# Efficient Error-Correcting Codes for Sliding Windows

Ran Gelles[1], Rafail Ostrovsky[1,2,⋆], and Alan Roytman[1]

[1] Department of Computer Science, University of California, Los Angeles
[2] Department of Mathematics, University of California, Los Angeles
{gelles,rafail,alanr}@cs.ucla.edu

**Abstract.** We consider the task of transmitting a data stream in the sliding window model, where communication takes place over an *adversarial* noisy channel with noise rate up to 1. For any noise level $c < 1$ we design an efficient encoding scheme, such that for any window in which the noise level does not exceed $c$, the receiving end decodes at least a $(1-c-\varepsilon)$-prefix of the window, for any small $\varepsilon > 0$. Decoding more than a $(1-c)$-prefix of the window is shown to be impossible in the worst case, which makes our scheme optimal in this sense. Our scheme runs in polylogarithmic time per element in the size of the window, causes constant communication overhead, and succeeds with overwhelming probability.

## 1 Introduction

As data continues to grow in size for many real world applications, streaming algorithms play an increasingly important role. Big data applications, ranging from sensor networks [9] to analyzing DNA sequences [5], demand streaming algorithms in order to deal with the tremendous amount of information being generated in a very short period of time. For certain applications, it is useful to only maintain statistics about recent data (rather than the entire stream). For instance, we may be interested in analyzing stock market transactions within the last hour, or monitoring and analyzing packets entering a network to detect suspicious activity, or identifying patterns in genomic sequences. This model is known as the *sliding windows model*, in which we care about a fixed-length window of time (say, 1 hour), which "slides" forward as time moves on (e.g., the last one hour, etc.).

While in the standard sliding window model, the aim is to maintain some statistics of an input stream (usually, using only polylogarithmic memory), our

⋆ Research supported in part by NSF grants CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

V. Geffert et al. (Eds.): SOFSEM 2014, LNCS 8327, pp. 258–268, 2014.

focus is applications in which the stream is generated in one place and then communicated to another place, as is the case for sensor networks, for instance. Inevitably, such applications are particularly vulnerable to data corruption while transmitting symbols: sensors are often placed in harsh environments connected to a base unit by a wireless channel which is susceptible to various kinds of errors (a weak signal, noise from other nearby transmitters, physical blockage of the transmitting medium, or even physical damage to the sensor). A natural question arises, how can we handle this constant stream of corrupted data while still being able to make sense of the information?

Our paper precisely aims to answer this question. We study encoding schemes for transmitting data streams in the sliding windows setting. We aim to design encoding schemes that can tolerate a high amount of errors while keeping the added redundancy low. Informally, we mainly consider noise rates above $1/2$, while adding only a constant overhead.[1] Another requirement of our scheme is to be efficient, that is, the encoding and decoding time (per element) must be at most polylogarithmic in the window size $N$.[2]

Before we describe our result in more details, let us explain why simple solutions for this task fail to work. Assume that an encoding scheme is meant to protect against a burst of noise of length $n$. A straightforward solution would be to cut the stream into chunks of size $N = (2 + \varepsilon)n$ and encode each chunk using some standard error-correcting code that can deal with a fraction of errors of almost $1/2$ (e.g., a Shannon error-correcting code [12] with good distance). This solution has three downsides. First, if the noise within one block is even slightly more than $n$, there are some messages for which the entire block will be incorrectly decoded. Second, the receiving side will be able to decode a block only after receiving the block entirely, which implies a delay of $2n$. Last, and maybe the most important downside, is that the rate of errors in one window using this solution cannot exceed $1/2$. We can replace the Shannon code with a code that resists a fraction of noise arbitrarily close to 1, such as the code of Micali, Peikert, Sudan, and Wilson [8], however this code uses an alphabet size that grows with the block size, and the obtained scheme would have a super-constant overhead. No codes over a constant-size alphabet are known to resist more than $1/2$ fraction of errors.

