# Scrambling Adversarial Errors Using Few Random Bits, Optimal Information Reconciliation, and Better Private Codes

Adam Smith*
Weizmann Institute of Science
adam.smith@weizmann.ac.il

January 17, 2006

**Abstract**

When communicating over a noisy channel, it is typically much easier to deal with random, independent errors with a known distribution than with adversarial errors. This paper looks at how one can use schemes designed for random errors in an adversarial context, at the cost of relatively few additional random bits and without using unproven computational assumptions.

The basic approach is to permute the positions of a bit string using a permutation drawn from a $t$-wise independent family, where $t = o(n)$. This leads to two new results:

- We construct *computationally efficient* information reconciliation protocols correcting $pn$ adversarial binary Hamming errors with optimal communication and entropy loss $n(h(p) + o(1))$ bits, where $n$ is the length of the strings and $h()$ is the binary entropy function. Information reconciliation protocols are important tools for dealing with noisy secrets in cryptography; they are also used to synchronize remote copies of large files.
- We improve the randomness complexity (key length) of efficiently decodable capacity-approaching *private codes* from $\Theta(n \log n)$ to $n + o(n)$.

We also present a simplified proof of an existential result on private codes due to Langberg (FOCS '04).

# 1 Introduction

## 1.1 Partially Dependent Errors

Suppose Alice sends $n$ bits to Bob over a binary channel, and at most $\tau = pn$ of them are flipped. A code $C : \{0,1\}^k \to \{0,1\}^n$ can correct all such corruptions if and only if the Hamming distance between any two codewords (points in the image) is at least $2\tau + 1$. There are proven limits on how well such codes can perform, and known codes which can be decoded in polynomial time perform even more poorly. In contrast, codes which correct *random* errors (say, where each bit is flipped independently with probability $p$, or where a random subset of $pn$ bits is flipped) perform much better: there are explicit, polynomial-time decodable codes which transmit at rates arbitrarily close to the Shannon capacity $1 - h(p)$. [1] This is typically a factor of 2 greater than existential upper bounds on the performance of codes for adversarial errors; the advantage over known, polynomial-time decodable codes is even greater.

*Supported by the Louis L. and Anita M. Perlman Postdoctoral Fellowship.

[1] The function $h()$ is the binary entropy function $h(p) = -p \lg p - (1-p) \lg(1-p)$. All logarithms in this paper are base 2.

We show that for concatenated codes [For66], which can correct random errors at transmission rates arbitrarily close to the Shannon capacity, decoding continues to work with high probability *even when the errors are only (almost) $t$-wise independent* for $t = o(n)$. In other words, for this class of codes, the errors need only be slightly random in order to achieve rate close to $1 - h(p)$ (the entropy of an almost $t$-wise independent error distribution can be as low as $O(t)$). The proof consists of re-analyzing the performance of concatenated codes using bounds on sums of partially dependent random variables.

This observation leads to a general strategy for dealing with an adversarial channel, assuming – crucially – that the errors introduced by the adversary are independent of Alice's random coins. We will see two settings in which this assumption is justified: *information reconciliation* and a class of *private codes*. The general approach is:

1. Alice chooses at random a permutation $\pi : [n] \to [n]$ (where $[n] = \{1, \ldots, n\}$ from a family of permutations which is $t$-wise independent, meaning that the images of any $t$ indices $i_1, \ldots, i_t \in [n]$ look like a sample of $t$ points chosen uniformly without replacement from $[n]$. Such permutations can be chosen using only $O(t \log n)$ random bits (Kaplan, Naor and Reingold [KNR05])[2]. We refer to the algorithm mapping seeds to permutations as the *KNR generator*.

2. Alice now encodes her message $m$ using a concatenated code $C$ with rate close to capacity, permutes the bits of $C(m)$ using the inverse $\pi^{-1}$. We abuse notation and denote the permuted string by $\pi^{-1}(C(m))$. She sends $\pi^{-1}(C(m))$ through the channel to Bob.

3. Bob can 'unpermute' the corrupted word which he received by applying $\pi$ (this assumes that Alice can somehow send Bob the short description of $\pi$). Bob ends up with $C(m)$, *corrupted in a set of positions which was chosen (almost) $t$-wise independently.* That is, if the adversary added an error vector $e \in \{0,1\}^n$, the code is faced with decoding the error vector $\pi(e)$. By the result mentioned above, the code $C$ will correct the errors introduced by the adversary with high probability, and Bob can learn the original message $m$.

The idea of permuting a string to randomize errors and thus reach capacity is by no means new. However, previous approaches require choosing, and sending or storing, a fully random permutation [BBR88] (this requires $\Theta(n \log n)$ random bits) or assume the existence of a pseudo-random generator [Lip94, DGL04]. Our approach can be seen as a derandomization of the generic approach, replacing a cryptographic pseudo-random generator based on a hardness assumption with a specific, combinatorial generator tailored to this application. We now explain two applications of this idea in more detail.

## 1.2 Information Reconciliation

Suppose that Alice and Bob share an $n$-bit secret string. Alice's copy $w$ of the shared string is slightly different from Bob's copy $w'$. Alice would like to send a short message $S(w)$ to Bob which allows him to correct the errors in $w'$ (and thus recover $w$) whenever $w$ and $w'$ differ in at most $\tau$ bits. The randomized map $S()$ that Alice applies to $w$ is called a *non-interactive information reconciliation scheme*, or simply a *sketch*, correcting $\tau$ binary Hamming errors.[3] A typical example of a sketch is

$$S(w) = \mathsf{syn}_C(w),$$

where $\mathsf{syn}_C$ is the syndrome of a linear error-correcting code $C$ with block length $n$ [BBR88]. (The syndrome is the linear map given by the parity check matrix of the code.) If $C$ has dimension

[2]The constructions of [KNR05] only produce *almost* $t$-wise independent permutations. See Section 3.2 for details.

[3]This is a different use of the term "sketch" than one sometimes sees in the algorithms literature, where it means, roughly, a short string allowing one to estimate distances between a particular vector and other points.

$k$, then $\mathsf{syn}_C(w)$ is only $n-k$ bits long. If the minimum distance of $C$ is at least $2\tau+1$, then $\mathsf{syn}_C(w)$ allows Bob to correct any $\tau$ errors in $w'$. Moreover, the correction process is efficient (polynomial-time) if one can correct $\tau$ channel errors using the code in polynomial time.

