Optimal Error Correction Against Computationally Bounded Noise

Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson

MIT CSAIL, 77 Massachusetts Ave, Building 32, Cambridge, MA, 02139.
{silvio,cpeikert,madhu,dwilson}@mit.edu

Abstract. For computationally bounded adversarial models of error, we construct appealingly simple, efficient, cryptographic encoding and unique decoding schemes whose error-correction capability is much greater than classically possible. In particular:

1. *For binary alphabets, we construct positive-rate coding schemes which are uniquely decodable from a $1/2 - \gamma$ error rate for any constant $\gamma > 0$.*
2. *For large alphabets, we construct coding schemes which are uniquely decodable from a $1 - \sqrt{R}$ error rate for any information rate $R > 0$.*

Our results are qualitatively stronger than related work: the construction works in the public-key model (requiring no shared secret key or joint local state) and allows the channel to know everything that the receiver knows. In addition, our techniques can potentially be used to construct coding schemes that have information rates approaching the Shannon limit. Finally, our construction is qualitatively optimal: we show that unique decoding under high error rates is *impossible* in several natural relaxations of our model.

1 Introduction

The theory of error correction is concerned with sending information reliably over a "noisy channel" that introduces errors into the transmitted data. In this setting, a *sender* starts with some *message*, which is a fixed-length string of symbols over some alphabet. The sender *encodes* the message into a longer string over the same alphabet, then transmits the block of data over a *channel*. The channel introduces *errors* (or *noise*) by changing some of the symbols of the transmitted block, then delivers the corrupted block to the *recipient*. Finally, the recipient *decodes* the block (hopefully to the intended message). Whenever the sender wants to transmit a new message, the process is repeated.

Two quantities are of special interest in this setting: the *information rate* (i.e., the ratio of the message length to the encoded block length) and the *error rate* (i.e., the ratio of the number of errors to the block length).

Coding schemes having high information rate and tolerating high error rate are, of course, the most desirable. Small alphabets are desirable too, and in particular most natural channels are indeed best at transmitting only bits.

But the question remains: *how should we model a noisy channel?*

Standard channels. There are two historically popular ways to model a noisy channel. Shannon's *symmetric channel* independently changes each symbol to a random different one, with some fixed probability. Hamming's *adversarial channel* changes symbols in a worst-case fashion, subject only to an upper bound on the number of errors per block of data. In particular — though this is not often stated explicitly — the adversarial channel is computationally *unbounded.* Working with this "pessimistic" model certainly ensures the robustness of the resulting coding scheme, but it also severely restricts the information and error rates. For instance, when the alphabet is binary, the error rate must be less than 1/4 for unique decoding to be possible (unless the blocks are exponentially longer than the messages).

One way to recover from a higher error rate is to relax the task of decoder, allowing it to output a short *list* of messages which contains the intended one. To tolerate adversarial channels with high error rates, list decoding seems to be the best one can do — but under a more *"realistic"* model of an adversarial channel, is it possible to *uniquely* decode under high error rates?

Computationally bounded channels. In 1994, Lipton [9] put forward the notion of a computationally bounded channel, which is essentially a Hamming channel restricted to *feasible* computation. That is, the channel still introduces errors adversarially (always subject to a given error rate), but must do so in time polynomial in the block length.

We posit that natural processes can be implemented by efficient computation, so *all* real-world channels are, in fact, computationally bounded. We therefore have confidence that results in this model will be as meaningful and applicable as classical codes. Indeed, the nature of the model is such that if some malicious (or natural!) process is capable of causing incorrect decoding, then that process can be efficiently harnessed to break standard hardness assumptions. In contrast to coding schemes which are only guaranteed to work against channels that are modelled by *very specific, limited* probabilistic processes, results in this model apply *universally* to any channel which can be modelled by efficient computation.

Remarkably, under standard cryptographic assumptions and assuming that sender and receiver share secret randomness, Gopalan, Lipton, and Ding [3] proved that for such a bounded channel, it is possible to decode correctly from higher error rates. Unfortunately, their result requires the communicating parties to share a secret key which is unknown to the channel.

More significantly, though the bounded-channel model was first envisioned over a decade ago, nobody has yet shown an *essential* use of this assumption to yield any unique benefits over an unbounded channel. That is, previous constructions still work when the channel is computationally unbounded, as long as the sender and receiver share some secret randomness. The bounded-channel assumption is used to reduce the *amount* of shared randomness that is needed, but not to eliminate it altogether. This computational assumption is thus an *additional* one, and does not supplant the assumption of secret randomness shared between the sender and receiver.

Our goal is to provide a general method for optimal error correction, exploiting the bounded-channel assumption in an essential way.

1.1 Our Contributions

Our setting. We work in a very simple cryptographic setting: we assume that a one-way function exists (the "minimal" cryptographic assumption) and that the sender has a public key known to the receiver (and, perhaps, to the channel as well).

The sender (but *not* the receiver) keeps a small amount of state information, which he uses when encoding messages. Because the sender keeps state, our constructions are actually *dynamic* coding schemes, in which the same message results in a different encoding each time it is sent.

Our results. Our setting yields great benefits in error correction for both binary and large alphabets. Namely,

1. *For binary alphabets, we construct positive-rate dynamic coding schemes which are uniquely decodable from a $1/2-\gamma$ error rate for any constant $\gamma > 0$.*
   Classically, a $1/4-\gamma$ error rate is the best possible for unique decoding (and positive information rate). We stress that *in any reasonable model*, decoding of *any kind* (even list decoding) is impossible under an error rate of $1/2$. Therefore this result is optimal in a very strong

sense, and matches the best possible error rates in the weaker Shannon model.