**Contributions and Techniques.** We provide a family of constant-rate encoding schemes in the sliding window model such that for any $c < 1$, if the rate of noise in the last window is less than $c$, the decoding of a $(1 - c)$-prefix of the window succeeds with overwhelming probability. Our scheme has polylogarithmic time complexity per element and linear space complexity. Formally,

[1] Loosely speaking, the *overhead* of a scheme is the amount of data communicated in one window divided by the amount of input data in that window, as a function of the windows size $N$.

[2] A similar polylogarithmic bound on the memory consumption would also be desired. However, at least for the sender, we show that the memory consumption must be at least linear in the window size, $\Omega(N)$.

**Theorem 1.** *For any $c < 1$ and any $\varepsilon > 0$, there exists an efficient encoding/decoding scheme for streams in the sliding window model with the following properties.*

*For any window $W$ of size $N$, if the fraction of errors within $W$ is less than $c$, then except with negligible probability $N^{-\omega(1)}$, the decoder correctly decodes a prefix of $W$ of size at least $(1 - c - \varepsilon)N$. The decoding takes poly$(\log N)$ time per element and the encoding takes $O(1)$ amortized time.*

Clearly, for a noise rate $c$, there is no hope of decoding more than a $(1 - c)$-prefix of the window (e.g., consider decoding at time $N$ when the channel was "blocked," i.e. fully corrupted, starting at time $(1 - c)N$), and our protocol is optimal in this aspect. Although a suffix of the current window may not be decoded, the data is not lost and will be decoded eventually as the window slides (if the noise rate is below $c$). Moreover, the delay of our scheme is $(c+\varepsilon)N$, since we only guarantee that the receiver correctly decodes the first $(1 - c - \varepsilon)N$ symbols in each window, even though $N$ symbols have already been received. Given that our goal is to decode as large a prefix as possible, and that we cannot hope to decode more than a $(1-c)$-prefix of the window, any scheme must incur a delay of at least $cN$, and hence the delay of our scheme is near-optimal.

To provide some intuition and insight into the techniques we use, we begin with the work of Franklin, Gelles, Ostrovsky, and Schulman for encoding streams in the unbounded model [1]. Roughly speaking, their scheme works as follows. At any particular time $n$ (i.e., the length of the stream seen so far), the stream is split into blocks of size $\log n$, each of which is encoded using an online coding scheme called a *tree-code* [10,11]. At each time step, a constant number of blocks are chosen uniformly at random and the next symbol is sent (in each chosen block) according to the encoding determined by the tree code (it is assumed that the sender and receiver have access to a shared random string, so the receiver knows which random blocks are chosen by the sender). This encoding is concatenated with a weak message-authentication code, that detects a large fraction of errors and effectively makes the scheme resistant to higher noise rates. We observe that tree codes have several shortcomings. First, decoding a codeword using a tree code takes exponential time in the block size. Second, tree codes (with constant rate) are known to exist, but take exponential time to construct (though efficient relaxations exist, see [2]). These shortcomings imply that any efficient scheme is restricted to using tree codes on words of at most logarithmic length, and thus obtain at least a polynomially small failure probability.

Our scheme takes inspiration from the ideas in [1]. We also divide our window into blocks of equal size, however we consider only blocks that are in the window. The key observation is that we no longer need the "online" property of tree-codes. This is because, in the sliding window model, we care only about a fixed window size $N$, and the blocks within this window are well defined in advance. Hence, the blocks do not increase in size when more elements arrive, as in the unbounded case [1]. To completely avoid the need for an "online" coding, our scheme waits until an entire block arrives and then encodes it using an

error-correcting block code.[3] This allows us to replace the tree-code with a very efficient block code, e.g., the almost-optimal linear-time code of Guruswami and Indyk [3], and improve upon the efficiency in [1] to constant amortized time for the sender and polylogarithmic time for the receiver per time step. Moreover, we are not bound to logarithmic block size anymore, and by using polylogarithmic block size we reduce the failure probability from polynomially small as in [1] to negligible[4], while keeping the scheme time-efficient.

