Optimal Coding for Streaming Authentication and Interactive Communication

Matthew Franklin[1], Ran Gelles[2],
Rafail Ostrovsky[2,3,⋆], and Leonard J. Schulman[4,⋆⋆]

[1] Department of Computer Science, University of California, Davis
franklin@cs.ucdavis.edu
[2] Department of Computer Science, University of California, Los Angeles
{gelles,rafail}@cs.ucla.edu
[3] Department of Mathematics, University of California, Los Angeles
[4] E&AS Division, Caltech
schulman@caltech.edu

Abstract. Error correction and message authentication are well studied in the literature, and various efficient solutions have been suggested and analyzed. This is however not the case for *data streams* in which the message is very long, possibly infinite, and not known in advance to the sender. Trivial solutions for error-correcting and authenticating data streams either suffer from a long delay at the receiver's end or cannot perform well when the communication channel is noisy.

In this work we suggest a constant-rate error-correction scheme and an efficient authentication scheme for data streams over a noisy channel (one-way communication, no feedback) in the shared-randomness model. Our first scheme does not assume shared randomness and (non-efficiently) recovers a $(1-2c)$-fraction prefix of the stream sent so far, assuming the noise level is at most $c < 1/2$. The length of the recovered prefix is tight.

To be able to overcome the $c = 1/2$ barrier we relax the model and assume the parties pre-share a secret key. Under this assumption we show that for any given noise rate $c < 1$, there exists a scheme that correctly decodes a $(1-c)$-fraction of the stream sent so far with high probability, and moreover, the scheme is efficient. Furthermore, if the noise rate exceeds c, the scheme aborts with high probability. We also show that no constant-rate authentication scheme recovers more than a $(1-c)$-fraction of the stream sent so far with non-negligible probability,

⋆ Supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 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 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 author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.
⋆⋆ Supported in part by NSF Award 1038578.

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part II, LNCS 8043, pp. 258–276, 2013.

thus the relation between the noise rate and recoverable fraction of the stream is tight, and our scheme is optimal.

Our techniques also apply to the task of interactive communication (two-way communication) over a noisy channel. In a recent paper, Braverman and Rao [STOC 2011] show that any function of two inputs has a constant-rate interactive protocol for two users that withstands a noise rate up to 1/4. By assuming that the parties share a secret random string, we extend this result and construct an interactive protocol that succeeds with overwhelming probability against noise rates up to 1/2. We also show that no constant-rate protocol exists for noise rates above 1/2 for functions that require two-way communication. This is contrasted with our first result in which computing the "function" requires only one-way communication and the noise rate can go up to 1.

Keywords: data stream, private codes, adversarial noise, authentication, tree codes, interactive communication.

1 Introduction

The tasks of *error-correction* and of *authentication* are well studied in the literature. In both cases, a sender (Alice) wishes to send a message over a one-way, noisy channel to a receiver (Bob). To do so, Alice produces a longer, redundant message and sends it over the channel. The added redundancy helps Bob in recovering the original message if possible, or aborting otherwise. The overhead of this process is the amount of redundancy added to each message; in this work we focus on *constant-rate* schemes, i.e., schemes in which the transmitted message is at most constant-times longer.

Interestingly, in all known authentication schemes (and in many of the error-correction codes) there are two important assumptions: (1) the message to be communicated has a given length n and (2) the message is fully known to the sender in advance. These two assumptions don't hold anymore when the information to be transmitted is in the form of a *data stream*, which is a long, possibly infinite, sequence of symbols $x_1, x_2, \ldots$ over some alphabet Σ, where each x_i arrives at the sender's end at time i and is unknown beforehand.

In this paper, we investigate the question of transmitting data streams over an adversarially noisy channel. Within this framework we consider two related questions, namely, error-correction and authentication of data streams. Loosely speaking, in error-correction schemes, the receiver decodes the correct message as long as the noise level is below some threshold (but possibly outputs a wrong message if the noise exceeds that threshold). In authentication schemes, the receiver's task is to indicate whether or not the received (decoded) message is indeed the one sent to him. To see the relation between these two tasks note that if the corruption level of an adversary is guaranteed to be lower than the threshold, any error-correction guarantees that the receiver decodes the original message. However, while no constant-rate error-correction scheme can withstand a noise level higher than 1/2, this is not the case for authentication schemes that

are capable of indicating a change in the message even when the adversary has a full control of the channel. On the other hand for the task of authentication, it is generally assumed that the parties pre-share a secret key.

Standard error-correction and authentication methods do not apply directly to the model of data streams. The straightforward method to perform error-correction (or authentication) of a data stream is to cut the stream into chunks and separately encode each chunk. The problem now is that while the adversary is limited to some *global* noise rate, there is no restriction on the noise level of any local part of the stream. Specifically, the adversary can corrupt a single chunk in its entirety (while not exceeding the global amount of allowed noise), and cause Bob to decode this chunk in a wrong way. Even if this event is noticed by Bob since the chunk fails the authentication, the information carried within this chunk is lost unless Bob requests a retransmission of that chunk, i.e., unless the communication is interactive. The same problem exists (with high probability) when the noise is random rather than adversarial, given that the stream is long enough or infinite.

A possible mitigation to the above is to increase the chunks' size. This, however, has an undesirable side effect—Bob needs to wait until receiving a complete chunk in order to decode and authenticate it. This means that the information received in the very recent bits is inaccessible to Bob until the chunk is completely received. Our goal is thus, to construct a constant-rate scheme that can withstand a constant fraction of errors (globally) and still guarantee the correct decoding and authenticity of the information received so far. To the best of our knowledge, no such solution is known.

1.1 Our Results

In this work we construct optimal encoding schemes for both interactive and non-interactive (streaming) communication, and show a dramatic difference between these two cases in the following sense. For each case, we show an upper bound on the noise rates that allow a successful constant-rate communication, and construct a protocol that achieves the bound. Interestingly, the bound for one-way communication is different from the interactive one.

Specifically, our result for one-way communication is a constant-rate coding scheme for data streams that withstands noise rates of less than $1/2$. Informally, as long as the global noise rate up to some time n does not exceed some parameter $c < 1/2$, a fraction of $1 - 2c$ of the stream sent up to time n can be recovered (see Section 4). For constant-rate schemes, it is clear that $c < 1/2$ is a hard limit and no scheme can succeed when the noise is higher. In order to achieve schemes that withstand higher noise rates we must relax the model and give the users more resources. Indeed, with the use of shared randomness (i.e., a shared secret key) we can break the $c = 1/2$ barrier. To emphasize the fact that the parties are allowed to share a secret key, we refer schemes in this model as *authentication schemes* rather than error-correction schemes, based on the relation of these two tasks mentioned above (codes that assume a private shared key are also known as *private codes* [16], see Related Work).