Formally, a sketch consists of two (randomized) algorithms "sketch" ($S : \{0,1\}^n \to \{0,1\}^\ell$) and "recover" ($\mathsf{Rec} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$). The parameter $\ell$ is called the length of the sketch.

**Definition 1.** *A sketch corrects $\tau$ adversarial errors with probability $1-\epsilon$ if for* all *pairs $w, w' \in \{0,1\}^n$ which differ in at most $\tau$ positions, $\mathsf{Rec}(w', S(w)) = w$ with probability at least $1-\epsilon$ over the random coins of $S$ and* Rec.

No guarantee is provided about the output of Rec when the distance between $w$ and $w'$ is more than $\tau$. Also, we assume that $w'$ is chosen before the value $S(w)$ (that is, the pair $w, w'$ is independent of the coins of $S(w)$).

Sketches are useful for cryptographic settings where secrets may be subject to noise, such as when keys come from biometrics or other measurements [JW99, JS02, DRS04, BDK+05], quantum cryptography [BBR88, BBCM95, BS92], Maurer's bounded storage model [Din05, DS05], and several variants on settings with correlated randomness, e.g. [RW04, RW05].[4] They have also been considered in communication complexity, e.g. [Orl92, Orl93, CPSV00, MT02, MTZ03].

We focus here on the *length* of the sketch: how many bits must Alice send to Bob? When $w$ is drawn uniformly from $\{0,1\}^n$, at least $nh(p)(1-o(1))$ bits are necessary if the scheme corrects $\tau = pn$ errors: $S(w)$ allows Bob to distinguish $w$ from all other strings at distance $pn$ from $w'$, that is from about $2^{nh(p)}$ candidates. The same bound applies to the *entropy loss* of the sketch, which we discuss at the end of this section.

Techniques from previous work allow one to construct several protocols matching this bound [BBR88, Lip94, DGL04, Gur03, DRS04, RW05]. The results are not stated explicitly in the form that interests us; in Section 3.1 we cast them in the setting of reconciliation protocols and compare the parameters they achieve. To our knowledge, all of them either

- work only for a known distribution on the error vector $w \oplus w'$ (typically, the errors are assumed to be independent),
- require an exponential-time computation on Bob's part to recover $w$, or
- assume the existence of pseudo-random generators.

The general approach of the previous section yields the first sketch for binary strings which solves all the problems above: the sketch has length $n(h(p)+o(1))$, makes no assumptions on the distribution of the pair $w, w'$ except that the Hamming distance $\mathsf{dist}(w, w')$ is bounded, allows Bob to recover $w$ with all but negligible probability in polynomial time, and requires no unproven computational assumptions. Here we give a high-level description:

**Protocol 1.** *Given a parameter $\delta > 0$:*
*(Step 1) Alice chooses a permutation $\pi$ which is almost $t$-wise independent for $t = \delta n/\log(n)$. Using the KNR permutation generator, this takes only $O(t \log n) = O(\delta n)$ bits. (Step 2) Alice permutes the bits of $w$ using $\pi^{-1}$ (the inverse is for technical reasons), and encodes the result using*

[4] In some of these settings, Bob is simply Alice at a later point in time. The sketch is "transmitted" by storing it– and so having the sketch be non-interactive is important. Nevertheless, the lower bound of $nh(p)$ discussed below on the communication complexity and entropy loss of sketches applies equally well to interactive schemes, so the protocols we discuss are optimal even if interaction is possible. There are some reconciliation settings, not directly relevant to our discussion, where interaction is known to help; see, for example, the work of Orlitsky [Orl90].

*the concatenated code $C$ described above which has rate $(1 - h(p) - \delta)$ and efficiently corrects any $t$-wise independent error distribution. (Step 3) Alice sends*

$$S(w) = \underbrace{\textit{description of } \pi}_{O(\delta n)\ \textit{bits}}, \underbrace{\mathsf{syn}_C(\pi(w))}_{n(h(p)+\delta)\ \textit{bits}}.$$

This sketch corrects $pn$ errors with probability $1 - \exp(\Omega(\delta^3 n/\log n))$, as long as $\delta$ is not too small (larger than some constant times $\operatorname{loglog} n/\sqrt{\log n}$). See Section 3.2 for a precise analysis.

The Relation to Entropy Loss In cryptographic settings, a sketch is usually used as the first step of a protocol, following which Alice and Bob derive a shared, secret key from $w$. Suppose that Eve is tapping the line and trying to learn as much as possible. The most important parameter of a sketch is then the *entropy loss* of the scheme. Entropy loss can be defined in several ways (see, e.g., [DRS04, RW05]) but in all cases it can be interpreted as the amount of "information," in bits, revealed about $w$ by $S(w)$. Typical proofs of security bound the entropy loss of a scheme simply by bounding *the number of bits sent from Alice to Bob which depend on the message $w$.* [5] This suffices as a definition of entropy loss for our purposes. We refer the reader to [DRS04, RW05] for discussions of the correct general definition.

The lower bound of $n(h(p) - o(1))$ applies to the entropy loss as well as the length of a sketch (again, by considering the special case of random, independent errors). Since the length of a sketch provides an upper bound on its entropy loss, communication-optimal sketches are also optimal for entropy loss. Nevertheless, some sketches from previous work are optimal for entropy loss but not communication, and so we include entropy loss as a separate entry in our comparison of various protocols (Table 1 in Section 3.1).

### 1.3 Private Codes

*Private codes* were named and first studied explicitly by Langberg [Lan04], with the goal of communicating at the Shannon capacity even in the face of adversarial errors. The idea is that if Alice and Bob share a key ahead of time —secret from the adversary controlling the channel— then they can send information at a higher rate than they could by using a standard code.