The resulting scheme is very similar to, and can be seen as an extension of the "code scrambling" method suggested by Lipton [7]. Lipton's scheme encodes a single message by chopping the message into blocks, with encoding and decoding being done per block. Then, the scheme permutes the entire message and adds a random mask to the permuted string, using some shared randomness. The scheme has a negligible failure probability for any random, computationally bounded channel with error probability $p < 1/2$. In contrast, our scheme works in the sliding window model and potentially encodes an infinite message. This requires a more clever "scrambling" technique than simply permuting the entire message. The random mask performed in [7] is replaced with an error-detection code, which increases the resilience of our scheme. We analyze our scheme against an unbounded adversary and show that it can handle a corruption rate that is arbitrarily close to 1 while guaranteeing a constant rate and a negligible failure probability.

While our scheme resists fully adversarial channels, it requires a large amount of shared-randomness. We can reduce the amount of shared randomness via standard techniques (same as in [7]), under the assumption of a computationally bounded channel. In this case a short random key is assumed to be shared between the parties from which randomness is expanded as needed via a pseudo-random generator.

**Other Related Work.** Coding schemes that assume the parties pre-share some randomness first appeared in [13], and were greatly analyzed since. The main advantage of such codes is that they can deal with adversarial noise, rather than random noise. Langberg [6] showed codes that approach Shannon's bound and require only $O(\log n)$ randomness for a block size of $n$, as well as an $\Omega(\log n)$ lower bound on the amount of necessary randomness. The construction of Langberg also implies an efficient code with $O(n \log n)$ randomness. This result was improved to $n + o(n)$ randomness by Smith [14]. Explicit constructions with $o(n)$ randomness are not yet known (see [14]).

## 2 Preliminaries

For a number $n$ we denote by $[n]$ the set $\{1, 2, \ldots, \lfloor n \rfloor\}$, and for a finite set $\Sigma$ we denote by $\Sigma^{\leq n}$ the set $\cup_{k=1}^{n} \Sigma^k$. Throughout the paper, log() denotes the

[3] For a block size $B$, this incurs an additional delay of $O(B)$, but since our goal is to decode a $(1 - c - \varepsilon)$-prefix of the window, we already have an inherent delay of $cN$ in any case (clearly, $B$ will be sub-linear in $N$).

[4] See Section 2 for a formal definition of a negligible function.

binary logarithm (base 2). A *data stream* $S$ is a (potentially infinite) sequence of elements $(a_0, a_1, a_2 \ldots)$ where element $a_t \in \{1, \ldots, u\}$ arrives at time $t$. In the *sliding window model* we consider at each time $t \geq N$ the last $N$ elements of the stream, i.e. the window $W = \{a_{t-(N-1)}, \ldots, a_t\}$. These elements are called *active*, whereas elements that arrived prior to the current window $\{a_i \mid 0 \leq i < t-(N-1)\}$ are *expired*. For $t < N$, the window consists of all the elements received so far, $\{a_0, \ldots, a_t\}$. Finally, we say that a function $f(N)$ is *negligible* if for any constant $c > 0$, $f(N) < \frac{1}{N^c}$ for sufficiently large $N$.

*Shared Randomness Model.* We assume the following model known as the *shared-randomness model*. The legitimate users (the sender and the receiver) have access to a random string $\mathsf{Rand}$ of unbounded length, which is unknown to the adversary. Protocols in this model are thus *probabilistic*, and are required to succeed with high probability over the choice of $\mathsf{Rand}$. We assume that all the randomness comes from $\mathsf{Rand}$ and that for a fixed $\mathsf{Rand}$ the protocols are deterministic.

*Error-detection Codes: The Blueberry Code.* The Blueberry-Code (BC) is a randomized error-detection code introduced by Franklin, Gelles, Ostrovsky, and Schulman [1], in which each symbol is independently embedded into a larger symbol space via a random mapping. Since the mapping is random and unknown to the adversary, each corruption is detected with some constant probability.