This leads to our first main result: we construct a constant-rate authentication scheme for data streams sent over a noisy (possibly adversarial) channel. For any constant fraction of noise c less than 1, our scheme succeeds in decoding at least a $(1-c)$-fraction of the stream so far, with high probability. The decoded part is always the *prefix* of the stream. The decoded prefix is authenticated, meaning that there is only a negligible probability that the scheme outputs a different string. Furthermore, our scheme is *efficient*. More formally (see formal theorems in Section 5), we show that for any noise rate $0 \leq c < 1$ and small constant $\varepsilon > 0$:

- There exists an efficient constant-rate scheme that, at time n, decodes a prefix of length at least $(1-c)n - \varepsilon n$ of the stream sent so far.
- Any constant-rate protocol that decodes a prefix of length $(1-c)n + \varepsilon n$ succeeds with probability at most $2^{-\Omega(\varepsilon n)}$ in the worst case.

Our scheme is unconditionally secure and does not make any (cryptographic) assumptions, other than pre-sharing a secret random string. The amount of randomness utilized by the scheme grows with the message length, and can be unbounded if the data stream is infinite. However, if we only consider a computationally bounded adversary, the required amount of randomness is relatively small (polynomial in the security parameter). With the aid of a pseudo-random generator, the parties only need to pre-share a small seed, from which they generate randomness at will. Moreover, such a solution scales to the multiparty case by a simple public-key infrastructure construction. Each user generates a pair of a public and a secret key, and any pair of users perform Diffie-Hellman key-exchange [6] to obtain a secret shared authentication-key used as the pseudo-random generator's seed.

We apply the same techniques used in our streaming-authentication scheme onto the task of *interactive communication* to get our second main result. In the interactive communication scenario, two parties perform an arbitrary interactive protocol over a noisy channel, while keeping the amount of exchanged data only a constant factor more than an equivalent protocol for a noiseless channel (i.e., the encoding is constant-rate). This question was initially considered for both random and adversarial noise by Schulman [22,23,24] who showed a constant-rate encoding scheme that copes with a noise rate of up to 1/240, and recently revisited by Braverman and Rao [5] who showed how to deal with noise rates less than 1/4. In addition, Braverman and Rao show that 1/4 is the highest error rate any protocol can withstand, as long as the protocol defines whose turn it is to speak at every round regardless of the observed noise. The fascinating open question left by the work of Braverman and Rao is whether other methods could extend the 1/4 bound.

In this work we improve the bound obtained by [5] by allowing the parties to pre-share a secret key. Specifically, we show how to convert any interactive protocol (for noiseless channel) into a constant-rate protocol that withstands any adversarial noise level smaller than 1/2, given pre-shared randomness. We also show that for higher noise rates, no constant-rate interactive protocol exists

for tasks that depend on inputs of *both* parties. Similarly to previous results for interactive communication with adversarial noise [24,5,9], our decoding scheme is inefficient. Very recently, Brakerski and Kalai [2] showed how to augment previous results of interactive communication protocols and achieved *efficient* schemes that withstand adversarial noise (the computation efficiency was further improved by Brakerski and Naor [3] to $O(N \log N)$). Note that the bounds (on adversarial noise) obtained by [2] are improved by our work as well, since we improve the bounds of the underlying schemes used by [2].

1.2 Our Methods

The Blueberry Code. The main ingredient of our construction is an error-detection code we name the *Blueberry code*[1]. The Blueberry code uses the shared randomness in order to detect corruptions made by the channel, and marks them as *erasures*. One can think about this code as a weak message authentication code (MAC) that authenticates each symbol separately with a constant probability (see [11] for a formal definition of MAC). To this end, each symbol of the input alphabet Σ is randomly and independently mapped to a larger alphabet Γ (the channel alphabet). This means that only a small subset of the channel alphabet is meaningful and the other symbols serve as "booby-traps". Since each symbol is encoded independently, any corruption is caught with constant probability $\frac{|\Sigma|-1}{|\Gamma|-1}$ and marked with a special sign $\perp$ to denote it was deleted by the channel. Most of the corruptions made by an adversary become erasures and only a small fraction (arbitrarily small, controlled by the size of $|\Gamma|$) turns into errors.

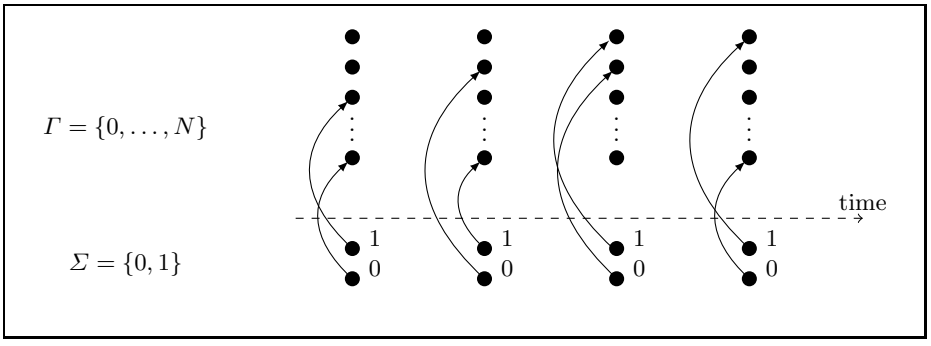

Fig. 1. A demonstration of the Blueberry code: at any given time each symbol in Σ is randomly mapped to a symbol of Γ. Symbols of Γ with no incoming arrow are "booby-traps", which serve to detect corruptions.

[1] The name of the Blueberry code is inspired by the children's book "The case of the hungry stranger" [1] in which a blueberry pie is gone missing, and the thief (who turns out to be the dog) is identified by his big blue grin.

The main insight that leads to our results is the different ways error correction codes deal with errors and erasures. We observe that, in terms of Hamming distance, the impact of a single error is twice as harmful as a single erasure. Indeed, assume that the Hamming distance of two strings, x and y, is m. Then if x was communicated but y is decoded it means that at least $m/2$ errors have occurred, or alternatively, at least m erasures. More generally, assuming we decode by minimizing the Hamming distance, then our decoding fails if the number of errors e and the number of erasures d satisfy $2e + d \geq m$.

Combining Blueberry Codes and Tree Codes. The second ingredient of our work is encoding via *tree codes* [24], an online encoding that has a "self-healing" property: when decoding a stream at time n, the tree will decode correctly up to a particular time t such that the stream suffix between times t and n is the longest suffix in which the error rate is high. This means, for instance, that even if all the transmissions until some time t' were corrupted (and thus the decoding failed at those times), if the noise rate up to time $n > t'$ is low enough, not only can we decode between t' and n, but we will also be able to decode the *entire* stream up to time n.