**Definition 2.** *An adversarial channel introducing $pn$ errors is a randomized map $\mathcal{N} : \{0,1\}^n \to \{0,1\}^n$ such that for all inputs $w$, the distance $\mathsf{dist}(w, \mathcal{N}(w))$ is at most $pn$ with probability 1.*

**Definition 3.** *A $[n, k, \ell_{key}]$* private code *is a pair of algorithms $PC, D$, where $PC : \{0,1\}^k \times \{0,1\}^{\ell_{key}} \to \{0,1\}^n$ and $D : \{0,1\}^n \times \{0,1\}^{\ell_{key}} \to \{0,1\}^k$. The private code corrects $pn$ errors with probability $1 - \epsilon$ if for all messages $x \in \{0,1\}^k$ and for all adversarial channels $\mathcal{N}$ introducing at most $pn$ errors, the probability over $r$ that $D(\mathcal{N}(PC(x; r)); r) = x$ is at least $1 - \epsilon$. The private code is* efficient *if both $PC$ and $D$ run in polynomial time.*

Efficient Private Codes For any $p \in (0, 1)$, there are *efficient* private codes which achieve capacity, in the sense that they correct $pn$ errors with high probability and transmit a message of $k \approx n(1 - h(p))$ bits. The rate is optimal since sharing a secret key does not increase the capacity of a binary symmetric channel, and a private code must, in particular, correct binary symmetric errors with high probability.

A construction of such private codes is implicit in [DGL04] and described in detail in [Lan04]. The scheme has key length $\ell_{key} = \Theta(n \log n)$, since the key contains a uniformly random permutation

[5] There are exceptions. For example, the analysis of the code-offset scheme for non-linear codes in [DRS04] does not have so simple a form, but no instances are known where the generality of non-linear codes is useful.

of $[n]$. The general strategy described in Section 1.1 applies here: the uniform random permutation can be replaced by an almost $t$-wise independent permutation, reducing the key length from $n \log n$ to $n + o(n)$. We obtain the following protocol:

**Protocol 2.** *Given a shared key $(a, s)$, where $a \in \{0,1\}^n, s \in \{0,1\}^{\delta n}$:*
*(Step 1) Alice runs KNR generator with $t = \delta n / \log n$ on seed $s$ to obtain a permutation $\pi_s : [n] \to [n]$. (Step 2) Alice encodes $x$ using the concatenated code $C$ of Section 1.1 for partially dependent errors. (Step 3) Alice sends:*

$$PC(x; a, s) \quad = \quad \pi_s^{-1}(C(x)) \oplus a.$$

*To decode the received string $y$, Bob runs the decoder for $C$ on $\pi(y) \oplus a$.*

The scheme corrects $pn$ adversarial errors with error $\epsilon = \exp(-\Omega(\delta^3 n / \log n)$. The idea of the analysis is that the mask $a$ eliminates all dependency between the errors in the channel and the permutation $\pi_s$, and thus we can apply the general approach from Section 1.1. See Section 4 for a precise analysis and a comparison to parameters achieved in previous work.

Inefficient Schemes Based on List-Decoding For any $p \in (0,1)$, there *exist* private codes which correct $pn$ errors, transmit a message of $k \approx n(1 - h(p))$ bits, and use a very short secret key: only $4 \log n + 2 \log\left(\frac{1}{\epsilon}\right)$ bits (Langberg, [Lan04]). This amount of randomness is optimal up to constant factors [Lan04].

In Appendix A, we present a simplified proof of Langberg's result, based on a cryptographic intuition. Roughly: given a good list decodable code $LDC$, and a message authentication scheme (MAC), Alice can use the shared randomness as the MAC key and send the encoding of the original message $x$ together with the tag given by the MAC:

$$PC(x; r) = LDC(x, mac(x; r)).$$

Incorrect decoding is roughly equivalent to a forged MAC tag, which occurs with very low probability. The problem with this construction is that it is not computationally efficient. The scheme relies on the existence of a good list-decodable code but, as discussed in Section 3.1, polynomial-time list-decodable codes that approach capacity are only known in the extreme ranges of $p$ ($p \approx 0$ or $p \approx \frac{1}{2}$). The scheme of Protocol 2 is, surprisingly, the best known (capacity-achieving) private code with efficient decoding.

### 1.4 Organization of this Paper

Section 2 presents a family of capacity-approaching concatenated codes and shows that $t$-wise independence of the errors is sufficient to guarantee correct decoding with high probability. Section 3 of the paper discusses communication-optimal reconciliation schemes. We first discuss protocols which follow directly from ideas in previous work (Section 3.1), and then analyze the performance of Protocol 1 (Theorem 4). Finally, Section 4 we look at the application to private codes (Theorem 5). Langberg's existential construction of good private codes can be found in Appendix A.

## 2 Codes for Partially Dependent Errors

**Definition 4.** *A random variable $E = (E_1, \ldots, E_n)$ on $\{0,1\}^n$ is* $t$-wise independent of weight $pn$ *if for any set $I = \{i_1, \ldots, i_t\}$ of $t$ indices in $[n]$, the restriction $E|_I \stackrel{def}{=} (E_{i_1}, \ldots, E_{i_t})$ has statistical distance at most $2^{-t}$ from the distribution it would have if $E$ was chosen uniformly from the set of binary vectors with exactly $pn$ ones.*

We say a code $C$ corrects an error distribution $E$ with probability $1-\epsilon$ if there is a decoding algorithm $Dec$ such that, for all messages $m$, the probability over $E$ that $Dec(C(m) \oplus E) = m$ is at least $1-\epsilon$. [6]

**Theorem 1.** *For any $\delta = \Omega(\log\log n/\sqrt{\log n})$, there is a family of linear concatenated codes of length $n$ and rate $R = 1 - h(p) - \delta$ which corrects* all *$t$-wise independent error distributions of weight $pn$ with probability $1 - 2^{-\Omega(\delta^2 t)}$ as long as $\omega(\log n) < t < \delta n/10$.*

We will typically set $t = \delta n/\log n$, in which case the error probability of the decoding scales as $2^{-\Omega(\delta^3 n/\log n)}$.

## 2.1 Underlying Concatenated Code

The code construction has two parameters: the block size $b$ in bits, which is at most $O(\log n)$, and the overhead $\delta$, which determines the rate $R = n(1-h(p)-\delta)$. The block length will ultimately be set to about $\log\left(\frac{1}{\delta}\right)/\delta^2$, although we will leave it as an explicit parameter to make the construction easier to follow.

Inner Code: a linear code of block length $b$ and rate $R_0 = 1 - h(p) - \delta/2$ which corrects random errors with probability $\mu = 2^{-\Omega(\delta^2 b)}$. We will later choose $b$ large enough so that $\mu \le \delta/10$.