**Definition 2.** *For $i \geq 1$ let $\mathsf{BC}_i : \Sigma_I \to \Sigma_O$ be a random and independent mapping. The* Blueberry code *maps a string $x \in \Sigma_I^*$ into a string $\mathsf{BC}(x) \in \Sigma_O^*$ of the same length where $\mathsf{BC}(x) = \mathsf{BC}_1(x_1)\mathsf{BC}_2(x_2)\cdots\mathsf{BC}_n(x_n)$.*

Conditioned on the fact that the legitimate users use a message space $\Sigma_I \subset \Sigma_O$, each corruption can independently be detected with probability $1 - q$, where $q = \frac{|\Sigma_I|-1}{|\Sigma_O|-1}$ (hence, $q$ is the probability that it remains undetected). The users are assumed to be sharing the random mappings $\mathsf{BC}_i$s at the start of the protocol.

*Linear-time Error-correcting Codes.* We will use error-correcting codes which are very efficient (i.e., linear-time in the block size for encoding and decoding). Such codes were initially (explicitly) constructed by Spielman [16] (see also [15]). Specifically, we use a linear-time error-correcting code with almost optimal rate given by Guruswami and Indyk [3]:

**Lemma 3 ([3]).** *For every rate $0 < r < 1$, and all sufficiently small $\delta > 0$, there exists an explicit family of error-correcting codes $\mathsf{ECC} : \Sigma^{rn} \to \Sigma^n$, $\mathsf{ECC}^{-1} : \Sigma^n \to \Sigma^{rn}$ over an alphabet of size $|\Sigma| = O_{\varepsilon,r}(1)$ that can be encoded and (uniquely) decoded in time linear in $n$ from a fraction $e$ of errors and $d$ of erasures provided that $2e + d \leq (1 - r - \delta)$.*

*Communication and Noise Model.* Our communication model consists of a channel $ch : \Sigma \to \Sigma$ subject to corruptions made by an adversary (or by the channel itself). For all of our applications we assume that, at any given time slot (i.e.,

for any arriving element of the stream), a constant number $R$ of channel instantiations are allowed. We say that $R$ is the blowup or overhead of our scheme.

The noise model is such that any symbol $\sigma$ sent through the channel can turn into another symbol $\tilde{\sigma} \in \Sigma$. It is not allowed to insert or delete symbols. We assume the noise is *adversarial*, that is, the noise is made by an all-powerful entity that is bounded only in the amount of noise it is allowed to introduce (the adversarial noise model subsumes the common noise models of random-error and burst-error). We say that the *corruption rate* in the window is $c$, if the fraction of corrupted transmissions over the last $N$ time steps is at most $c$.

## 3 A Polylogarithmic Sliding Window Coding Scheme

We consider the problem of streaming authentication in the sliding windows model. In this setting, we have a fixed window size $N$ (assumed to be known in advance). At each time step, one element expires from this window and one element arrives.

The idea is the following. The sender maintains blocks of size $s$ of elements from the current window, which means there are $\frac{N}{s}$ blocks in the window. All the blocks have the same amount of active elements, except the last block which may only be partially full, and the first one which may have several elements that have already expired. When all the elements of the first block have expired we remove it and re-number the indices; additionally, when the last block becomes full we introduce a new (empty) block to hold the arriving elements.

When the sender has received an entire block[5] from the stream, the block is encoded using a linear-time error-correcting code [3]. At each time step, one of the $\frac{N}{s}$ blocks is chosen uniformly at random and the next (unsent) symbol of the encoded string is communicated over the channel after being encoded via an error-detection code (this gives us better rate, and maximizes the amount of decoded information). The protocol is described in Algorithm 1.

We now continue to analyze the properties of Algorithm 1, and show how to fix its parameters so it will satisfy the conditions of Theorem 1.