Encoding via both a tree code and a Blueberry code immediately gives a streaming authentication method: the Blueberry code prevents the adversary from corrupting too many transmissions without being noticed, and given that the noise level is low enough, the tree code correctly decodes a prefix of the stream whose length is determined by the average noise level up to that time.

Efficient Constructions. The only caveat of the above construction is that tree code decoding is not necessarily efficient and may be in the worst case exponential in the length of the received transmission. We obtain an efficient authentication scheme by splitting the stream into small segments and repeatedly sending random segments of the history. That way, even if some part of the transmission was changed by the channel, the same information will keep being retransmitted at random future times, and eventually (with high probability) will be received at the other side intact.

Roughly speaking, we use $n/\log n$ tree codes to encode chunks of the stream (each of length roughly $\log n$). Note that as n grows, so does the number of the trees in use, and the expected depth of each tree. At each time step, we randomly select one of the $n/\log n$ trees and transmit the next label of the path defined by the corresponding chunk of the stream. For most of the trees, the expected number of labels transmitted is $\Theta(\log n)$, and the decoding of the specific chunk succeeds except with polynomially small probability. Since each tree code is used to encode a word of length $O(\log n)$, the decoding can be performed efficiently by an exhaustive search.

1.3 Other Related Works

The works of Even, Goldreich and Micali [7] and Gennaro and Rohatgi [10] consider authentication of data streams, however the focus of these schemes is not

only to authenticate the message but also to prevent the sender from denying having signed the information. These constructions rely on cryptographic primitives such as one-time signatures. Another related line of research [21,19,12] pursues authentication of streams over *lossy* channels, usually in the multicast setting.

Coding schemes that assume the parties pre-share some randomness (also known as *Private Codes* [16]) first appeared in [25], and were greatly analyzed since. The main advantage of such codes is that they can deal with *adversarial noise*, rather than a random noise. Langberg [16] considers private codes for adversarial channels that approach Shannon's bound and require only $O(\log n)$ randomness for block size n, as well as an $\Omega(\log n)$ lower bound for the needed 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 [26]. Explicit constructions with $o(n)$ randomness are yet unknown (see [26]).

Error correction codes for *computationally bounded* noise models were first addressed by Lipton [17] who constructs error-correction codes given pre-shared randomness and later considered by Micali, Peikert, Sudan and Wilson [18] who only assume sharing a short public-key, and recently by the surprising result of Guruswami and Smith [13] who assume no shared setup between the users. Locally Decodable codes with constant-rate in the public-key model were introduced by Hemenway and Ostrovsky [14] and later improved by Hemenway, Ostrovsky, Strauss and Wootters [15].

2 Preliminaries, Model and Definitions

We denote the set $\{1, 2, \ldots, n\}$ by $[n]$, and for a finite set Σ we denote by $\Sigma^{\leq n}$ the set $\cup_{k=1}^{n} \Sigma^k$. The Hamming distance $\Delta(x, y)$ of two strings $x, y \in \Sigma^n$ is the number of indices i for which $x_i \neq y_i$. Throughout the paper, log() denotes the binary logarithm (base 2) and ln() denotes the natural logarithm (base e).

Shared Randomness Model. We assume the following *shared-randomness model*. The legitimate users (Alice and Bob) have access to a random string R of unbounded length, which is unknown to the adversary (Eve). Protocols in this model are thus *probabilistic*, and are required to succeed with high probability over the choice of R. We assume that all the randomness comes from R and that for a fixed R the protocols are deterministic.

Tree Codes. A d-ary *tree code* [24] over alphabet Σ is a rooted d-regular tree of arbitrary depth N whose edges are labeled with elements of Σ. For any string $x \in [d]^{\leq N}$, a d-ary tree code $\mathcal{T}$ implies *an encoding* of x, $\mathsf{TCenc}_{\mathcal{T}}(x) = w_1 w_2 .. w_{|x|}$ with $w_i \in \Sigma$, defined by concatenating the labels along the path defined by x, i.e., the path that begins at the root and whose i-th node is the x_i-th child of the $(i-1)$-th node. We usually omit the subscript $\mathcal{T}$ when the tree is clear from the context. Note that tree code encoding is *online*: to communicate $\mathsf{TCenc}(x\sigma)$ where $\sigma \in [d]$ given that $\mathsf{TCenc}(x)$ was already communicated, we only need to send one symbol of Σ. Hence, if $|\Sigma| = O(1)$ the encoding scheme has a constant rate.

For any two paths (strings) $x, y \in [d]^{\leq N}$ of the same length n, let ℓ be the longest common prefix of both x and y. Denote by $anc(x, y) = n - |\ell|$ the distance from the n-th level to the least common ancestor of paths x and y. A tree code has distance α if for any $k \in [N]$ and any distinct $x, y \in [d]^k$, the Hamming distance of $\mathsf{TCenc}(x)$ and $\mathsf{TCenc}(y)$ is at least $\alpha \cdot anc(x, y)$.

For a string $w \in \Sigma^n$, decoding w using the tree code $\mathcal{T}$ means returning the string $x \in [d]^n$ whose encoding minimizes the Hamming distance to the received word, namely,

$$\mathsf{TCdec}_{\mathcal{T}}(w) = \operatorname*{argmin}_{x \in [d]^n} \Delta(\mathsf{TCenc}_{\mathcal{T}}(x), w).$$

A theorem by Schulman [24] proves that for any d and $\alpha < 1$ there exists a d-ary tree code of unbounded depth and distance α over alphabet of size $d^{O(1/(1-\alpha))}$. However, no efficient construction of such a tree is yet known. For a given depth N, Peczarski [20] gives a randomized construction for a tree code with $\alpha = 1/2$ that succeeds with probability at least $1 - \epsilon$, and requires alphabet of size at least $d^{O(\sqrt{\log \epsilon^{-1}})}$. Braverman [4] gives a sub-exponential (in N) construction of a tree code, and Gelles, Moitra and Sahai [9] provide an efficient construction of a randomized relaxation of a tree code of depth N, namely a *potent tree code*, which is powerful enough as a substitute for a tree code in most applications.

Communication Model. Our communication model consists of a channel $ch : \Sigma \to \Sigma$ subject to corruptions made by an adversary (or by the channel itself). The noise model is such that any symbol σ sent through the channel can turn into another symbol $\tilde{\sigma} \in \Sigma$. It is not allowed to insert or delete symbols. For all of our applications we assume that one symbol $\sigma_i \in \Sigma$ is sent at any time slot i.[2] We say that the adversarial *corruption rate* is c if for n transmissions, at most cn symbols were corrupted.

3 The Blueberry Code

Definition 3.1. *For $i \geq 1$ let $B_i : [L+1] \to [L+1]$ be a random and independently chosen permutation. The* Blueberry code *maps a string x of arbitrary length n to*

$$B(x) = B_1(x_1)B_2(x_2)\cdots B_n(x_n).$$

We denote such a code as $B : [L+1]^* \to [L+1]^*$.