This code is a linear code selected by exhaustive search (see details below). It can be decoded in time $\text{poly}(2^b) = \text{poly}(n)$.

Outer Code: A linear code with length $n_1 = n/b$ and rate $R_1 = 1 - \delta/2$ over an alphabet of size $2^{R_0 b}$, which corrects $\Omega(\delta n')$ adversarial errors in polynomial time. For concreteness, say the code corrects $\delta n_1/5$ errors.

We can use a Reed-Solomon or algebraic-geometric code (details below). These codes can be decoded in time $O(n^3)$ and even, in some cases, in time $O(n \operatorname{polylog} n)$.

The final code is the concatenation of the codes above: the message is first encoded with the outer code, and then each of the symbols is encoded with the inner code. The final block length is $n = n_1 b$, and the final rate is $R_0 R_1 \ge 1 - h(p) - \delta$, as desired.

By a line of analysis due to Forney [For66], and which is now standard, this code will correct random, independent errors well. The decoder simply attempts to correct errors in each of the blocks separately, by exhaustive search. Each block corrects all of its errors with probability $1-\mu$, where $\mu = 2^{-\Omega(\delta^2 b)}$. The outcome of the decoding operation within each block is independent of all the other blocks, and so with overwhelming probability, the number of incorrectly decoded blocks is at most, say, $2\mu n_1$. The decoder then concatenates the resulting messages and attempts to decode the result using the outer code.

We chose $\mu < \delta/10$ and so the high probability bound on the number of bad blocks, $2\mu n_1$, is at most the error-correction threshold of the outer code, $\delta n_1/5$. As long as the $2\mu n_1$ bound is not exceeded, the outer code corrects all the blocks which decoded incorrectly in the first phase, and recovers the original message. The overall decoding error of this process is $2^{-\Omega(\delta^2 n)}$.

Note that this is the simplest approach to decoding a concatenated code (it is called *hard decoding*), but it is sufficient for our purposes since we do not attempt to optimize the constants in the exponent.

We now give more detailed information on the component codes of this construction.

[6] Note that for linear codes, the quantification over $m$ is unimportant since the code is invariant under translation; we can typically assume that $C(m) = 0^n$ without loss of generality.

INNER CODE DETAILS: RANDOM LINEAR CODES. For any $\epsilon_1, R_0, p$, and for sufficiently large $b$, there exists a linear code with block length $b$ and rate $R_0 = 1 - h(p) - \rho$ which corrects binary symmetric errors with flip parameter $p$ with error probability at most $2^{-\rho^2 b/3}$. Such a code can be found with high probability in time $\text{poly}(2^b)$. The code can be encoded in time $b^2$ and decoded in time $2^{b(h(p)+\rho)}$ by exhaustive search. For an analysis of the performance of random coding, see, e.g., [CT91]. The extension to linear codes follows by noting that pairwise independence of the codewords is sufficient for the proofs to work, and the running time of code construction comes from considering a $2^b$-size subclass of linear codes such as Wozencraft codes [vL92].

OUTER CODE DETAILS: RS AND AG CODES For any length $n$, dimension $k \leq n$ and prime power alphabet size $q > n$, Reed-Solomon codes are poly-time constructible and decodable $[n, k, d]_q$ codes with distance $d = n - k + 1$. For any length $n$, dimension $k \leq n$ and (possibly very small) alphabet size $q$ which is a prime power squared, there exists a poly-time constructible and decodable family of $[n, k, d]_q$ codes with $d \geq n - k - \frac{n}{\sqrt{q}-1} + 1$. See [vL92, Sti93] for details.

If the dimension $R_0 b$ is of the inner code is large enough (at least $\log(n_1)$), then one can use a Reed-Solomon code, which corrects $\delta n_1/4$ errors. When $b$ is smaller, one may use an algebraic-geometric code, which corrects $\frac{1}{4} \cdot n_1(\delta - 1/\sqrt{2^{R_0 b}})$ errors (we need to round $R_0 b$ to an even integer, but this is not a problem as it is at least some large constant). We will later set $b$ so that the number of corrected errors is at least $\delta n_1/5$.

## 2.2 Code Performance with Partially Dependent Errors

We now prove that the hard decoding algorithm mentioned above performs well even when the errors are only $t$-wise independent. The main tool is a Chernoff-like bound on a sum of almost $t$-wise independent variables. Suppose we have events denoted by binary random variables $X_1, \ldots, X_n$ such that every subset of $t$ of the events is close to independent, in the sense that the probability that they occur simultaneously is not too high. Then the sum of those variables is tightly concentrated about its mean. The bound we use here is due to Schmidt *et al.* [SSS95, Eqn. (2)]. It also appears in Ding *et al.* [DHRS04, Lemma 5.9].

**Fact 2 ([SSS95, DHRS04]).** *Suppose $X_1, \ldots, X_n$ are binary random variables, each with expectation at most $\mu$, such that for every set of $t$ indices $i_1, \ldots, i_t$, we have $\Pr[X_{i_1} = \cdots X_{i_t} = 1] \leq 2\mu^t$. Then for $B > 1$, the probability that the sum $\sum_{i=1}^n X_i$ exceeds $B\mu n$ is at most $2\left(B - \frac{t}{\mu n}\right)^{-t}$.*

Let $E$ be any $t$-wise independent error distribution of weight $\tau n$, as in Definition 4. We will need two claims relating the performance on $E$ to the case of independent errors:

**Lemma 3.**

1. *In each block, the probability of a decoding error with errors from $E$ is at most twice the probability of a decoding error with random independent errors.*

   *Consequently, each block is wrong with probability at most $\mu = 2^{-\Omega(\delta^2 b)}$.*

2. *For any subset of $t/b$ of the blocks, the probability over $E$ that a decoding mistake occurs simultaneously in all blocks is at most $2\mu^{t/b}$.*

We first use the lemma to prove Theorem 1, and then give the proof of the lemma.