**Proposition 4.** *Suppose that the rate of corruptions in the window, for a given time $t$, is at most $c < 1$. Fix a small enough $\varepsilon > 0$. Denote by $\mathsf{TOTAL}_k$ the number of total transmissions for $B_k$ up to time $t$; by $\mathsf{CORRUPT}_k$ the number of transmissions in $B_k$ up to time $t$ which are corrupted; and by $\mathsf{ERR}_k$ the number of errors in $B_k$ up to time $t$ (i.e., corrupted transmissions that were not identified by the error-detection code $\mathsf{BC}$). Then, for any $k \in [(1-c-\varepsilon)\frac{N}{s}]$ the following holds.*

1. $(c+\varepsilon)Rs \leq \mathbb{E}[\mathsf{TOTAL}_k] \leq Rs$.
2. $\mathbb{E}[\mathsf{CORRUPT}_k] \leq cRs$.
3. $\mathbb{E}[\mathsf{ERR}_k] \leq cqRs$.

[5] We assume only complete blocks, that is, the scheme skips the last block until it contains exactly $s$ elements. This causes an additional delay of $s$ time slots.

Let the parameters of the protocol $c, \varepsilon < 1$ be fixed. Let $s, R \in \mathbb{N}$ be such that $s < N$. Assume an online (symbol-wise) error-detection code $\mathsf{BC}$ with failure probability $q$ per corrupted symbol. Assume a linear-time error-correction code $\mathsf{ECC}() : \Sigma^s \to \Sigma^{s/r}$ with a rate $r < 1$ to be fixed later.

Sender: Maintain blocks $B_k$ of size at most $s$ for $1 \le k \le \lfloor \frac{N}{s} \rfloor + 1$; any arriving element is appended to the last non-empty block.[6] If all elements in the first block expire, add a new (empty) block and remove $B_1$. Reindex the blocks so that the first block in the window is $B_1$ and the last is $B_{\lfloor \frac{N}{s} \rfloor + 1}$.
Maintain a counter $count_k$ for each block (initialized to 0 when the block is added).
**foreach** *time step* $t$ **do**
  **for** $j = 1, \ldots, R$ **do**
    Choose $k \in [\frac{N}{s}]$ uniformly at random.
    $count_k \leftarrow count_k + 1$.
    Send the next symbol of $\mathsf{BC} \circ \mathsf{ECC}(B_k)$ which has not yet been sent according to $count_k$ (send $\perp$, if all the symbols were communicated).
  **end**
**end**

Receiver: The receiver maintains the same partition of the stream into blocks (with consistent indexing).
**foreach** *time step* $t$ **do**
  **for** $j = 1, \ldots, R$ **do**
    Assume the sender, at iteration $(t, j)$, sent a symbol from $B_k$. Let $x^k$ denote the string obtained by concatenating all the symbols received so far that belong to the same $B_k$ (preserving their order of arrival).
    Let $B'_k = \mathsf{ECC}^{-1} \circ \mathsf{BC}^{-1}(x^k)$.
  **end**
**end**
**Output** $B'_k$ for any $k \in [\frac{N}{s}]$.

**Algorithm 1.** Sliding window error-correcting scheme

*Proof.*

1. Define $\mathsf{TOTAL}_k$ to be the number of times bucket $B_k$ is chosen up to time $t$ (here, we mean the same bucket of data, regardless of the fact that its index $k$ is changed over time). Since the number of elements which appear in the window after $B_k$ is at least $N - ks$ and at most $N - (k-1)s$, and because $B_k$ is chosen with probability exactly $\frac{s}{N}$, we have that $\mathbb{E}[\mathsf{TOTAL}_k] = \Theta(R(N - ks) \cdot \frac{s}{N})$. Specifically, for any $k$, $R(N - ks)\frac{s}{N} \le \mathbb{E}[\mathsf{TOTAL}_k] \le R(N - ks + s)\frac{s}{N}$. The result follows since $k \le (1 - c - \varepsilon)\frac{N}{s}$.

[6] For the very first elements of the stream, we artificially create a window of size $N$ with, say, all 0's (as if the scheme had already been running for $N$ time steps) and similarly divide it up into blocks. This is done to keep notation consistent.