We use the Blueberry code in the shared-randomness model where the legitimate parties share the random permutations B_i, unknown to the adversary (these kind of codes, determined by a random string unknown to the channel are referred to as *private codes* by [16]). Although B_i is a permutation on $[L+1]$, we actually

[2] The channel time slots need not correspond with the times in which stream symbols are received. I.e, it is possible that between the arrival of stream elements x_i and x_{i+1}, several channel-symbols are transmitted.

use it to encode strings over a smaller alphabet $[S+1]$ with $S < L$; that is, we focus on the induced mapping $B : [S+1]^* \to [L+1]^*$. The adversary does not know the specific permutations B_i, and has probability of at most S/L to change a transmission into a symbol whose pre-image is in $[S+1]$.

Definition 3.2. *Assume that at some time i, $y_i = B_i(x_i)$ is transmitted and $\tilde{y}_i \neq y_i$ is received. If $B_i^{-1}(\tilde{y}) \notin [S+1]$, we mark the transmission as* an erasure *(specifically, the decoding algorithm outputs $\perp$); otherwise, this event is called* an error.

Corollary 3.3. *Let $x \in [S+1]^n$ and assume $B(x)$ is communicated over a noisy channel. Every symbol altered by the channel will cause either an* error *with probability S/L, or an* erasure *with probability $1 - S/L$.*

Assuming $S \ll L$, most of the corruptions done by the channel are marked as erasures, and only a small fraction of the corruptions percolate through the Blueberry code and cause an error.

Lemma 3.4. *Let $S, L \in \mathbb{N}$ be fixed and assume a Blueberry code $B : [S+1]^* \to [L+1]^*$ is used to transmit a string $x \in [S+1]^n$ over a noisy channel. For any constant $0 \leq c \leq 1$, if the channel's corruption rate c, then with probability $1 - 2^{-\Omega(n)}$ at least a $(1 - 2\frac{S}{L})$-fraction of the corruptions are marked as erasures.*

Proof. Denote by z_i the random variable which is 1 if the i-th corrupted-transmission is marked as an erasure and 0 otherwise. These are independent Bernoullis with probability $1 - \frac{S}{L}$. Let $Z = \sum_i z_i$ and note that $\mathbb{E}[Z] = cn(1 - \frac{S}{L})$. By Chernoff-Hoeffding inequality,

$$\Pr_R \left[\frac{1}{n} \sum_i z_i < c\left(1 - 2\tfrac{S}{L}\right) \right] < e^{-2n(cS/L)^2}.$$

Corollary 3.5. *Let $S, L \in \mathbb{N}$ be fixed. If out of n received transmissions, cn were marked as erasures by a Blueberry code $B : [S+1]^* \to [L+1]^*$, then except with probability $2^{-\Omega(n)}$ over the shared randomness, the adversarial corruption rate is at most $c/(1 - 2\frac{S}{L})$.*

We will use the Blueberry code concatenated with another (outer) code that is less sensitive to erasures than to errors. From the outer code's point of view, this effectively increases the channel's "error rate resilience" from $1 - 2c$ to $1 - c(1 + S/L)$. The construction of the code B from independent B_i's allows us to encode and decode each x_i independently, which is crucial for on-line applications in which the message x to be sent is not fully known in advance.

4 Error Correction of Data Streams

Before we reach our main result, we begin with a simple, non-efficient, constant-rate error-correction scheme for data streams that withstands noise $c < 1/2$

and decodes a prefix of length $1 - 2c$ of the stream sent so far. The scheme is obtained by simply encoding the stream via a tree code $\mathcal{T}$ with large enough distance parameter $\alpha \in (0, 1)$ and a constant-size alphabet, which depends on α.

Theorem 4.1. *For any constants $c < 1/2$ and $\varepsilon > 0$ there exists a constant-rate error-correction scheme for data stream $x_1, x_2, \ldots$ such that at any given time n the receiver outputs a string $x'_1, x'_2, \ldots, x'_n$, and if the noise rate until time n is at most c, then*

$$x'_1, x'_2, \ldots, x'_{(1-2c)n-\varepsilon n} = x_1, x_2, \ldots, x_{(1-2c)n-\varepsilon n}$$

that is, a prefix of the stream of length at least $(1-2c)n-\varepsilon n$ is correctly decoded.

Proof. Assume Alice encodes each stream symbol using $\mathsf{TCenc}_{\mathcal{T}}()$ using some tree code $\mathcal{T}$ whose parameters we fix shortly.

For a specific time n, consider a string $\tilde{x} \in \{0,1\}^n$, such that $anc(x, \tilde{x}) \geq (2c + \varepsilon)n$. Due to the tree distance property, the Hamming distance between $\mathsf{TCenc}(\tilde{x})$ and $\mathsf{TCenc}(x)$ is at least $\alpha(2c + \varepsilon)n$. Assume Eve causes e errors, a maximal-likelihood decoding will prefer x over $\tilde{x}$ as long as $\lfloor \alpha(2c + \varepsilon)n \rfloor > 2e$. Since Eve's corruption rate is limited to c, we know that $e \leq cn$. By setting $\alpha > \frac{2c}{2c+\varepsilon}$ we guarantee that $\alpha(2c + \varepsilon)n > 2e$, and Bob decodes a string x' such that $anc(x, x') < (2c + \varepsilon)n$ with certainty. □

5 Perpetual Authentication

Sending a data stream over a noisy channel is not a simple task, especially when the noise model is adversarial. Our goal is to design an encoding and decoding scheme such that the encoding has a constant rate and the decoding recovers the encoded transmitted stream, or else aborts. Furthermore, we wish an "authentication" guarantee, that is, if the decoding scheme did not abort, it decodes the *correct* data with high probability (note that the probability that the scheme aborts potentially differs from the probability that the decoding scheme outputs incorrect data). The amount of recoverable data depends on the noise and the goal is to output (and authenticate) the longest possible prefix of the stream, given a constant corruption rate.

Definition 5.1. *A* $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication Scheme *with constant rate r is an encoding $e : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^r$ that encodes a stream $x_1, x_2, \ldots$ into a stream $y_1 = e(x_1, R)$, $y_2 = e(x_1 x_2, R)$, …, $y_i = e(x_1 \cdots x_i, R)$, and a decoding $d : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ such that the following holds. For any n, and for any adversary $Adv(x_1 \cdots x_n, y_1 \cdots y_n) = y'_1 \cdots y'_n$, either $d(y'_1 \cdots y'_n, R) = x'_1 x'_2 \cdots x'_n$ or $d(y'_1 \cdots y'_n, R) = \perp$, and if at most $c(n)$ transmissions were corrupted, then*

1. *the scheme aborts with probability at most $\kappa(n)$,*

$$\Pr_R[d(y'_1 \cdots y'_n, R) = \perp] < \kappa(n).$$

2. *if not aborted, the probability to decode an incorrect $\gamma(n)$-prefix of the stream is at most $\kappa(n)$,*

$$\Pr_R[d(y'_1 \cdots y'_n, R) \neq \perp \ \wedge \ x'_1 \cdots x'_{\gamma(n)} \neq x_1 \cdots x_{\gamma(n)}] < \kappa(n).$$

Eve is given both the raw stream and the channel transmissions, however she does not know the shared random string R used as the secret authentication key. It is desired that as long as Eve corrupts only a small fraction of the transmissions, Bob will be able to correctly decode a prefix of the stream, or otherwise be aware of the adversarial intervention and abort.

