $p \in \left(1 - \frac{1}{2^{m-\Delta-1}}, 1 - \frac{p_r}{2^{m-\Delta-1}}\right]$, $S'_1(p) = m - \Delta - 1$ and $S_1(p) \geq m - \Delta - 1$ as long as $1 - p_r \geq \frac{1}{2} \iff p_r \leq \frac{1}{2}$. Finally, for $p \in \left(1 - \frac{p_r}{2^{m-\Delta-1}}, 1\right]$, $S'_1(p) = m - \Delta$ and $S_1(p) \geq m - \Delta$ as long as $p_r \leq \frac{1}{2}$. We thus have that $\forall p, S_1(p) \geq S'_1(p)$.

We know that the wrapping chain goes back to 0 when $p \in (\iota_{m-\Delta+1}, 1]$ and the non-wrapping when $p \in (\zeta_{m+1}, 1]$. By definition, we have $(\iota_i, \iota_{i+1}] = (\zeta_i, \zeta_{i+1}]$ for $i \in [1, m - \Delta - 1]$. To show that the wrapping chain never goes back to the $0_{m-\Delta}$ state sooner than the non-wrapping one goes back to the 0_m state, it thus suffices to have that the half-open line segment $(\zeta_{m-\Delta}, \zeta_{m+1}]$ be smaller than $(\iota_{m-\Delta}, \iota_{m-\Delta+1}]$. By definition, we have $\zeta_{m-\Delta} = \iota_{m-\Delta}$, so we only require $\zeta_{m+1} \leq \iota_{m-\Delta+1}$ which holds as long as $p_r \leq \frac{1}{2^{\Delta+1}}$.

This means that a draw on S_1 can only output an equally long or longer sequence than a draw on S'_1. Hence, we may have the same number or a smaller number of draws on S_1 than on S'_1. However, each draw produces e exactly one 1 in both S_1 and S'_1. Also, note that draws on S_0 and S'_0 produce only sequences of 0s and of equal length. Therefore, at the end of the processes, the number of 1s in $\tilde{X}$ equals the number of draws on S_1 which is equal to or smaller than the

number of draws on S'_1 which is equal to the number of 1s in $\tilde{Y}$, thus completing the proof. □

We are now ready to prove our theorem.

Proof. (Theorem 4) For clarity, we abuse notation and always consider the Hamming weight of the string that the random variables output.

$$\begin{aligned}
\Pr\left[\sum_{i\in[n]} Y_i \leq \delta n\right] &= \Pr\left[\tilde{Y} \leq \delta n\right] \\
&= \Pr\left[Z_{\tilde{Y}} \leq \delta n\right] \\
&\leq \Pr\left[Z_{\tilde{Y}} \leq \delta n | Z_{\tilde{Y}} \geq Z_{\tilde{X}}\right] \cdot \Pr\left[Z_{\tilde{Y}} \geq Z_{\tilde{X}}\right] \\
&= \Pr\left[Z_{\tilde{X}} \leq \delta n\right] \\
&\leq \Pr\left[\sum_{i\in[n]} X_i \leq \delta n\right]
\end{aligned}$$

□

In Tables 5,6,7 we compute the failure probability of our wrapping Expand-Convolute code for various parameters.

	δ	Ours ($R = \frac{1}{5}$)	Ours ($R = \frac{1}{2}$)
$C = 3$	0.005	$4.19 \cdot 10^{-12}$	$2.51 \cdot 10^{-11}$
	0.02	$1.46 \cdot 10^{-10}$	$7.10 \cdot 10^{-10}$
	0.05	$2.20 \cdot 10^{-8}$	$8.01 \cdot 10^{-8}$
$C = 2.5$	0.005	$8.11 \cdot 10^{-10}$	$2.57 \cdot 10^{-8}$
	0.02	$1.57 \cdot 10^{-8}$	$4.15 \cdot 10^{-7}$
	0.05	$1.04 \cdot 10^{-6}$	$2.11 \cdot 10^{-5}$
$C = 2.3$	0.005	$1.57 \cdot 10^{-8}$	$4.08 \cdot 10^{-7}$
	0.02	$2.39 \cdot 10^{-7}$	$5.27 \cdot 10^{-6}$
	0.05	$1.13 \cdot 10^{-5}$	$1.95 \cdot 10^{-3}$

Table 5. Failure probabilities for $k = 2^{20}$ for our wrapping construction with $m = 5$. We have expected row weight $w = C \ln n$ and we assume we start at stationary (see Section 5.2.8). We use $\tilde{\delta}$ such that $\tilde{\delta}n - \hat{k}(m-1) = \delta n$.

	δ	Ours ($R = \frac{1}{5}$)	Ours ($R = \frac{1}{2}$)
$C = 3$	0.005	$4.61 \cdot 10^{-14}$	$6.46 \cdot 10^{-13}$
	0.02	$1.63 \cdot 10^{-12}$	$1.85 \cdot 10^{-1111}$
	0.05	$2.59 \cdot 10^{-10}$	$2.18 \cdot 10^{-9}$
$C = 2.5$	0.005	$7.77 \cdot 10^{-11}$	$7.01 \cdot 10^{-10}$
	0.02	$1.52 \cdot 10^{-9}$	$1.15 \cdot 10^{-8}$
	0.05	$1.03 \cdot 10^{-7}$	$6.10 \cdot 10^{-7}$
$C = 2.3$	0.005	$1.52 \cdot 10^{-9}$	$1.15 \cdot 10^{-8}$
	0.02	$2.33 \cdot 10^{-8}$	$1.50 \cdot 10^{-7}$
	0.05	$1.14 \cdot 10^{-6}$	$5.82 \cdot 10^{-6}$

Table 6. Failure probabilities for $k = 2^{20}$ for our wrapping construction with $m = 16$. We have expected row weight $w = C \ln n$ and we assume we start at stationary (see Section 5.2.8). We use $\tilde{\delta}$ such that $\tilde{\delta}n - \hat{k}(m-1) = \delta n$.

	δ	Ours ($R = \frac{1}{5}$)	Ours ($R = \frac{1}{2}$)
$C = 3$	0.005	$1.60 \cdot 10^{-11}$	$2.561 \cdot 10^{-10}$
	0.02	$5.66 \cdot 10^{-10}$	$7.32 \cdot 10^{-9}$
	0.05	$8.99 \cdot 10^{-8}$	$8.61 \cdot 10^{-7}$
$C = 2.5$	0.005	$4.16 \cdot 10^{-8}$	$2.32 \cdot 10^{-7}$
	0.02	$8.11 \cdot 10^{-7}$	$3.79 \cdot 10^{-6}$
	0.05	$5.54 \cdot 10^{-5}$	$2.01 \cdot 10^{-4}$
$C = 2.3$	0.005	$7.48 \cdot 10^{-7}$	$3.49 \cdot 10^{-6}$
	0.02	$1.15 \cdot 10^{-5}$	$4.57 \cdot 10^{-5}$
	0.05	$5.60 \cdot 10^{-4}$	$1.77 \cdot 10^{-3}$

Table 7. Failure probabilities for $k = 2^{20}$ for our wrapping construction with $m = 10$. We have expected row weight $w = C \ln n$ and we assume we start at state 0 (see Section 5.2.8). We use $\tilde{\delta}$ such that $\tilde{\delta}n - \hat{k}(m-1) = \delta n$.