2. $\mathsf{CORRUPT}_k$ is the amount of corrupted transmissions in $B_k$ up to time $t$. The number of corrupted transmissions in the window is at most $cRN$, and since each transmission has probability $s/N$ of belonging to $B_k$, we get $\mathbb{E}[\mathsf{CORRUPT}_k] \leq RcN \cdot \frac{s}{N} = cRs$.
3. We use an error-detection code in which any change is caught with probability $1-q$ but makes an error with probability $q$. Assuming that the rate of corruptions in the window is at most $c$, we have $\mathbb{E}[\mathsf{ERR}_k] \leq cRN \cdot q \cdot \frac{s}{N} = cqRs$.

□

**Proposition 5.** *Suppose the current window's corruption rate is at most $c$ for some constant $c > 0$, and let $\varepsilon > 0$ be any sufficiently small constant. Suppose we divide up the stream into blocks of size $s$. Then there exist constants $R, q, r, \delta = O_{c,\varepsilon}(1)$ such that, except with probability $N2^{-\Omega(s)}$, $B_k$ is correctly decoded by the receiver for every $k \in [(1-c-\varepsilon)\frac{N}{s}]$.*

*Proof.* We consider the worst-case scenario in which the adversary corrupts a $c$-fraction of the transmissions, and moreover all corruptions occur in the last $cRN$ transmissions (as this simultaneously maximizes the expected number of corruptions in each block $B_k$ with $k \in [(1-c-\varepsilon)\frac{N}{s}]$, since the expected number of corruptions of such blocks grows with time as long as the block remains in the window).

For a specific $k$, the probability of incorrectly decoding block $B_k$ is bounded by $\Pr[2\mathsf{ERR}_k + \mathsf{DEL}_k > (1-r-\delta)\frac{s}{r}]$. Here, $r$ and $\delta$ are the two parameters of the error-correction scheme specified in Lemma 3. Namely, $\frac{1}{r}$ is the overhead incurred due to encoding each block of size $s$ and $\delta$ is a parameter which trades off error tolerance against alphabet size.

By Proposition 4, we know $\mathbb{E}[\mathsf{ERR}_k] = cqRs$. Hence, using Chernoff bounds we know that for any $\xi > 0$, we have $\Pr[\mathsf{ERR}_k \geq (1+\xi)\mathbb{E}[\mathsf{ERR}_k]] \leq e^{-\frac{\xi^2}{3}cqRs}$. Deletions in block $B_k$ come from two sources. The first source, denoted by $\mathsf{D}_k^1$, stems from choosing block $B_k$ less than $s/r$ times (i.e., $\mathsf{TOTAL}_k$ is small). The second source, denoted by $\mathsf{D}_k^2$, comes from the $\mathsf{BC}$ code detecting corruptions (note that $\mathsf{DEL}_k = \mathsf{D}_k^1 + \mathsf{D}_k^2$).

Note that $\mathsf{D}_k^1 = \max(s/r - \mathsf{TOTAL}_k, 0)$, and in order to make it small with high probability, we can require $\mathbb{E}[\mathsf{TOTAL}_k] > s/r$. Using Proposition 4, $\mathbb{E}[\mathsf{TOTAL}_k] \geq (c+\varepsilon)Rs$, thus by choosing $R = \frac{(1+\xi)}{r(c+\varepsilon)}$ for some $\xi > 0$, and applying Chernoff bounds, we get that

$$\Pr\left[\mathsf{TOTAL}_k \leq \frac{s}{r}\right] \leq \Pr\left[\mathsf{TOTAL}_k < \left(1 - \frac{\xi}{1+\xi}\right)\mathbb{E}[\mathsf{TOTAL}_k]\right] < e^{-\frac{\xi^2 s}{2(1+\xi)r}}.$$

Hence, except with exponentially small probability, we know $\mathsf{TOTAL}_k \geq \frac{s}{r}$, which implies that $\mathsf{D}_k^1 = 0$. Next, observe that $\mathsf{ERR}_k + \mathsf{D}_k^2 = \mathsf{CORRUPT}_k$ and that by Proposition 4 and by applying the Chernoff bound we know that for any $\xi > 0$, $\Pr[\mathsf{CORRUPT}_k > (1+\xi)cRs] < e^{-\frac{\xi^2}{3}cRs}$.