We show the following dichotomy: If the adversarial corruption rate is some constant c, then there exists a streaming authentication stream that decodes a prefix of at most $(1-c)$-fraction of the stream received so far. In addition, there does not exist a streaming authentication scheme that is capable of decoding a longer prefix with non-negligible probability.

Theorem 5.2. *In the shared-randomness model, for every constants c, ε such that $0 \leq c < 1$ and $0 < \varepsilon \leq (1-c)/2$ there exists a constant-rate $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(n)})$-Streaming Authentication Scheme. Moreover, there exists an* efficient *constant-rate $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-Streaming Authentication Scheme.*

For any constant $c_{th} > c$, if the adversarial corruption rate exceeds c_{th}, the schemes abort with overwhelming probability over the shared randomness.

Theorem 5.3. *Assume that a bitstream $x_1, x_2, \ldots$ is communicated using some encoding protocol with a constant rate, and assume that at time n the receiver decodes the bitstring $x'_1, \ldots, x'_n$. If the rate of adversarial corruptions is $0 \leq c \leq 1$, then for any constant $\varepsilon > 0$,*

$$\Pr[x'_1 \cdots x'_{(1-c)n+\varepsilon n} = x_1 \cdots x_{(1-c)n+\varepsilon n}] \leq 2^{-\Omega(\varepsilon n)}$$

where the probability is over the coin tosses of the decoding algorithm, assuming $\{x_i\}$ are uniformly, independently distributed.

We now prove Theorem 5.3 and then construct the protocols guaranteed by Theorem 5.2

Proof. Consider an adversary that, starting at time $(1-c)n$, corrupts all the transmissions. It is easy to verify that the corruption rate is c. Clearly, from time $(1-c)n$ and on, the effective capacity of the channel is 0. This means that the decoder has no use of transmissions of times $\geq (1-c)n$ and he decodes only using transmissions received up to time $(1-c)n$. However, due to the streaming nature of the model, transmissions at times $< (1-c)n$ depend only on $x_1, \ldots, x_{(1-c)n}$ (the suffix of the stream is yet unknown to the sender). The receiver has no information about any bit x_i with $i > (1-c)n$ and his best strategy is to guess them. The probability to correctly guess the last εn bits is at most $2^{-\lfloor \varepsilon n \rfloor}$. □

In order to construct a streaming authentication scheme, we use two concatenated layers of online codes. The inner code is a Blueberry $B : [S+1]^* \to [L+1]^*$

code with constant S, L, and the outer code A is an online code that allows a prefix decoding in the presence of errors and erasures. The entire process can be described by

$$(x_1, \ldots) \xrightarrow{A} (y_1, \ldots) \xrightarrow{B} (z_1, \ldots) \xrightarrow{channel} (\tilde{z}_1, \ldots) \xrightarrow{B^{-1}} (\tilde{y}_1, \ldots) \xrightarrow{A^{-1}} (\tilde{x}_1, \ldots)$$

We begin with a simple and elegant construction which, although not efficient, demonstrates the power of the Blueberry code.

Proposition 5.4. *Let c, ε be constants $0 \leq c < 1$, $0 < \varepsilon \leq (1-c)$ and let $A = \mathsf{TCenc}()$ be an encoding using a binary tree code and B a Blueberry code with constant parameters determined by c, ε. The concatenation of A and B is a $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(n)})$-streaming authentication scheme.*

Proof. Assume that in order to encode the bitstream $x_1, x_2, \ldots$, we use a binary tree code over alphabet $[S+1]$ with distance α to be determined later, concatenated with a Blueberry-code $B : [S+1]^* \to [L+1]^*$. We show that if at time n we decode a string $\tilde{x}_1 \cdots \tilde{x}_n$ whose prefix $\tilde{x}_1 \cdots \tilde{x}_{(1-c-\varepsilon)n}$ differs from $x_1 \cdots x_{(1-c-\varepsilon)n}$, then the corruption rate was larger than c.

For a specific time n, consider a string $\tilde{x} \in \{0,1\}^n$, such that $anc(x, \tilde{x}) \geq (c+\varepsilon)n$. Due to the tree distance property, the Hamming distance between $\mathsf{TCenc}(\tilde{x})$ and $\mathsf{TCenc}(x)$ is at least $\alpha(c+\varepsilon)n$. Assume Eve causes d erasures and e errors, a maximal-likelihood decoding will prefer x over $\tilde{x}$ as long as $\lfloor \alpha(c+\varepsilon)n \rfloor > 2e + d$.

If Eve's corruption rate is limited to c, Lemma 3.4 implies that with overwhelming probability at most $2cnS/L$ of these corruptions become errors and the rest are marked as erasures. Setting $\alpha > \frac{c}{c+\varepsilon}(1 + \frac{2S}{L})$ we guarantee that $\alpha(c+\varepsilon)n > 2 \cdot 2cnS/L + cn(1 - 2S/L)$,[3] thus Bob decodes with overwhelming probability a string $\tilde{x}$ such that $anc(x, \tilde{x}) < (c+\varepsilon)n$, as claimed.

Note that the actual fraction of adversarial corruptions can be estimated out of the number of erasures marked by the Blueberry code. We abort the decoding if at a specific time n the number of erasures exceeds cn. Lemma 3.4 guarantees that if the adversary corrupts more than a $c/(1 - \frac{2S}{L})$-fraction of the transmissions, she will cause at least cn erasures, except with negligible probability. Choosing L such that $(1 - \frac{2S}{L}) \geq \frac{c}{c_{th}}$ completes the proof for the non-efficient case of Theorem 5.2. □

We note that although in the above proof we require ε to be constant, for the case of $c = 0$ (i.e., when the channel is not inherently noisy) we can let ε be smaller. For instance, if we let $\varepsilon = \kappa/n$ for a security parameter κ, the scheme is comparable to a (non-streaming) authentication scheme with the same security parameter: in order to change even a single bit in a prefix of length n, after $n + \kappa$

[3] It is required to have $\alpha < 1$, thus the choice of (the constant) L should depend on ε and c, specifically, $L > 2S\frac{c}{\varepsilon}$. Also note that S depends on α, however L is independent of both. For a fixed value of α (and $S = d^{O(1/(1-\alpha))}$) there is always a way to choose a constant L that satisfies the conditions.

symbols were transmitted, the adversary must change at least $\alpha\kappa/2$ transmissions, and will be caught except with probability $2^{-\Omega(\kappa)}$. Since the above holds for any time n, we get a perpetual authentication of the stream.