*Proof of Theorem 1.* Let $X_1, \ldots, X_{n_1}$ be binary random variables which indicate whether a decoding error occurred in each of the blocks. By part (1) of the lemma, the expectation of $X_i$ is at most

$\mu$. Now look at any subset of $t' = t/b$ blocks. By part (2) of the lemma, the probability that all $X_i$'s in that set occur simultaneously is at most $2\mu^{t'}$. Thus, the $X_i$'s satisfy the conditions of the tail bound above (Fact 2) with parameters $n_1, t'$ and $\mu$. We are interested in the probability that more than $\delta n_1/5$ errors occur, so $B = \delta/(5\mu)$. The probability of a global decoding error is at most $2\left(\frac{\delta}{5\mu} - \frac{t/b}{\mu n_1}\right)^{-t/b}$. When $t < \delta n/10$, this is at most $2\left(\frac{10\mu}{\delta}\right)^{t/b}$. Recall that $\mu$, the probability of a mistake in each block, is $2^{-\Omega(\delta^2 b)}$. By making $b$ sufficiently large (a constant times $\log\left(\frac{1}{\delta}\right)/\delta^2$), the fraction $\mu/\delta$ is dominated by the block decoding error $\mu$, and so the global decoding error is at most $2^{-\Omega(\delta^2 t)}$, as desired. Since we assumed that $\delta = \Omega(\text{loglog}\, n/\sqrt{\log n})$, the block length $b$ is always $O(\log n)$ and so decoding can always be done in polynomial time. □

*Remark* 2.1. The constants in the proof above are fairly easy to calculate. A random linear code of length $b$ and rate $1 - h(p) - \rho$ corrects random, independent errors with probability at least $1 - 2^{-\rho^2 b/3}$. Going through the proof above, one gets that $b$ should be about $24 \log\left(\frac{1}{\delta}\right)/\delta^2$, and the global decoding error is bounded above by $2^{-\delta^2 t/24}$. ♢

*Proof of Lemma 3.* Part 1: Let $D_1$ be the distribution on $\{0,1\}^b$ obtained by flipping $b$ coins independently with probability $p$ of outcome 1. Let $D_2$ be the distribution on $\{0,1\}^b$ obtained by randomly selecting a substring of exactly $b$ bits from a string of $n$ bits containing $pn$ ones. We can think of $D_1$ as sampling $b$ positions *with replacement* from the same string as $D_2$; the distributions are equal conditioned on $D_1$ never having a collision (i.e. sampling the same position twice). A collision occurs with probability at most $b^2/n$, which tends to 0 since $b = O(\log n)$. Let $S \subseteq \{0,1\}^b$ be the set of errors that are incorrectly decoded by a particular linear code.

$$\Pr(D_2 \in S) = \Pr(D_1 \in S \mid \text{no collision}) \leq \frac{\Pr(D_1 \in S)}{\Pr(\text{no collision})} \leq \frac{\Pr(D_1 \in S)}{1 - o(1)}$$

We are in fact concerned with the performance of the code on a particular subset of $b$ positions from the random vector $E$. By definition, these have statistical difference at most $2^{-t}$ from $D_2$. Since $t > b$, this is much smaller than the probability of a decoding error under $D_1$ or $D_2$ (which are both $2^{-\Omega(\delta^2 b)}$). Overall, moving from independent errors to errors from $E$ costs a factor of at most 2 in the probability of a decoding mistake.

Part 2: Consider any $t/b$ blocks. These involve $t$ bits from $E$, which are close to an "ideal" distribution — a sample of exactly $t$ positions from a string of $n$ bits with $pn$ ones. Consider what happens under this ideal distribution when we decode the inner code using the maximum likelihood algorithm, which outputs the codeword closest to the received word. The probability of correct decoding then decreases monotonically with the number of bits flipped in a particular block. When we condition on the event that a decoding mistake occurs in block 1, the probability of a decoding mistake then goes down in all other blocks, since fewer bits are likely to be flipped there. Similarly, conditioning on decoding mistakes happening simultaneously in any $i$ blocks makes a decoding mistake of any other block less likely. The probability that $t/b$ blocks simultaneously make mistakes is thus at most $\mu^{t/b}$ under the ideal distribution (recall $\mu$ is the mistake probability in each block). If we consider now what happens under the distribution $E$, the probability of simultaneous decoding mistakes is at most $\mu^{t/b} + 2^{-t}$ (by the Definition 4). Finally, since $\mu > 2^{-b}$, we can bound the mistake probability by $2\mu^{t/b}$, as desired. □

# 3 Communication-Optimal Information Reconciliation

## 3.1 Optimal Reconciliation Protocols Using Previous Work

As mentioned in the introduction, several communication-optimal reconciliation protocols can be derived directly using ideas from the literature. None of the protocols appears explicitly in the

form that interests us and so, in this section, we translate the relevant work to our setting. Table 1 states the parameters and basic properties of various protocols. As a point of comparison we also include a protocol based on the syndrome of a standard linear code lying on the Gilbert-Varshamov bound (this corresponds to the best existential results on such codes).

Reducing Random to Worst-Case Errors Assume, for a moment, that the differences between $w$ and $w'$ are somehow guaranteed to be random and independent. We could then match the $nh(p)$ lower bound efficiently by using the syndrome $\mathsf{syn}_C$ of a polynomial-time decodable code which achieves the Shannon capacity on a binary symmetric channel (e.g. the concatenated code of Section 2.1). The code has dimension $n(1-h(p)-o(1))$, and so the sketch is only $n(h(p)+o(1))$ bits long.

Of course, we want to avoid such a strong assumption. In all of the settings where sketches are used, it makes sense to minimize the requirements on the exact stochastic properties of the noise process being dealt with. For example, it is easy to estimate the number of errors introduced by some particular noise process, but it is difficult, in general, to estimate the precise distribution on the errors, or even to test if it satisfies specific "target" properties such as independence. Consider, for example, biometric data. Their binary representations are typically the result of several transformations of the original non-binary data. Close inputs (say, iris scans) may well correspond to close binary strings, but there is no reason to suppose errors in different parts of the string representation would be independent.