2. *For large alphabets, we construct dynamic coding schemes which are uniquely decodable from a* $1-\sqrt{R}$ *error rate for any information rate* $R>0$.
   The $1-\sqrt{R}$ error rate is actually a consequence of known list decoding algorithms, and not imposed by our technique. Note that when $R<1/4$, we can uniquely decode from error rates much greater than $1/2$, which is impossible in the Hamming model.

To achieve these results, we actually prove a very general *reduction*, namely,

> If one-way functions exist, *(dynamic) unique decoding* from e errors in the bounded-channel model reduces to efficient *(static) list decoding* from e errors in the Hamming model (with no asymptotic loss in information rate).

We obtain results 1 and 2 above by applying this reduction to the classical Guruswami-Sudan [7] and Reed-Solomon codes.

Optimality of our model. There are three defining characteristics of our model: (1) the sender is stateful (the amount of state required is minimal; either a single counter value or a local clock would suffice) while the receiver is stateless, (2) the sender keeps a secret key which is unknown to the channel, and (3) the channel is assumed to be computationally bounded.

We show that our model is qualitatively optimal: relaxing any of these three requirements makes the task of unique decoding under high error rates *impossible*. Thus our construction can be seen as the "best possible" use of the bounded-channel assumption for error correction. See Section 4.3 for details.

Overview of the construction. Starting with any static code, we specify a *cryptographic sieving* procedure, which only certain "authentic" codewords will pass. Authentic words are hard for the adversary to compute (even after seeing other authentic codewords), but easy for the sender to generate and for the recipient to sieve out.

Upon receiving a corrupted word, the recipient first *list decodes* it. Of course, list decoding only provides a set of candidate codewords. In order to uniquely decode, the recipient next uses the cryptographic sieve to filter out only the authentic word(s). Provided that the number of errors is suitably limited, the intended codeword is guaranteed to appear in the decoded list and pass the sieve. However, it may not be alone: though the bounded channel cannot produce any *new* authentic codewords, it may be able to cause *prior* ones to appear in the decoded list. This is where the sender's state comes into play: dynamic encoding allows the receiver to choose the "freshest" word that passes the sieve, resulting (with overwhelming probability) in correct, unique decoding.

2 Related Work

We wish to contrast our results with several other models and techniques for tolerating high error rates.

2.1 List Decoding

One of the best-known methods of decoding beyond classical limits under adversarial error is known as *list decoding*. In list decoding, a received word is not decoded to a unique message, but rather to a short *list* of possible messages. If the number of errors is within the *list-decoding radius*, the original message will appear in the list.

There exist codes with rate approaching the Shannon capacity of the channel and yielding constant-size lists (cf. [5]); however, no efficient list decoding algorithms are known for such codes. Still, many popular codes have efficient list-decoding algorithms that can decode significantly beyond the half-the-distance bound.

The obvious drawback of list decoding is that one typically desires to know the *unique* message that was sent, rather than a list of possible messages. The works presented below, as well as our cryptographic sieve, use list decoding as a tool to extend the decoding radius, then employ additional assumptions in order to identify the correct, unique message in the list.

2.2 List Decoding with Side Information

Guruswami [4] achieves unique decoding under high error rates for binary alphabets. However, he makes two strong assumptions: first, the communicating parties must share a *side channel* which is *noise-free*, and must use it every time a message is sent. (Note that the side channel does not trivialize the problem, because it is only used to send strings that are much shorter than the messages.) Second, the adversary must not know what is sent over the side channel when introducing errors in the main channel; this imposes either a privacy or timing constraint on the side-channel communication.

2.3 Code Scrambling using Shared Randomness

The code-scrambling method of Gopalan, Lipton, and Ding [3] assumes that the communicating parties share some secret random (or pseudorandom) data. The randomness is used to "scramble" codewords, which reduces adversarial noise to (random) Shannon noise. Under such a random-error model, and for certain properly-chosen codes, maximum-likelihood decoding yields the correct word with high probability.

Code scrambling and our cryptographic sieve are both based on the minimal cryptographic assumption of the existence of one-way functions.[1] But our underlying model compares favorably to that of code scrambling in some important ways:

Cryptographic setup. The code-scrambling method requires a random secret key to be shared between the communicating parties and kept secret from the channel. Such a setup requires either the parties to meet in advance (which may be unrealistic), or some *interactive* protocol to establish the private key (in addition, such a protocol would have to deal with the noisy channel that separates the two parties!).

In contrast, our cryptographic sieve works in the *public key* setting: we only require the sender to have a single public key that is known to the recipient. In fact, our results hold even when the channel possesses all the information available to the receiver, and is potentially even more computationally powerful than the receiver. Previous results certainly do not allow the channel to be this powerful.

Local state. In reality, two communicating parties usually send and receive many messages over time. Using a classical (static) code, this is no problem: each message is simply encoded and decoded on its own, with no implications for correctness. However, when shared randomness and state are introduced, one must be more careful.

The first observation is that in the code-scrambling method, the shared key (or portion thereof) must only be used *one time*. If any part is re-used,

[1] The two employ different cryptographic primitives, both of which are implied by one-way functions. Gopalan *et al* use a pseudorandom generator, while our solution uses an existentially unforgeable signature scheme [2, 10].

the adversary gains information about how the codewords are scrambled, and may be able to introduce "non-random" errors in the future. Therefore, the code-scrambling method requires both parties to keep *synchronized state*. That is, they must always agree on what fresh portion of their shared (pseudo)random key should be used for scrambling and unscrambling the next message. If the parties fall out of sync (say, due to an unexpected transmission or hardware failure), then future messages will decode incorrectly.[2]

In contrast, our cryptographic sieve only requires the sender to maintain a small amount of local state, independent of the recipient (who is stateless). If a message is dropped or ignored, there is no effect on the correctness of future messages. [3]

We also compare our quantitative results with those of Gopalan *et al*:

Binary alphabets. For binary alphabets, the code-scrambling method can yield coding schemes that handle the optimal error rate of $\epsilon = 1/2 - \gamma$ for any $\gamma > 0$. In addition, the information rate is optimal, because it meets the Shannon limit of $1 - H(\epsilon)$.