The case where $c > 0$ has a meaning of communicating over a noisy channel (regardless of the adversary). The users do not abort the authentication scheme although they know the message was changed by the channel. Instead, the scheme features both error-correction and authentication abilities and the parties succeed to recover (a prefix of) the original message with high probability.

5.1 Efficient Streaming Authentication

We now complete the proof of Theorem 5.2 by defining an efficient randomized code A_{eff} for prefix-decoding in the presence of errors and erasures. The protocol partitions the stream into words of logarithmic size and encodes each using a tree code. At any time n, one of the $O(n/\log n)$ words is chosen at random and its next encoded symbol is transmitted. The value n increases as the protocol progresses which means that the length of each encoded word increases as well. This however causes no problem: each word is encoded by a tree code (rather than, say, a block code), which is performed in an online manner without assuming knowledge of the word's length. Decoding can be performed efficiently by an exhaustive search since each word is of logarithmic length in the current time n. We note that the parties hold the entire stream in their memory throughout the protocol, which is different from the common practice of streaming algorithms in which there is only a single party (rather than two) which aim to compute some statistics of the stream using poly-logarithmic memory.

Proposition 5.5. *For any constants $0 \le c < 1$, $0 < \varepsilon \le (1-c)/2$ and a constant $c_1 > 0$, there exist efficient constant-rate encoding and decoding scheme such that, for any set of infinite strings $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ the following holds for any sufficiently large time n except with polynomially small probability in n. If the corruption rate at time n is at most c then the scheme correctly decodes a prefix of length $c_1 \log n$ of each one of the strings $\mathbf{x}^k$ with $k \in \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$. Moreover, up to time n the encoding scheme assumes knowledge of only strings $\mathbf{x}^k$ with $k \le n/\log n$.*

In the full version of this paper [8] we show that a protocol that satisfy Proposition 5.5 can be obtained by concatenating Protocol 1 (see below) with a Blueberry code $B : [S+1]^* \to [L+1]^*$. We show that with high probability, $\Theta(\log n)$ symbols of $\mathsf{TCenc}(\mathbf{x}^k)$ are transmitted by time n for every k in the range $K_n \triangleq \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$. Moreover, at least a constant fraction of these transmissions were not corrupted by the adversary. Therefore, we can use Proposition 5.4 to decode a prefix of length $O(\log n)$ of each of the codewords indexed by K_n, with high probability.

Finally, in Appendix A we show how to split the stream $x_1, x_2, \ldots$ into words $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$, so that the prefix $x_1, \ldots, x_{(1-c-\varepsilon)n}$ completely appears in the $O(\log n)$-prefix of strings $\{\mathbf{x}^k\}$ with $k \in K_n$. This gives an efficient $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-authentication scheme and completes the proof of Theorem 5.2.

Let $0 \leq c < 1$ and $0 < \varepsilon < (1-c)/2$ be fixed parameters of the protocol. Let c_0, c_1 be some constants which depend on c and ε. Let $\mathcal{T}$ be a tree code over alphabet $[S+1]$ with distance α to be set later.

A_{eff} **Encoding:** For every $k > 0$ set $count_k = 0$.
At any time $n > 1$, repeat the following process for $j = 1, 2, \ldots, c_0$:
(a) randomly choose $k \in \{1, \ldots, \lfloor n/\log n \rfloor\}$.
(b) set $count_k = count_k + 1$.
(c) transmit $y_{n,j} \in [S+1]$, the next symbol of the encoding of $\mathbf{x}^k$ using $\mathcal{T}$, that is, the last symbol of

$$\mathsf{TCenc}(\mathbf{x}_1^k \cdots \mathbf{x}_{count_k}^k) = \mathsf{TCenc}(\mathbf{x}_1^k \cdots \mathbf{x}_{count_k-1}^k) \circ y_{n,j}.$$

A_{eff} **Decoding:** For every $(i,j) \in \mathbb{N} \times [c_0]$ we denote by $\mathsf{ID}(i,j)$ the identifier k of the string $\mathbf{x}^k$ used at iteration (i,j). For each time n, mark all the transmissions $y_{i,j}$ with $i < \varepsilon n/4$ as erasures, and decode $\mathbf{x}^k$ for $\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil \leq k \leq \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil$:
let $Y_k = \{(i,j) \mid \mathsf{ID}(i,j) = k\}$. Decode the received string indexed by Y_k. That is, set

$$\hat{\mathbf{x}}^k = \mathsf{TCdec}(y_{|_{Y_k}}),$$

where $y_{|_{Y_k}}$ is the string given by concatenating all $y_{i,j}$ with $(i,j) \in Y_k$, where $y_{i,j}$ comes before $y_{i',j'}$ if $i < i'$ or $(i = i') \wedge (j < j')$. Consider a prefix of length $c_1 \log n$ of $\hat{\mathbf{x}}^k$ and ignore the rest.

Protocol 1: An efficient protocol for communicating a logarithmic prefix of $\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \}$.

5.2 Extensions for Streaming Authentication

There are several possible extensions to the above results, which we briefly discuss here. See [8] for full details and proofs.

Efficient Streaming Authentication Scheme with Exponentially Small Error. It is possible to improve the efficient scheme of Theorem 5.2 so that it aborts with polynomially small probability, however, given that it did not abort, the probability that the decoded prefix is incorrect is *exponentially small*. More accurately, the 'trust' Bob has in the decoded string *increases* with the amount of received transmissions. Thus, except for the last fraction of the stream, the decoded stream is equal to the one sent by Alice with overwhelming probability.

Theorem 5.6. *For any $0 \leq c < 1$, $0 < \varepsilon \leq \frac{1}{2}(1-c)$ there exists an efficient $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-streaming authentication protocol that, for any time n in which the decoding procedure did not abort, for any $1 \leq \ell \leq (1-c-\varepsilon)n$ it holds that*

$$\Pr[x'_\ell \neq x_\ell] < 2^{-\Omega(n)}.$$

Decoding a Prefix Longer than $(1-c)n$. Although our scheme decodes a prefix of length at most $(1-c)n$ in the worst case, the successfully decoded prefix can be in fact longer. The worst case, as demonstrated by Theorem 5.3, happens when the adversary blocks the suffix of the transmitted stream. On the other hand, if the adversary blocks the prefix of the transmissions, then the scheme of Proposition 5.4 correctly decodes the entire stream! In fact, the protocol succeeds to decode the entire prefix for any time n that satisfies the following γ-suffix condition, if the tree distance satisfies $\alpha > \gamma$.