A natural way around this is for Alice to choose a random permutation $\pi : [n] \to [n]$ (here $[n] = \{1, \ldots, n\}$) and use it to permute the bits of $w$ (Bennett *et al.* [BBR88, p. 216]). She sends the pair $\pi, \mathsf{syn}_C(\pi(w))$ to Bob, who computes $\mathsf{syn}_C(\pi(w'))$ and XORs this with the received string to obtain $\mathsf{syn}_C(\pi(w \oplus w'))$. Now the distribution on $\pi(w \oplus w')$ depends only on the distance between $w$ and $w'$. If it is at most $pn$, then running the syndrome decoding algorithm on $syn_C(\pi(w) \oplus \pi(w'))$ will recover the error vector $w \oplus w'$ with high probability.[7]

The problem is that this scheme requires a lot of communication since the description of $\pi$ is $n \log n$ bits. The next natural step is to use a cryptographic pseudo-random generator to choose $\pi$, and send only the seed of the generator instead of an explicit description of $\pi$ (this idea is implicit in, for example, [DGL04]). This reduces the total communication to only $n(h(p)+o(1))$, but requires an unproven hardness assumption.

How, then, can we get around this? A first observation is that this solution does *not* require the adversary introducing the errors to be computationally bounded; instead, it relies on the fact that the decoding algorithm for the code is polynomial time. The adversary's strategy is limited to the choice of $w$ and $w'$ and so she can always be described by a circuit of size $2n$. The class of tests which the generator must fool is given by applying the syndrome decoder for $C$ to $\mathsf{syn}_C(\pi(w \oplus w'))$ and checking if the result is indeed $w \oplus w'$. Since we have some control over the choice of the code $C$, the problem begins to look more like a classical, algorithmic de-randomization question: how many random bits are necessary to fool the decoder for $C$ on all inputs? Protocol 1 and our analysis of concatenated codes show that $o(n)$ bits are sufficient. Removing the assumption of a

[7] We are in fact using the code to correct errors which are not independent, but rather chosen randomly conditioned on exactly a certain number of errors occurring. It is not difficult to see that concatenated codes correct such errors with high probability, e.g. by tracing through the argument of Section 2.1. More generally, say a decoder is *monotone* if for every error pattern $e$ which it correctly repairs, it also correctly repairs all error patterns which are subsets of $e$. (Since we deal with linear codes it is sufficient to talk about the error pattern, not the actual corrupted word). Then restricting oneself to uniformly random errors of weight $\tau \leq pn$ can increase the mistake probability of the code by at most a factor $O(\sqrt{n})$ since (1) the worst case occurs with exactly $pn$ errors and (2) such strings occupy a $\Omega(1/\sqrt{n})$ fraction of the binary symmetric distribution. The hard decoder for concatenated codes may not always be monotone, but the fraction of errors on which it behaves monotonically is high enough for the same argument to work.

| Scheme | Entropy Loss | Length $\ell$ | Decoding error $\log(1/\epsilon)$ | Efficient/ explicit? | Notes |
|---|---|---|---|---|---|
| Syndrome of code on GV bound | $nh(2p)$ | = ent. loss | $\epsilon = 0$ | no/no | Efficient codes do worse |
| List-dec. + hash [Gur03, DRS04] | $nh(p)+1$ | ent. loss + $\delta n$ | $\frac{1}{2}\delta n - \log n - 2$ | no/no | $\delta > \frac{\log n}{n}$ |
| Random perm'n [BBR88] | $n(h(p)+\delta)$ | $n(\log n + 1)$ | $\Omega(\delta^2 n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ |
| Perm'n via p.r.g. [DGL04] | $n(h(p)+\delta)$ | ent. loss + $n^\alpha$ ($\alpha$ constant) | $\omega(\log n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ and **cryptographic** p.r.g. |
| $t$-wise ind. perm'n (this work) | $n(h(p)+\delta)$ | ent. loss + $\delta n$ | $\Omega(\delta^3 n/\log n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ |

Table 1: Sketches correcting $pn$ out of $n$ adversarial binary Hamming errors. The term "explicit" means that a description of the protocol can be computed in polynomial time from a unary description of $n$ and binary representations of the other parameters. The parameters described here assume $p$ is a constant bounded away from 0 and $\frac{1}{2}$. For $p = o(1)$, standard codes in fact do quite well. For $p = \frac{1}{2} - o(1)$, efficient, explicit near-optimal list-decodable codes are known [GS00].

cryptographic generator additionally allows us to give an exact expression for the probability of a decoding error.

AN INEFFICIENT SOLUTION VIA LIST-DECODING A different approach to dealing with adversarial errors is for Alice to compute and send the syndrome of $w$ with respect to a linear, *list-decodable* code. A code $LDC : \{0,1\}^k \to \{0,1\}^n$ is $(L, pn)$-list decodable if for every word $y \in \{0,1\}^n$, there are at most $L$ codewords at distance $pn$ or less from $y$.

If Bob receives $\mathsf{syn}_{LDC}(w)$, he can use it to compute a list of at most $L$ candidates for $w$ which are within distance $pn$ of $w'$. If Alice appends a small hash of $w$ to her message then Bob can use it to select which of the candidates is the correct one.

Linear codes with polynomial with list size $L = n$ are known to *exist* with dimension $n(1-h(p)) - 1$ [GHSZ02]. For Bob to select the correct $w$ from the list, it is sufficient to append an almost pairwise-independent hash of $w$ whose output length is about $\log L$. The description of the hash function also takes about $\log L$ bits. The scheme looks like:

$$S(w) = \underbrace{\text{description of } hash,\ hash(x)}_{o(n) \text{ bits}}, \underbrace{\mathsf{syn}_{LDC}(w)}_{nh(p)+1 \text{ bits}}.$$

The idea of sieving a list in this way was perhaps always implicit in work on list-decodable codes. It was used for coding with a "side channel" [Gur03], private codes [Lan04, MPSW05] and mentioned in the context of reconciliation in [DRS04]. It is also used in complexity-theory, where the hash is often provided as advice to a small circuit.

The main problem with this approach is efficiency. Except for extreme ranges of the error rate ($p \approx 0$ or $p \approx \frac{1}{2}$), polynomial time constructions of capacity-approaching list-decodable binary codes are not known, never mind efficient decoding algorithms for them (see [GS00] for a discussion). Nonetheless, the parameters of this construction are better than those obtained via randomization; they are included for comparison in Table 1.