Putting these bounds together, we know that for any constant $\xi > 0$,

$$2\mathsf{ERR}_k + \mathsf{DEL}_k = \mathsf{ERR}_k + \mathsf{CORRUPT}_k \leq (1+\xi)cqRs + (1+\xi)cRs \qquad (1)$$

(except with probability exponentially small in $s$). As long as this term is smaller than $(1-r-\delta)\frac{s}{r}$, then by Lemma 3 block $B_k$ will be decoded correctly. Recalling that we set $R = \frac{(1+\xi)}{r(c+\varepsilon)}$ and substituting into the right-hand side of Eq. (1), we get the following constraint on $r$:

$$r \leq 1 - \delta - \frac{c}{c+\varepsilon}(1+\xi)^2(1+q). \tag{2}$$

Hence, for any constant rate $0 < r < 1 - \frac{c}{c+\varepsilon}$, we can always choose sufficiently small constants $\delta, q, \xi = O_{c,\varepsilon}(1)$ so that the constraint in Eq. (2) is satisfied.

So far we have only argued that we can decode *one* block $B_k$ correctly except with probability exponentially small in $s$. We simply apply the union bound to get that, except with probability $N2^{-\Omega(s)}$, *every* block $B_k$ for $k \in [(1-c-\varepsilon)\frac{N}{s}]$ can be decoded correctly. □

Hence, with very high probability, we are able to guarantee that the entire prefix of the window can be decoded correctly at each time step. We now seek to analyze the efficiency of our scheme.

**Proposition 6.** *For any time $t$, the time complexity of Algorithm 1 is $O(1)$ (amortized) for the sender, and $O(s)$ for the receiver.*

*Proof.* Omitted. □

Finally, we set $s = \omega(\log N)$ and obtain Theorem 1 immediately from Proposition 5 and Proposition 6.

*Memory Consumption.* We now consider the memory requirements of the sender and receiver. We focus on the space required for the "work" memory, which we consider to be any memory required to perform computation that is separate from the space for input, output, and randomness bits.

It is easy to see that, in our scheme, both the sender and receiver take linear space $O(N)$. For the sender, we obtain a matching $\Omega(N)$ lower bound: consider the case that the channel is completely "jammed" between time 0 and $cN$ (each symbol is replaced with a random symbol), and then no more errors happen until time $N$. Since the noise rate in the first window is $c$, we expect the decoder to correctly output elements $a_1, \ldots, a_{(1-c)N}$ at time $N$. Since no information passed through the channel until time $cN$, at that time, the sender must possess (at least) the information $a_1, \ldots, a_{cN}$, thus his memory is lower bounded by $cN = \Omega(N)$.

As for the decoder, technically, the size of the output memory must be $\Omega(N)$ in order to decode a prefix of the window. However, if we wish to obtain a lower bound on the decoder's "work" memory, we note the following. In order for the decoder to guarantee at most a negligible probability of failure, the decoder's "work" memory must be $\omega(\log N)$. Otherwise, the decoder can save at most $O(\log N)$ transmissions, where a proportion $c$ of them is corrupt in expectation. Regardless of the coding used for these elements, it would fail with non-negligible probability. Closing the memory-gap for the decoder is left for further research.

*Effective Window.* Since the decoder is able to decode a $(1-c-\varepsilon)$-prefix of the current window, we can think of him as effectively decoding, at each time $t$ a sliding window of size $N'$, where $N' = (1-c-\varepsilon)N$, whose start point is the same as the real window. The effective window ends $cN$ time steps before the real window of the stream, which is precisely why the delay of our scheme is $cN$.