Definition 5.7. *For any constant $0 \leq \gamma < 1$, we say that time n satisfies the γ-suffix condition if any suffix $x_t \dots x_n$ has at most $\gamma(n-t)$ corrupted transmissions.*

Definition 5.8. *Let $c < 1$ and $\gamma \in (c, 1)$ be given. For any time n let $N_\gamma(n)$ be the latest index that satisfies the γ-suffix condition. When n is clear from the context, we denote $N_\gamma(n)$ simply as N_γ.*

The following Lemma guarantees that, for any $\gamma \in (c, 1)$ it holds that $(1 - c/\gamma)n \leq N_\gamma(n) \leq n$.

Lemma 5.9. *For every corruption rate c and constant $1 < \xi < 1/c$ there exist a time $t > (1 - \frac{1}{\xi})n$ that satisfies the $c\xi$-suffix condition.*

For a corruption rate c and any $\varepsilon > 0$, and for any time n, if the decoding algorithm did not decode up to time n, then that time n did not satisfy the suffix condition for $\gamma = c/(c+\varepsilon)$, but then, by Lemma 5.9, there must exist a time $N_\gamma > (1 - c - \varepsilon)n$ that satisfies the γ-suffix condition, and at that time the protocol correctly decoded the entire stream (up to time N_γ). Bob does not know the value of N_γ but he can estimate it by checking the number of erasures marked by the Blueberry code.

Proposition 5.10. *Bob can efficiently compute a (lower-bound) estimation N'_γ for N_γ, such that $N'_\gamma > (1 - c - \epsilon)n$ and*

$$\Pr[N'_\gamma > N_\gamma] < 2^{-\Omega(N'_\gamma - N_\gamma)}.$$

Reducing the Amount of Shared Randomness. Our schemes rely on the fact that the parties share a secret random string whose length increases with the size of the information to be communicated. This assumption is sometimes not satisfied in practical applications, especially when considering a multiparty setting in which any two parties run a separate instance of the scheme.

We can mitigate the need for a long shared randomness if the adversary is assumed to be polynomial, assuming standard cryptographic assumptions (specifically, hardness of DDH). To this end, each user generates a pair (sk, pk) of a secret and a public key, broadcasts the public key pk and keeps sk secret. When two users initiate an authentication scheme instance, they first perform a Diffie-Hellman [6] key exchange and obtain an authentication key. They both use

the authentication key as a seed to a pseudo-random-generator that generates a long random string for the authentication scheme. Under the DDH assumption, a polynomially-bounded adversary has only negligible information about the authentication key nor the generated randomness, and the authentication scheme remains secure.

6 Interactive Communication

In this section we extend our discussion to the 2-way communication model of *interactive communication*. We show that for adversarial corruption rate of $1/2$ or higher, no constant-rate protocol can compute functions that require interaction between the parties, while with the usage of the Blueberry code we show how to construct a protocol for any function assuming adversarial corruption rate below $1/2$.

Assume that Alice and Bob wish to compute some function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, where Alice holds $x \in \mathcal{X}$ and Bob holds $y \in \mathcal{Y}$ in the shared-randomness model. The computation is performed interactively: at each round, both parties communicate a message which depends on their input and previous transmissions. At the end of the computation Alice outputs $z_A \in \mathcal{Z}$ and Bob outputs $z_B \in \mathcal{Z}$, and we say that f was correctly computed if $z_A = z_B = f(x, y)$.

In the full version [8] we prove the following separation theorems,

Theorem 6.1. *For any function f which depends on both x and y, the following holds. If the adversarial corruption rate is $\frac{1}{2}$ or higher then no constant-rate interactive protocol correctly computes f with probability higher than the probability of guessing $f(x, y)$ given only the input x (or only the input y).*

Theorem 6.2. *For any constants $\varepsilon > 0$ and for any function f and inputs x, y, there exists an interactive protocol with constant overhead such that if the adversarial corruption rate is at most $c = \frac{1}{2} - \varepsilon$, the protocol outputs $f(x, y)$ with overwhelming probability over the shared random string R.*

References

1. Bonsall, C.: The case of the hungry stranger. HarperCollins (1963)
2. Brakerski, Z., Kalai, Y.T.: Efficient interactive coding against adversarial noise. In: IEEE Annual Symposium on Foundations of Computer Science, pp. 160–166 (2012)
3. Brakerski, Z., Naor, M.: Fast algorithms for interactive coding. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, pp. 443–456 (2013)
4. Braverman, M.: Towards deterministic tree code constructions. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 161–167. ACM (2012)
5. Braverman, M., Rao, A.: Towards coding for maximum errors in interactive communication. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 159–166. ACM, New York (2011)

6. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
7. Even, S., Goldreich, O., Micali, S.: On-line/Off-line digital signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
8. Franklin, M., Gelles, R., Ostrovsky, R., Schulman, L.J.: Optimal coding for streaming authentication and interactive communication. In: Electronic Colloquium on Computational Complexity (ECCC) (2012), http://eccc.hpi-web.de/report/2012/104
9. Gelles, R., Moitra, A., Sahai, A.: Efficient and explicit coding for interactive communication. In: IEEE Annual Symposium on Foundations of Computer Science, pp. 768–777 (2011)
10. Gennaro, R., Rohatgi, P.: How to sign digital streams. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 180–197. Springer, Heidelberg (1997)
11. Goldreich, O.: Foundations of cryptography. Basic applications, vol. II. Cambridge University Press, New York (2004)
12. Golle, P., Modadugu, N.: Authenticating streamed data in the presence of random packet loss. In: ISOC Network and Distributed System Security Symposium, NDSS 2001 (2001)
13. Guruswami, V., Smith, A.: Codes for computationally simple channels: Explicit constructions with optimal rate. In: IEEE Annual Symposium on Foundations of Computer Science, pp. 723–732 (2010)
14. Hemenway, B., Ostrovsky, R.: Public-key locally-decodable codes. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 126–143. Springer, Heidelberg (2008)
15. Hemenway, B., Ostrovsky, R., Strauss, M.J., Wootters, M.: Public key locally decodable codes with short keys. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) RANDOM 2011 and APPROX 2011. LNCS, vol. 6845, pp. 605–615. Springer, Heidelberg (2011)
16. Langberg, M.: Private codes or succinct random codes that are (almost) perfect. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004, pp. 325–334. IEEE Computer Society, Washington, DC (2004)
17. 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)
18. 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)
19. Miner, S., Staddon, J.: Graph-based authentication of digital streams. In: IEEE Symposium on Security and Privacy, pp. 232–246 (2001)
20. Peczarski, M.: An improvement of the tree code construction. Information Processing Letters 99(3), 92–95 (2006)
21. Perrig, A., Canetti, R., Tygar, J., Song, D.: Efficient authentication and signing of multicast streams over lossy channels. In: IEEE Symposium on Security and Privacy, pp. 56–73 (2000)
22. Schulman, L.J.: Communication on noisy channels: a coding theorem for computation. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 724–733 (1992)
23. Schulman, L.J.: Deterministic coding for interactive communication. In: STOC 1993: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, pp. 747–756. ACM, New York (1993)
24. Schulman, L.J.: Coding for interactive communication. IEEE Transactions on Information Theory 42(6), 1745–1756 (1996)