### 3.2 Analysis of Protocol 1

ALMOST $t$-WISE INDEPENDENT PERMUTATIONS A family of permutations $\pi : [n] \to [n]$ is $\beta$-almost $t$-wise independent if for every $t$ elements $i_1, \ldots, i_t$ in $[n]$, the vector of random variables $(\pi(i_1), \ldots, \pi(i_t))$ has statistical difference at most $\beta$ from the distribution it would have if $\pi$ were

a uniformly random permutation, i.e. from a sample of $t$ points in $[n]$ chosen uniformly *without* replacement. Kaplan, Naor and Reingold [KNR05] give a polynomial time construction of such families from which one can sample using $O(t \log n + \log(1/\beta))$ random bits.

Analysis Suppose Alice runs Protocol 1, setting $\beta = 2^{-t}$ when she uses the KNR generator, and uses the code $C$ from Theorem 1. The description of $\pi$ requires $O(\delta n)$ bits. We obtain:

**Theorem 4.** *For $\delta = \delta(n) = \Omega(\frac{\log\log n}{\sqrt{\log n}})$, there exists an explicit, polynomial-time computable and decodable family of sketches tolerating $pn$ errors with length $\ell = n(h(p) + \delta)$ and error $\epsilon = 2^{-\Omega(\delta^3 n/\log n)}$.*

Recall, as a point of comparison, that any sketch tolerating $pn$ errors must have length $\ell$ at least $n(h(p) - o(1))$, since one can use it to transmit any vector of weight $pn$, and there are $2^{-n(h(p)-o(1))}$ such vectors when $p$ is constant.

*Proof.* Fix particular inputs $w, w'$. It suffices to show that the distribution $\pi^{-1}(w \oplus w')$ is $t$-wise independent, in the sense of Definition 4, for $t = \delta n/\log n$. Consider any $t$ locations $i_1, \ldots, i_t$ in $\pi^{-1}(w \oplus w')$. Their values are given by the bits at positions $\pi(i_1), \ldots, \pi(i_t)$ in $w \oplus w'$. This latter string has at most $pn$ ones. If $\pi$ were a truly random permutation, this would be a uniform sample of $t$ positions without replacement. Since $\pi$ is $2^{-t}$-almost $t$-wise independent, the restriction of $w \oplus w'$ is $2^{-t}$ close to what it would be if $\pi$ were random, as required.

By Theorem 1, the decoding algorithm for $C$ recovers the string $\pi^{-1}(w \oplus w')$ with probability at least $1 - \epsilon$ where $\epsilon = 2^{-\Omega(\delta^2 t)} = 2^{-\Omega(\delta^3 n/\log n)}$. □

# 4 Improved Efficient Private Codes

Protocol 2 yields the following:

**Theorem 5.** *For any $\delta = \delta(n) = \Omega(\log\log n/\sqrt{\log n})$, there exists an* explicit, efficient *family of $[n, Rn, \ell_{key}]$ private codes correcting $pn$ adversarial errors with probability $1 - \epsilon$, where $R = 1 - h(p) - \delta$, $\ell_{key} \leq n(1 + \delta)$ and $\epsilon = 2^{-\Omega(\delta^3 n/\log n)}$.*

*Proof.* Because of the mask $a$, the adversary obtains no information on the string $\pi_s^{-1}((C(x))$, and so the set of positions corrupted in the channel is independent of $\pi_s$. Let $e$ be the error vector introduced by the adversary—we may think of it as fixed. As in the proof of Theorem 4, the code $C$ is confronted with the error distribution $\pi(e)$ where $\pi$ comes from the KNR generator. This distribution satisfies Definition 4. Applying Theorem 1 yields the desired bound. □

Table 2 compares the properties of the new construction with schemes from previous work.

# References


[BBCM95] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BBR88] C. Bennett, G. Brassard, and J. Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[BDK+05] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer-Verlag, 2005.

| Scheme | Rate $k/n$ | Key length $\ell_{key}$ | Decoding err. $\log(1/\epsilon)$ | Efficient/ explicit? | Notes |
|---|---|---|---|---|---|
| Standard Code on GV bound | $1-h(2p)$ | 0 | $\epsilon = 0$ | no/no | Efficient codes do worse |
| [Lan04] | $1-h(p)-\delta$ | $2\delta n$ | $\delta n - 2\log n - 2$ | no/no | $\delta > \frac{2\log n}{n}$ |
| [DGL04] | $1-h(p)-\delta$ | $n(\log n + 1)$ | $\Omega(\delta^2 n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ |
| [DGL04] | $1-h(p)-\delta$ | $n^\alpha$ ($\alpha$ constant) | $\omega(\log n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ and **cryptographic** p.r.g. |
| This work | $1-h(p)-\delta$ | $(1+\delta)n$ | $\Omega(\delta^3 n/\log n)$ | yes/yes | $\delta = \Omega(\frac{\log\log n}{\sqrt{\log n}})$ |

Table 2: Private codes correcting $pn$ out of $n$ adversarial binary Hamming errors. The parameters described here assume $p$ is a constant bounded away from 0 and $\frac{1}{2}$. See notes for Table 1.


[BS92] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992.

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[CPSV00] Graham Cormode, Mike Paterson, Süleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *SODA*, pages 197–206, 2000.

[CT91] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications, 1991.

[DGL04] Yan Zhong Ding, P. Gopalan, and Richard J. Lipton. Error correction against computationally bounded adversaries. Manuscript. Appeared initially as [Lip94], 2004.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *TCC 2004*, pages 446–472, 2004.

[Din05] Yan Zong Ding. Error correction in the bounded storage model. In Kilian [Kil05].

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Cachin and Camenisch [CC04].

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 654–663. ACM, 2005.

[Eli91] Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

[For66] G. David Forney. *Concatenated Codes*. PhD thesis, MIT, 1966.

[GHSZ02] Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1034, 2002.

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 181–190, Portland, Oregon, 21–23 May 2000.

[Gur03] Venkatesan Guruswami. List decoding with side information. In *IEEE Conference on Computational Complexity*, pages 300–. IEEE Computer Society, 2003.

[JS02] Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *IEEE International Symposium on Information Theory*, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Sixth ACM Conference on Computer and Communication Security*, pages 28–36. ACM, November 1999.

[Kil05] Joe Kilian, editor. *First Theory of Cryptography Conference — TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*. Springer-Verlag, February 10–12 2005.

[KNR05] Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k-wise (almost) independent permutations. In *APPROX-RANDOM*, pages 354–365, 2005.

[Lan04] Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 325–334, Washington, DC, USA, 2004. IEEE Computer Society.