Our method also provides unique decoding from a $1/2 - \gamma$ error rate. We stress that while our technique yields positive asymptotic information rate, it does not yet match the Shannon limit, because the information rate is dictated by the underlying efficiently list-decodable code. While list-decodable codes matching the Shannon limit for any error rate are known to exist, it is not known how to efficiently list decode them. Fortunately, improvements in list decoding techniques automatically carry over to our construction.

Large alphabets. Implemented with Reed-Solomon codes (which require large alphabets), code scrambling allows unique decoding from a $\min(1 - \sqrt{R}, 1 - 2R)$ error rate (where R is the information rate), while classical unique decoding of RS codes only allows a $(1-R)/2$ error rate. Therefore,

[2] To relax the synchronization requirements, one might imagine sending some "synchronizing information" along with each message. However, the synchronizing information is also subject to errors, so it must be protected by some encoding, and also be recoverable separately from the message. (Using a pseudorandom function for synchronization suffers from a similar problem.) Eliminating the synchrony requirement, while retaining desirable information and error rates, seems to be quite difficult.

[3] Of course, we cannot provide such a guarantee if the channel is allowed to arbitrarily *delay* messages and *swap* their order — however, neither can any scheme that uses synchronized state.

for $R \geq 1/3$ the code-scrambling method offers no advantage over classical decoding; for $R \in (1/4, 1/3)$ there are some benefits but they are not as pronounced as in the low-rate case.

In contrast, our method meets or exceeds the asymptotic error correction rate of the code-scrambling method, at all information rates. In particular, it allows unique decoding from a $1 - \sqrt{R}$ error rate, for all values of R.

Universality. Because it relies on the randomness of the errors, the analysis of the code-scrambling method depends essentially upon the properties of Reed-Solomon codes.[4] The authors also point out that a similar analysis can be applied to certain algebraic-geometry codes, and that experimental simulations using expander codes have shown positive results. However, each application of code scrambling to a new family of codes requires a new analysis (and yields, potentially, a new set of workable parameters).

Our construction, instead, is fully general: it uses an efficient list-decoding algorithm as a black-box, and requires no other special properties of the code. It reduces *unique decoding* against a *bounded* adversary to *list decoding* against an *unbounded* adversary, and retains all the asymptotic parameters of the code.

2.4 Private Codes

In a recent independent work, Langberg [8] describes "private codes" in which the sender and recipient use a shared secret random key (which is not known to the channel) to uniquely decode under high error rates.

Langberg assumes a computationally unbounded channel and focuses mainly on existential (rather than efficiently constructible) results, and on tight bounds for the length of the secret key. The construction uses certain combinatorial set systems to define a "secret subcode" C_r of a given static code C, based on the secret key r. Unique decoding is performed by maximum-likelihood decoding within C_r. The analysis and security proof of the scheme are somewhat complex and difficult to penetrate.

Compared to our cryptographic sieving, private codes share two main drawbacks with code scrambling. Namely,

---

[4] For Lemma 3.2 in Gopalan *et al*, the *maximum distance separability* (MDS) property of Reed-Solomon codes is the key ingredient. The MDS property is true of RS codes but not true in general.

1. *They require secret randomness to be shared between the sender and receiver and kept secret from the channel.* By contrast, in our model the channel is on "equal footing" with the receiver: it knows everything that is known to the latter.
2. *They require sender and receiver to keep synchronized state.* (Else they may not be able to understand each other whenever multiple messages are sent.) No such requirement exist in our model, and multiple messages can be safely sent.

Finally, in our view, we contribute a conceptually cleaner framework and simpler security proof. In retrospect, it is possible to cast private codes as a *specific construction* in our general framework of *message authentication* and *cryptographic sieving*.[5] (We thank Adam Smith for pointing out this relationship.)

3 Preliminaries

3.1 Notation

Messages and words, which are just vectors over some alphabet, are written in bold: e.g., $\mathbf{m}, \mathbf{x}, \mathbf{r}$. The concatenation of two vectors $\mathbf{x}, \mathbf{y}$ is denoted $\mathbf{x} \circ \mathbf{y}$. The set $\{1, \ldots, n\}$ is denoted by $[n]$. A negligible function in n is one which vanishes faster than $1/p(n)$ for any fixed polynomial p, and is denoted by $\nu(n)$.

3.2 Relevant Coding Theory

Basic concepts. The *message* is the information to be sent before it is encoded; it is a vector of k symbols from some finite *alphabet* Σ (by convention, we define $q = |\Sigma|$; a binary alphabet is one where $q = 2$). The message is encoded as a *codeword* of n symbols from Σ (n is called the *block length*). The *information rate* R of the code is defined as $R = k/n$; this is a measure of how much meaningful information is carried by each transmitted symbol.

[5] One can interpret private codes as using an (information-theoretically secure) secret-key *message authentication code* (MAC), rather than a (computationally secure) digital signature scheme. In this interpretation, the "secret subcode" C_r consists of encoded message-tag pairs, where the tag is a valid MAC of the message under secret key r. Maximum-likelihood decoding within C_r can be accomplished by first list decoding within C, then sieving out the decoded message-tag pairs that are authentic relative to r.

After passing through the channel, a (potentially corrupted) word is received and decoded, ideally back to the intended k-symbol message. However, in order for this to be the case, the number of errors must be suitably limited. The *Hamming distance* $\Delta(\mathbf{x}, \mathbf{y})$ between two words $\mathbf{x}$ and $\mathbf{y}$ is the number of symbols that differ between the two; we wish to decode the received word to the message whose codeword is nearest in Hamming distance.