## 4 Conclusions and Open Questions

We have shown an efficient (polylogarithmic-time) coding scheme for data streams in the sliding window model. Somewhat surprisingly, while solving a problem in the sliding window model is usually a more difficult task than in the unbounded model, the case of communicating a stream in the sliding window model is simpler than the unbounded case. This allows us to improve on the methods built for the unbounded case [1] and achieve a more simple and efficient scheme for sliding windows.

While standard error-correcting schemes can resist a noise rate of at most $1/2$ (in one window), our scheme allows any error rate less than one at the cost of requiring the parties to pre-share some randomness before running the scheme. Our scheme is also advantageous in terms of delay.

While we aimed at achieving a constant-rate scheme, we have not analyzed the minimal possible rate as a function of the noise ($R$ is clearly lower bounded by $1/(1-c)$). Achieving an efficient scheme with optimal rate remains as an open question.

Finally, we mention that we can obtain a scheme that assumes no shared randomness, if we restrict the error rate up to $1/2$, and assume an *additive* (a.k.a. *oblivious*) adversary. Here, the channel fixes an error string to be added to the transmitted codeword, without seeing the transmitted codeword (though it can depend on the message and the coding scheme). This is done by employing the techniques of Guruswami and Smith [4]. Although this construction is somewhat technically involved, there are no novel techniques. We omit the details due to lack of space. Since the error rate is bounded by $1/2$, the only advantage this scheme has over the naïve approach described in Section 1, is reducing the delay.

## References

1. Franklin, M., Gelles, R., Ostrovsky, R., Schulman, L.J.: Optimal coding for streaming authentication and interactive communication. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 258–276. Springer, Heidelberg (2013)
2. Gelles, R., Moitra, A., Sahai, A.: Efficient and explicit coding for interactive communication. In: FOCS 2011, pp. 768–777 (2011)
3. Guruswami, V., Indyk, P.: Linear-time encodable/decodable codes with near-optimal rate. IEEE Trans. on Information Theory 51(10), 3393–3400 (2005)
4. Guruswami, V., Smith, A.: Codes for computationally simple channels: Explicit constructions with optimal rate. In: FOCS 2010, pp. 723–732 (2010)

5. Kienzler, R., Bruggmann, R., Ranganathan, A., Tatbul, N.: Large-scale DNA sequence analysis in the cloud: a stream-based approach. In: Alexander, M., et al. (eds.) Euro-Par 2011, Part II. LNCS, vol. 7156, pp. 467–476. Springer, Heidelberg (2012)
6. Langberg, M.: Private codes or succinct random codes that are (almost) perfect. In: FOCS 2004, pp. 325–334. IEEE Computer Society, Washington, DC (2004)
7. Lipton, R.: A new approach to information theory. In: Enjalbert, P., Mayr, E.W., Wagner, K.W. (eds.) STACS 1994. LNCS, vol. 775, pp. 699–708. Springer, Heidelberg (1994)
8. Micali, S., Peikert, C., Sudan, M., Wilson, D.A.: Optimal error correction against computationally bounded noise. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 1–16. Springer, Heidelberg (2005)
9. Munir, S., Lin, S., Hoque, E., Nirjon, S., Stankovic, J., Whitehouse, K.: Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks. In: IPSN 2010, pp. 303–314 (2010)
10. Schulman, L.J.: Deterministic coding for interactive communication. In: STOC 1993, pp. 747–756 (1993)
11. Schulman, L.J.: Coding for interactive communication. IEEE Transactions on Information Theory 42(6), 1745–1756 (1996)
12. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review 5(1), 3–55 (2001), originally appeared in Bell System Tech. J. 27, 379–423, 623–656 (1948)
13. Shannon, C.E.: A note on a partial ordering for communication channels. Information and Control 1(4), 390–397 (1958)
14. Smith, A.: Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In: SODA 2007, pp. 395–404 (2007)
15. Spielman, D.: Computationally efficient error-correcting codes and holographic proofs. Ph.D. thesis, Massachusetts Institute of Technology (1995)
16. Spielman, D.: Linear-time encodable and decodable error-correcting codes. IEEE Transactions on Information Theory 42(6), 1723–1731 (1996)