[Lip94] Richard J. Lipton. A new approach to information theory. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS*, volume 775 of *Lecture Notes in Computer Science*, pages 699–708. Springer, 1994. The full version of this paper is in preparation [DGL04].

[MPSW05] Silvio Micali, Chris Peikert, Madhu Sudan, and David Wilson. Optimal error correction against computationally bounded noise. In Kilian [Kil05].

[MT02] Yaron Minsky and Ari Trachtenberg. Scalable set reconciliation. In *40th Annual Allerton Conference on Communication, Control and Computing, Monticello, IL*, October 2002. See also tehcnial report BU-ECE-2002-01.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[Orl90] Alon Orlitsky. Worst-case interactive communication i: Two messages are almost optimal. *IEEE Transactions on Information Theory*, 36(5):1111–1126, 1990.

[Orl92] Alon Orlitsky. Average-case interactive communication. *IEEE Transactions on Information Theory*, 38(5):1534–1547, 1992.

[Orl93] Alon Orlitsky. Interactive communication of balanced distributions and of correlated files. *SIAM J. Discrete Math.*, 6(4):548–564, 1993.

[RW04] Renato Renner and Stefan Wolf. The exact price for unconditionally secure asymmetric cryptography. In Cachin and Camenisch [CC04], pages 109–125.

[RW05] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In Bimal Roy, editor, *Advances in Cryptology—ASIACRYPT 2005*, Lecture Notes in Computer Science, Chennai, India, 4–8 December 2005. Springer-Verlag.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[Sti93] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer, 1993.

[vL92] J.H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1992.

[ZP81] Victor V. Zyablov and Mark S. Pinsker. List cascade decoding. *Problems of Information Transmission*, 17(4):29–34, 1981. (In Russian); pp. 236-240 (in English), 1982.

# A A Simple Construction of Private Codes

We present a simple proof of Langberg's result [Lan04] that private codes with short keys exist for decoding up to the Shannon capacity. We also observe along the way that one can ensure that the final code is linear for every particular value of the private key.

**Definition 5.** *A code $LDC : \{0,1\}^k \to \{0,1\}^n$ is $(L, pn)$-list decodable if for every word $y \in \{0,1\}^n$, there are at most $L$ codewords at distance $pn$ or less from $y$.*

**Definition 6.** *A one-time message authentication code (MAC) with forgery probability $\gamma$ and message length $m$ is a function $mac : \{0,1\}^m \times \{0,1\}^{\ell_{key}} \to \{0,1\}^{\ell_{tag}}$ such that for any computationally unbounded adversary $A$, and for all messages $x \in \{0,1\}^m$, the probability over the key $r \leftarrow \{0,1\}^{\ell_{key}}$ and the adversary's coins that $A(x, mac(x;r)) = x', mac(x';r)$, and $x' \neq x$, is at most $\gamma$.*

Given a list decodable code $LDC$ and a MAC, a natural private code is to use the shared randomness as the MAC key, and encode the resulting pair $(message, tag)$, i,e:

$$PC(x; r) = LDC(x, mac(x; r)).$$

This reduces the message length slightly, but allows the receiver to uniquely identify the original message.

**Lemma 6.** *Let $LDC : \{0,1\}^k \to \{0,1\}^n$ be a $(L, pn)$ list decodable code. Let mac be an authentication code with forgery probability $\epsilon/L$ and message length $k - \ell_{tag}$. Then there is a $[n, k - \ell_{tag}, \ell_{key}]$ private code which corrects any $pn$ errors with probability $1 - \epsilon$.*

*Proof.* The encoder, given key $r$ and message $x$, sends $LDC(x, mac(x; r))$. The decoder computes a list of at most $L$ candidate messages of the form $(x', tag')$ (this is efficient iff the code has an efficient list-decoding algorithm); it outputs a message $x$ if and only if it is the only message in the list with a valid tag. On one hand, the correct message and tag will always be in the list. On the other hand, no matter which $pn$ errors the adversary introduces, she gets at most $L - 1$ chances to forge a tag on an invalid message. By a union bound, she succeeds with probability at most $(L - 1)\gamma < \epsilon$. □

**Fact 7 (Folklore).** *For any message length $m$ and error $\gamma > 0$, there exists a MAC with keys of length $\ell_{key} \leq 2(\log n + \log\left(\frac{1}{\gamma}\right) + 1)$ and tags of length $\ell_{tag} \leq \log n + \log\left(\frac{1}{\gamma}\right) + 1$. For every fixed key $r$, the function $mac(\cdot; r)$ is linear.*

**Fact 8 ([ZP81, Eli91, GHSZ02]).** *For any $p \in (0, 1)$ and integers $n, L$, there exists a $(L, pn)$ code with dimension $k \geq n(1 - h(p) - 1/L)$. Moreover, the code can be chosen to be linear.*

Taking $L = n$ in the lemma above, we get:

**Corollary 9 ([Lan04]).** *For any $p, \delta \in (0, 1)$ such that $\delta < 1 - h(p)$, and for every integer $n$, there exists a $[n,\ n(1 - h(p) - \delta),\ 2\delta n]$ private code which corrects $pn$ errors with probability $1 - \epsilon$ where $\epsilon \leq n^2 2^{-\delta n+1}$. Moreover, for every fixed key the resulting code is linear.*

**Notes.** The MAC from the fact above is standard: Let $s = \lceil \log(m/\gamma) \rceil$. Take the bits of the message $x$ to give the coefficients of a polynomial $x(z)$ of degree $m$ over the field $GF(2^s)$ (and set the constant term to 0). The key consists of two field elements $a, b$ and the tag is $x(a) + b$. The adversary can only forge if he can find a pair $x'$ and $x'(a) - x(a)$. Now $b$ masks all information about $a$, and so we may think of $a$ as chosen after the adversary outputs his guess. The polynomial $(x' - x)$ takes on any particular value at most $m$ times since it has degree at least 1 and at most $m$. The adversary's success probability is thus at most $m/2^s \leq \gamma$.

The construction of (non-linear) list-decodable codes that achieve the Shannon capacity proceeds by choosing a random code and throwing out sets of codewords that are too close [ZP81, Eli91]. Ensuring that the code is also linear is more complicated; the result is due to Guruswami *et al.* [GHSZ02].