Definition 1 (Hamming distance). *For any* $\mathbf{x}, \mathbf{y} \in \Sigma^n$*, the* Hamming distance between $\mathbf{x}$ and $\mathbf{y}$*, denoted* $\Delta(\mathbf{x}, \mathbf{y})$*, is the number of positions* i *in which* x_i *and* y_i *differ:* $\Delta(\mathbf{x}, \mathbf{y}) = |\{i \in [n] \ : \ x_i \neq y_i\}|$.

Definition 2 (Coding scheme, rate). *An* $(n, k)_q$-coding scheme $C = (E, D)$ *over alphabet* Σ *is an* encoding function $E : \Sigma^k \to \Sigma^n$ *and a* decoding function $D : \Sigma^n \to \Sigma^k$ *for some positive integers* $n \geq k$*,* $q = |\Sigma| \geq 2$*. The* (relative) rate *or* information rate *of the scheme, denoted* R*, is defined as* $R = k/n$*. The scheme tolerates* error rate ρ *if, for all* $\mathbf{m} \in \Sigma^k$ *and all* $\mathbf{r}$ *such that* $\Delta(E(\mathbf{m}), \mathbf{r}) \leq \rho n$*,* $D(\mathbf{r}) = \mathbf{m}$.

List decoding. Even if the actual error rate exceeds the rate ρ tolerated by a coding scheme, in some contexts it may be sufficient to decode to a short *list* of messages containing the intended one. *List decoding* finds such a list of all messages which encode to within some distance ϵn of the received word, where ϵ may be significantly greater than ρ.

Definition 3 (List decodability). *An* $(n, k)_q$ *coding scheme* $C = (E, D)$ *over* Σ *is* $(\epsilon n, L)$-list decodable *if, for any* $\mathbf{r} \in \Sigma^n$*, there exist* $\ell \leq L$ *distinct messages* $\mathbf{m}_1, \ldots, \mathbf{m}_\ell \in \Sigma^k$ *such that* $\Delta(\mathbf{r}, E(\mathbf{m}_j)) \leq \epsilon n$ *for all* $j \in [\ell]$.

Asymptotics. In order to make meaningful asymptotic statements about the rate of a coding scheme or the efficiency of encoding and (list) decoding, we must consider infinite *families* having increasing block lengths.

Definition 4 (Family of coding schemes). *An infinite* family $\mathbb{C}$ *of coding schemes is a set* $\mathbb{C} = \{C_i\}_{i=1}^{\infty}$ *where* $C_i = (E_i, D_i)$ *is an* $(n_i, k_i)_{q_i}$ *coding scheme, and* $\lim_{i\to\infty} n_i = \infty$.

The (asymptotic) information rate *(often just abbreviated* rate*) of* $\mathbb{C}$*, denoted* $R(\mathbb{C})$*, is defined to be* $R(\mathbb{C}) = \liminf_{i\to\infty} k_i/n_i$.

If C_i *is an* $(\epsilon_i n_i, L_i)$*-list decodable coding scheme, then we say that* $\mathbb{C}$ *is* list decodable *under error rate* $\epsilon(\mathbb{C}) = \liminf_{i\to\infty} \epsilon_i$.

If $\{E_i\}$ *and* $\{D_i\}$ *(respectively) can be computed by two uniform polynomial-time algorithms, we say that the coding scheme is* efficient.

Definition 5 (List decoding algorithm). *If* $\mathbb{C} = \{C_i\}$ *is a family of* $(n_i, k_i)_{q_i}$ *coding schemes* $C_i = (E_i, D_i)$ *over alphabet* Σ_i*, and each* C_i *is* $(\epsilon_i n_i, L_i)$*-list decodable, then an efficient* list decoding algorithm *for these parameters is a polynomial-time algorithm* LD *such that for all* i *and any* $\mathbf{r} \in \Sigma_i^{n_i}$, $LD(\mathbf{r})$ *outputs all* $\ell \leq L_i$ *messages* $\mathbf{m}_1, \ldots, \mathbf{m}_\ell \in \Sigma_i^{k_i}$ *such that* $\Delta(\mathbf{r}, E_i(\mathbf{m}_j)) \leq \epsilon_i n_i$ *for all* $j \in [\ell]$.

Note that LD must run in polynomial time in the size of its input, so in particular the list size L_i must be polynomially related to n_i. Many families are indeed efficiently list decodable for high error rates.

3.3 Relevant Cryptography

We require signature schemes which are existentially unforgeable under chosen message attack [2]. Such schemes were first shown to exist under a hardness of factoring assumption, and later under the assumption that one-way functions exist [10].

4 Dynamic Coding Schemes

4.1 The Formal Model

The issues of a bounded adversary, stateful players, and chosen-message attacks are not captured by the classical coding theory definitions. Here we formally define a bounded noisy channel and the requirements for the sender and receiver. For ease of notation, we provide definitions modeling the channel as a uniform algorithm; a non-uniform treatment is easily adapted.

A *dynamic coding scheme* for a family of parameters $\{(n_i, k_i, q_i)\}_{i=1}^{\infty}$ (with $\lim_{i\to\infty} n_i = \infty$) is a triple of probabilistic polynomial-time algorithms (G, S, R) such that:

- $G(1^{k_i}, 1^{n_i})$ outputs a pair (pk, sk);
- $S(\mathbf{m}, sk, aux)$, where sk was produced by $G(1^{k_i}, 1^{n_i})$, $\mathbf{m}$ is of length k_i over a q_i-ary alphabet Σ_i, and aux is some local state, outputs $(\mathbf{x}, aux')$ where $\mathbf{x} \in \Sigma_i^{n_i}$, and aux' is the updated local state that will be provided on the next invocation of S;
- $R(\mathbf{r}, pk)$, where pk was produced by $G(1^{k_i}, 1^{n_i})$ and $\mathbf{r} \in \Sigma_i^{n_i}$, outputs some $\mathbf{m}' \in \Sigma_i^{k_i}$.

The *information rate* of such a scheme is $\liminf_{i\to\infty} k_i/n_i$.

An (adversarial) *channel* $\mathcal{C}$ with *error rate* ϵ is a probabilistic poly-time algorithm which interacts with a sender S and receiver R in a chosen-message attack, which proceeds as follows:

1. $G(1^{k_i}, 1^{n_i})$ produces (pk, sk).
2. On input pk to $\mathcal{C}$, the following process is repeated until $\mathcal{C}$ terminates:
   - On the jth iteration, $\mathcal{C}$ chooses a message $\mathbf{m}_j \in \Sigma_i^{k_i}$ and hands it to the sender.
   - The sender encodes $\mathbf{m}_j$ using $S(\mathbf{m}_j, sk, aux_j)$, yielding aux_{j+1} and some $\mathbf{x}_j \in \Sigma_i^{n_i}$, which is given to $\mathcal{C}$.
   - $\mathcal{C}$ produces a word $\mathbf{r}_j$ such that $\Delta(\mathbf{x}_j, \mathbf{r}_j) \le \epsilon n_i$ with probability $1 - \nu(n_i)$, and hands $\mathbf{r}_j$ to the recipient.
   - The recipient runs $R(\mathbf{r}_j, pk)$ and outputs a message $\mathbf{m}'_j$.

We say that $\mathcal{C}$ *succeeds* at causing an incorrect decoding if, for any j in the above experiment, $\mathbf{m}'_j \neq \mathbf{m}_j$. We say that a dynamic code *uniquely decodes from error rate* ϵ if, for any channel $\mathcal{C}$ of error rate ϵ, $\Pr[\mathcal{C} \text{ succeeds}] \le \nu(n_i)$, where the probability is taken over the random choices of G, S, R, and $\mathcal{C}$.

Remark 1. In contrast to many cryptographic definitions of an adversary, our channel is *not* allowed to drop or re-order messages. That is, the channel must deliver a corrupted message before requesting a new message from the sender.

Against a more powerful channel which *can* drop and re-order messages, we are still able to construct coding schemes which provide similar guarantees about message integrity — provided that the receiver also keeps state. (However, the receiver's state is independent of the sender's.) We omit the details in this version of the paper.

4.2 The Construction

Intuition. The first attempt at a cryptographic sieve is to just sign each message and send the signature along with it. This obviously doesn't work because the signature is also subject to errors. The natural fix is to protect the signature in transit by also encoding it, using appropriate parameters. Unfortunately, a careful analysis (even using list decodable coding schemes for both the message and signature) shows that this approach yields no improvement in overall rate.

The key insight is that the message and signature should be encoded *together*, rather than separately. To communicate a message $\mathbf{m}$, the sender first signs **m**, then encodes the message-signature pair. To decode a received word **r**, the recipient applies the list decoding algorithm to **r**, yielding a list of potential message-signature pairs. If the number of errors is suitably bounded, then the original pair will appear in the list, and the signature will verify. And by the unforgeability of the signature scheme, the other pairs will not verify and can be thrown out. Therefore the original message can be recovered uniquely.

There is one hidden difficulty in the above description: the list may actually include several pairs that verify, but which correspond to messages sent *in the past*.[6] This event cannot be used to break the signature scheme, because a valid forgery must involve a *new*, unsigned message. Therefore we must find a way to disambiguate among potentially several valid message-signature pairs in the list.

The solution is for the sender to maintain a short *counter* t. The current value of the counter is appended to each message (and signed along with it), then incremented. Among all valid message-signature pairs in the decoded list, the recipient chooses the one with the largest counter. In other words, the recipient always decodes to the "freshest" message in the list. We note that the recipient is *stateless*, while the sender need only maintain the value of the counter.

Indeed, there are only two essential requirements for the counter values: they must not be reused, and the receiver must be able to recognize the most recent counter value in a list. Any monotonically increasing sequence satisfies these requirements; in particular, a timestamp is sufficient. (Note that the sender's clock need not be synchronized with any other clock, nor is relative clock drift a concern.) Our construction may be viewed as confirmation of some conventional wisdom: one should always date and sign one's correspondence!

The formal construction. Let $\mathbb{C}$ denote some family $\mathbb{C} = \{C_i\}$ of $(n_i, k_i)_{q_i}$, (e_i, L_i) list decodable coding schemes that has an efficient encoding procedure E and an efficient list decoding procedure LD for parameters (e_i, L_i). Note that the efficiency constraints imply that n_i grows polynomially with k_i (in fact, we are most interested in the case where this growth is linear), and that the decoded list sizes are polynomial in n_i.

Also assume we are given some signature scheme $(\textsc{Gen}, \textsc{Sig}, \textsc{Ver})$ which is existentially unforgeable under an adaptive chosen-message attack. Recall that the key generator $\textsc{Gen}$ requires a security parameter

[6] This can happen if two prior message-signature pairs map to two codewords separated by, say, the minimum distance of the code.

k'; for simplicity we assume wlog that the corresponding signature length is k'. We require the signature size to be small relative to the message size, therefore when using code C_i we use a security parameters of, say, $k'_i = \sqrt{k_i}$.

Additionally, the sender S maintains a state variable t, which is a counter initialized to 0. This counter will also appended to the message, so again we want it to be short. When using code C_i, we append the value of the counter using $k'_i = \sqrt{k_i}$ symbols in some canonical way; this provides for $q_i^{k'_i}$ unique counters, which is super-polynomial in n_i.

We now describe the dynamic coding scheme. The codes will have message lengths of $k_i - 2k'_i = k_i - 2\sqrt{k_i}$ and corresponding block lengths of n_i. The algorithms are as follows:

- $G(1^{k_i}, 1^{n_i})$: let $k'_i = \sqrt{k_i}$. Compute and output $(pk, sk) \leftarrow \textsc{Gen}(1^{k'_i})$.
- $S(\mathbf{m}, sk, aux = t)$: compute $\sigma \leftarrow \textsc{Sig}_{sk}(\mathbf{m} \circ t)$. Output $(E(\mathbf{m} \circ t \circ \sigma), aux' = t + 1)$.
- $R(\mathbf{r}, pk)$: list decode $\mathbf{r}$: $(\mathbf{m}_1 \circ t_1 \circ \sigma_1), \ldots, (\mathbf{m}_\ell \circ t_\ell \circ \sigma_\ell) \leftarrow LD(\mathbf{r})$. Consider all pairs $(\mathbf{m}_i, t_i)$ such that $\textsc{Ver}_{pk}(\mathbf{m}_i \circ t_i, \sigma_i) = 1$ and $t_i = \max_j t_j$. If at least one such $\mathbf{m}_i$ exists and are all the same message, output that message. Otherwise, output $\perp$.

Theorem 1. *Assuming one-way functions exist, the above dynamic code (G, S, R) uniquely decodes from error rate $\epsilon(\mathbb{C})$ and has information rate $R(\mathbb{C})$.*

Proof. Clearly the rate of the scheme is $R(\mathbb{C})$, because $\liminf_{i\to\infty}(k_i - 2\sqrt{k_i})/n_i = \liminf_{i\to\infty} k_i/n_i = R(\mathbb{C})$.

Now suppose for contradiction there exists a channel $\mathcal{C}$ with error rate ϵ that causes R to incorrectly decode some message with probability $1/p(n_i)$ for some polynomial $p(\cdot)$ and for infinitely many n_i. We will use $\mathcal{C}$ to construct a forger F for the signature scheme. F will receive message requests from $\mathcal{C}$ and use its signing oracle to simulate the sender, then pass authentic codewords to the channel. The channel will produce corrupted words, and the forger will simulate the recipient on them. If decoding is incorrect at any point (which is detectable), the incorrect output can be used to construct a forgery. We now proceed more formally.

First note that for infinitely many n_i, $\mathcal{C}$ makes fewer than $q_i^{k'_i}$ queries, and the counter values are all distinct. From now on, we consider only those n_i. Now $F^{\mathcal{O}}(pk)$ works as follows: let $k_i = (k'_i)^2$, and n_i be the block length corresponding to k_i, and $t \leftarrow 0$. Run $\mathcal{C}(pk)$. When $\mathcal{C}$ requests a message $\mathbf{m}$ to be sent, query $\sigma \leftarrow \mathcal{O}(\mathbf{m} \circ t)$, and return $\mathbf{w} = E(\mathbf{m}, t, \sigma)$

to the channel. Receive $\mathbf{r}$ (a corrupted version of $\mathbf{w}$) from the channel, where $\Delta(\mathbf{r}, \mathbf{w}) \leq \epsilon n_i$ (except with probability $\nu(n_i)$), and list decode: $(\mathbf{m}_1 \circ t_1 \circ \sigma_1), \ldots, (\mathbf{m}_\ell \circ t_\ell \circ \sigma_\ell) \leftarrow LD(\mathbf{r})$, where $\ell \leq L_i$. By assumption on $\mathcal{C}$, $(\mathbf{m} \circ t \circ \sigma)$ appears in the list (except with probability $\nu(n_i)$). If there exists some $\mathbf{m}_i$ such that $\mathrm{VER}_{pk}(\mathbf{m}_i \circ t_i, \sigma_i) = 1$ and $t_i > t$, or such that $m_i \neq m$ and $t_i = t$, then output $(\mathbf{m}_i \circ t_i, \sigma_i)$ as a forgery. Otherwise, increment t and repeat with the next message chosen by $\mathcal{C}$, until it aborts.

Note that R decodes incorrectly if and only if some σ_i is a valid signature of $m_i \circ t_i$, and either $t_i > t$, or $t_i = t$ and $m_i \neq m$ (because all counters in the experiment are unique). In the first case, $\mathcal{O}$ was never queried on $m_i \circ t_i$, because t_i is too large. In the second case, $\mathcal{O}$ was never queried on $m_i \circ t_i$ because only $m \neq m_i$ was queried with counter $t = t_i$. Therefore $(m_i \circ t_i, \sigma_i)$ constitutes a forgery. The success probability of $F^{\mathcal{O}}$ on security parameter k'_i is negligibly smaller than $p(n_i)$ (the success probability of $\mathcal{C}$ on block lengths of size n_i), and by assumption n_i grows polynomially with $k_i = (k'_i)^2$. This contradicts the unforgeability of the signature scheme, as desired.

Remark 2. A similar construction works in the *private-key setting*, in which the sender and recipient share a private key that is unknown to the channel. Instead of signing each message, the sender uses a *message authentication code* (MAC) that is existentially unforgeable under chosen-message attack.[7] We stress that the receiver remains stateless in this modified scheme. The only difference in the proof is that the forger cannot detect a successful forgery, but instead chooses a random element of the decoded list during a random query by the channel. This alters the concrete security analysis of the scheme, but still leads to a non-negligible chance of forgery.

Corollary 1. *Assuming one-way functions exist, there exist binary dynamic coding schemes with error rate $1/2 - \gamma$ for any $\gamma > 0$ and positive asymptotic information rate.*

Proof. Apply Theorem 1 to the concatenated codes of Guruswami and Sudan [7]. The Reed-Solomon concatenated codes are efficiently encodable, have asymptotic rate $\Omega(\gamma^8)$, and can be efficiently list decoded under error rate $1/2 - \gamma$, for any constant $\gamma > 0$. The algebraic-geometry concatenated codes also have efficient algorithms and asymptotic rate $\Omega(\gamma^6 \log 1/\gamma)$.

[7] Such a MAC can be constructed using a pseudorandom function family, which exists if and only if one-way functions exist [1].

Alternatively, one may use the efficiently list-decodable codes of Guruswami *et al* [5], which have asymptotic rate $\Omega(\gamma^4)$ for error rate $1/2-\gamma$.

Corollary 2. *Assuming one-way functions exist, there exist large-alphabet dynamic coding schemes with error rate $1-\sqrt{R}$ for any information rate $R>0$.*

Proof. Apply Theorem 1 to Reed-Solomon codes. These codes are efficiently encodable and list decodable under error rate $1-\sqrt{R}$ using the Guruswami-Sudan algorithm [6].

4.3 Optimality of the Model

In our model, the sender keeps a secret key and maintains some local state between each encoding, and the channel is computationally bounded. Propositions 1, 2, and 3 establish that these features are *essential*: under several natural relaxations of the model, decoding from high error rates is impossible.

The following well-known lemma will be useful in our proofs:

Lemma 1 (Plotkin bound). *For any $2n+1$ strings $\mathbf{x}_1,\ldots,\mathbf{x}_{2n+1}\in\{0,1\}^n$, there exist i,j such that $i\neq j$ and $\Delta(\mathbf{x}_i,\mathbf{x}_j)<n/2$.*

Proposition 1. *Consider any dynamic coding scheme for parameters $\{(n_i,k_i,q_i)\}_i$ where both the sender and receiver are stateless.*

If $q_i=2$ and $2^{k_i}\geq 2n_i+1$ for infinitely many i, an adversarial channel with any error rate $>1/4$ can cause incorrect decoding with probability non-negligible in n_i.

For any values of q_i, an adversarial channel with any error rate $>1/2$ can cause incorrect decoding with probability non-negligible in n_i.

These results hold even if the sender and receiver are randomized and have secret or public keys, and the channel is polynomially bounded.

Proof. We first prove the result for binary alphabets (i.e., $q_i=2$). (We assume $2^{k_i}\geq 2n_i+1$ simply to guarantee that at least $2n+1$ distinct messages can be sent.)

For any n and $\mathbf{x},\mathbf{x}'\in\{0,1\}^n$, if $\Delta(\mathbf{x},\mathbf{x}')<n/2$, define $M(\mathbf{x},\mathbf{x}')$ to be some canonical $\mathbf{r}$ such that $\Delta(\mathbf{x},\mathbf{r})<\lceil n/4\rceil$ and $\Delta(\mathbf{r},\mathbf{x}')\leq\lceil n/4\rceil$. Note that M is easily computable. We now describe a channel $\mathcal{C}$ which will cause an incorrect decoding with non-negligible probability.

$\mathcal{C}$ chooses two distinct messages $\mathbf{m},\mathbf{m}'$ at random and queries the sender on $\mathbf{m}$, yielding $\mathbf{x}\in\{0,1\}^n$, then passes it to the receiver without error. $\mathcal{C}$ then queries the sender on $\mathbf{m}'$, yielding $\mathbf{x}'\in\{0,1\}^n$. If

$\Delta(\mathbf{x}, \mathbf{x}') < n/2$, $\mathcal{C}$ sends either $M(\mathbf{x}, \mathbf{x}')$ or $M(\mathbf{x}', \mathbf{x})$ to the recipient, each with probability $1/2$ (this requires introducing at most $\lceil n/4 \rceil$ errors to $\mathbf{x}'$). Otherwise, $\mathcal{C}$ sends $\mathbf{x}'$ uncorrupted.

Conditioned on $\Delta(\mathbf{x}, \mathbf{x}') < n/2$, the receiver's view *of the second message* is distributed identically to its view in a world where $\mathbf{m}'$ is queried *first* and $\mathbf{m}$ is queried second. (This relies on the statelessness of both sender and receiver.) Therefore, $\mathcal{C}$ will cause incorrect decoding with probability at least $1/2$.

It remains to bound $\Pr[\Delta(\mathbf{x}, \mathbf{x}') < n/2]$. Consider a thought experiment in which the channel additionally queries $2n - 1$ more distinct random messages before querying $\mathbf{m}$ and $\mathbf{m}'$. By the Plotkin bound, the encodings of some two messages will have Hamming distance less than $n/2$. Since all messages are random and each encoding is independent (due to the sender's statelessness), $\Pr[\Delta(\mathbf{x}, \mathbf{x}') < n/2] \geq 1/\binom{2n+1}{2} = \Omega(1/n^2)$. This completes the proof for binary alphabets.

For the large-alphabet case, we apply a similar (but simpler) argument. Since *all* pairs of codewords of length n are within Hamming distance n for any alphabet, an adversarial channel with any error rate $> 1/2$ can cause incorrect decoding with probability $1/2$.

Proposition 2. *Consider any dynamic coding scheme for parameters* $\{(n_i, k_i, q_i)\}_i$ *where all the sender's inputs are known to the channel.*

If $q_i = 2$ *and* $2^{k_i} \geq 2n_i + 1$ *for infinitely many* i*, an adversarial channel with any error rate* $> 1/4$ *can cause incorrect decoding with probability at least* $\Omega(1/n)$*.*

For any values of q_i*, an adversarial channel with any error rate* $> 1/2$ *can cause incorrect decoding with probability at least* $1/2$*.*

These results hold even if the sender and receiver are randomized and stateful, the receiver has secret inputs, and the channel is polynomially bounded.

Proof. Because the sender is a polynomial-time algorithm with only public inputs, it can be "simulated" by the channel. That is, the channel $\mathcal{C}$ can simply use the encoding function as a subroutine. Thus, instead of making queries to the sender, $\mathcal{C}$ can simply simulate the process, encoding messages itself. This difference is essential: since the sender is not actually encoding these messages, his internal state remains unchanged. The channel can thus simulate the execution of the sender on a variety of inputs with the same internal state.

Once again we start with the binary case. As in the proof of Proposition 1, for any n and $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$, if $\Delta(\mathbf{x}, \mathbf{x}') < n/2$, define $M(\mathbf{x}, \mathbf{x}')$ to be some canonical $\mathbf{r}$ such that $\Delta(\mathbf{x},\mathbf{r}) < \lceil n/4 \rceil$ and $\Delta(\mathbf{r},\mathbf{x}') \leq \lceil n/4 \rceil$. We now describe an adversarial channel that can cause incorrect decoding:

The channel first makes a *real* query to the sender on a random message $\mathbf{m}_1$ and receives $\mathbf{x}_1$, its encoding. For block length n, the channel then *simulates* the encoding $2n$ more random, distinct messages $\mathbf{m}_2, \ldots, \mathbf{m}_{2n+1}$. (If the sender is stateful, the channel encodes with the sender's state *at the time* $\mathbf{m}_1$ *was encoded.*) Denote the encoding of $\mathbf{m}_i$ as $\mathbf{x}_i$. By the Plotkin bound, there exist $\mathbf{x}_i, \mathbf{x}_j$ such that $i \neq j$ and $\Delta(\mathbf{x}_i, \mathbf{x}_j) < n/2$. By symmetry and without loss of generality, $\Pr[i = 1] \geq 2/(2n+1)$. In the event that $i = 1$, the channel corrupts $\mathbf{x}_1$ in the following way: it sends $M(\mathbf{x}_1, \mathbf{x}_j)$ with probability 1/2, and $M(\mathbf{x}_j, \mathbf{x}_1)$ otherwise.

Conditioned on $i = 1$, the recipient's view is distributed identically to the case where the roles of $\mathbf{x}_1$ and $\mathbf{x}_j$ are switched, and $\mathbf{x}_j$ was the real query. Therefore the recipient will decode incorrectly with probability $\Omega(1/n)$.

For large alphabets, a similar argument works with only one simulated message encoding.

Proposition 3. *Consider any dynamic coding scheme for parameters* $\{(n_i, k_i, q_i)\}_i$ *where and receiver is stateless and has only public inputs.*

If $q_i = 2$ *and* $2^{k_i} \geq 2n_i + 1$ *for infinitely many* i*, a* computationally unbounded *adversarial channel with any error rate* $> 1/4$ *can cause incorrect decoding with probability at least* $1/2$.

For any values of q_i*, a* computationally unbounded *adversarial channel with any error rate* $> 1/2$ *can cause incorrect decoding with probability at least* $1/2$.

These results hold even if the sender and receiver are randomized, and the sender has a public key.

Proof. We again start with the case of binary alphabets. Note that because the receiver is stateless, its output distribution on a given word $\mathbf{r}$ is always the same, regardless of what transmissions have preceded $\mathbf{r}$. We now describe an unbounded adversarial channel that can cause incorrect decoding:

For block length n, the channel will make up to $2n+1$ arbitrary distinct message queries $\mathbf{m}_1, \ldots, \mathbf{m}_{2n+1}$. Denote the sender's encoding of $\mathbf{m}_i$ as $\mathbf{x}_i$. When receiving $\mathbf{x}_j$, the channel exhaustively searches all $\mathbf{r}$ such that $\Delta(\mathbf{x}_j, \mathbf{r}) \leq \lceil n/4 \rceil$. Because the receiver is stateless and only has *public* inputs, the unbounded channel can compute the receiver's output distribution for word $\mathbf{r}$. There are two cases: (1) if for some $\mathbf{r}$ the receiver would fail to output $\mathbf{m}_j$ with probability $\geq 1/2$, the channel corrupts $\mathbf{x}_j$

as $\mathbf{r}$, sends it to the receiver, and halts; (2) otherwise, the channel sends $\mathbf{x}_j$ uncorrupted and makes the next query.

We now need only argue that the case (1) eventually occurs for some query. By the Plotkin bound, for some j there exists $i < j$ and an $\mathbf{r}$ such that $\Delta(\mathbf{x}_i, \mathbf{r}) \leq \lceil n/4 \rceil$ and $\Delta(\mathbf{r}, \mathbf{x}_j) < n/4$. Suppose that for queries $1, \ldots, j-1$, case (1) did not occur. Then by this assumption, on input $\mathbf{r}$ the receiver outputs $\mathbf{m}_i$ with probability $\geq 1/2$. Therefore $\mathbf{r}$ is close enough to $\mathbf{x}_j$ but fails to decode to $\mathbf{m}_j$ with probability at least $1/2$, and we are done.

For large alphabets, a similar argument works with only two distinct message queries.

5 Acknowledgements

We are grateful to Adam Smith for helpful discussions, including his explanation of the results of Langberg [8] in our framework. We also thank the anonymous reviewers for their helpful and thorough comments.

References

1. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
2. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
3. P. Gopalan, R. J. Lipton, and Y. Z. Ding. Error correction against computationally bounded adversaries. Manuscript, October 2004.
4. V. Guruswami. List decoding with side information. In *18th IEEE Annual Conference on Computational Complexity*, pages 300–312, 2003.
5. V. Guruswami, J. Håstad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. In *Proceedings of the 38th Annual Allerton Conference on Communication, Control and Computing*, 2000.
6. V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *IEEE Symposium on Foundations of Computer Science*, pages 28–39, 1998.
7. V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 181–190. ACM Press, 2000.
8. M. Langberg. Private codes or succinct random codes that are (almost) perfect. In *Proceedings of the forty-fifth annual IEEE Symposium on Foundations of Computer Science*, 2004.
9. R. J. Lipton. A new approach to information theory. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708. Springer-Verlag, 1994.
10. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, 1990.