25. Shannon, C.E.: A note on a partial ordering for communication channels. Information and Control 1(4), 390–397 (1958)
26. Smith, A.: Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, pp. 395–404. Society for Industrial and Applied Mathematics, Philadelphia (2007)

APPENDIX

A Construction of $\{\mathbf{x^1}, \mathbf{x^2}, \ldots\}$

For every k, define $\mathbf{x}^k$ to be the string that contains the stream prefix $x_{t(k)}$ downto x_1 concatenated with as many zeros as needed, $\mathbf{x}^k = x_{t(k)}x_{t(k)-1}\cdots x_2x_1000\cdots$, where $t(k)$ is defined to be the minimal time such that $t(k)/\log t(k) > k$. We say $\mathbf{x}^k$ is *declared* at time $t(k)$, meaning that only from this time and on the algorithm may choose to send symbols of the encoding of $\mathbf{x}^k$. It is easy to verify that the string $\mathbf{x}^k$ is well defined at the time it is declared (the corresponding x_i's are known).

If some string $\mathbf{x}^k$ is declared at time $t(k)$ then $\mathbf{x}^{k+1}$ will be declared at time $t(k+1) \approx t(k) + \log t(k) + O(\log\log t(k))$. By setting $c_1 = 2$ we are guaranteed that, for every $\varepsilon n/4 \le \ell \le (1-c-\varepsilon)n$, x_ℓ appears in a correctly decoded $c_1 \log n$-prefix of some $\mathbf{x}^k$ with $k \in K_n$.

Lemma A.1. *If $\mathbf{x}^k$ is the latest string declared at time $i > 8$, then $\mathbf{x}^{k+1}$ is declared at time sooner than $i + 2\log i$.*

Proof. Let $f(i) = \frac{i+2\log i}{\log(i+2\log i)} - \frac{i}{\log i}$. f is monotonically increasing, and $f(8) > 1$.

Corollary A.2. *For any time $n > 8$, and any ℓ, the bit x_ℓ is within the first $2\log n$ symbols of $\mathbf{x}^{\lceil \ell/\log \ell\rceil}$. Hence, every x_ℓ with $\varepsilon n/4 \le \ell \le (1-c-\varepsilon)n$, appears in a $2\log n$-prefix of (at least) one of the strings $\{\mathbf{x}^k\}_{k\in K_n}$.*

Unfortunately, with the above choice of $\mathbf{x}^k$s, only part of the stream, namely $x_{\varepsilon n/4}, \ldots, x_{(1-c-\varepsilon)n}$, is decoded by the protocol. In order to communicate the prefix $x_1, \ldots, x_{\varepsilon n/4}$ we run another instance of the scheme guaranteed by Proposition 5.5 for the following set of infinite strings $\{\mathbf{v}^1, \mathbf{v}^2, \ldots\}$. (We explain how to combine these two instances below). Define $\mathbf{v}^k$ in the following way

$$\mathbf{v}_i^k = \begin{cases} x_1 & k = 1,\ \forall i \\ x_{1+(\ell \bmod \lceil t(k)/2\rceil+1)} & k > 1,\ i = 1 \text{ and } \mathbf{v}^{k-1}_{2\log t(k-1)} = x_\ell \\ x_{1+(\ell \bmod \lceil t(k)/2\rceil+1)} & k > 1,\ i > 1 \text{ and } \mathbf{v}^k_{i-1} = x_\ell \end{cases}$$

It is easy to verify that at time n, the string $\mathbf{v}^{\lfloor n/\log n\rfloor}$ is well defined and known to the encoder.

Lemma A.3. *For every time $n > 256/(1-c-\varepsilon)$, any bit x_ℓ with $1 \le \ell \le \varepsilon n/4$ appears in a $2\log n$-prefix of (at least) one of the strings $\{\mathbf{v}^k\}_{k\in K_n}$.*

Proof. Note that the concatenation of $O(\log n)$-prefix of the $\mathbf{v}^k$s gives a string of the form $V \triangleq x_1 x_2 \ldots x_{\lceil t(k_1)/2 \rceil} x_1 x_2 \ldots x_{\lceil t(k_2)/2 \rceil} x_1 x_2 \ldots$, and V is decoded by Protocol 1 with high probability.[4] By taking $c_1 = 2$ and recalling that $\varepsilon < (1-c)/2$, (and thus, $(1-c-\varepsilon)n/4 > \varepsilon n/4$) the length of V is lower bounded by the amount of indices in prefixes of size $2\log\frac{1}{4}(1-c-\varepsilon)n$ of $\{\mathbf{v}^{(1-c-\varepsilon)n/4}, \ldots, \mathbf{v}^{(1-c-\varepsilon)n}\}$,

$$\begin{aligned} 2\log\frac{1}{4}(1-c-\varepsilon)n \left(\frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} - \frac{\frac{1}{4}(1-c-\varepsilon)n}{\log\frac{1}{4}(1-c-\varepsilon)n} \right) \\ \geq \frac{3}{2}(1-c-\varepsilon)n - 4\frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \\ \geq (1-c-\varepsilon)n \end{aligned}$$

where the last inequality holds for $n > \frac{256}{1-c-\varepsilon}$. Consider the latest place in V where x_1 appears. If that place is at least $(1-c-\varepsilon)/4$ indices from the end of V, it is clear that $x_1 \ldots x_{(1-c-\varepsilon)/4}$ appears in the $(1-c-\varepsilon)/4$-suffix of the decoded V. For the other case, let the bit that precedes this x_1 be x_ℓ. By the way we defined $\mathbf{v}^k$ it follows that $\frac{3}{8}(1-c-\varepsilon) \leq \ell \leq \frac{1}{2}(1-c-\varepsilon)$ which means that $x_1 \ldots x_{(1-c-\varepsilon)/4}$ must appear in a prefix of size $3/4 \cdot (1-c-\varepsilon)n$ of V. Since $(1-c-\varepsilon)n/4 > \varepsilon n/4$, the claim holds. □

One cannot run Protocol 1 twice, once for $\{\mathbf{x}\}$ and once for $\{\mathbf{v}\}$. Indeed, Eve can block all the transmissions of one of the instances, thus prevent the correct decoding of the stream with probability one, while her corruption rate does not exceed $c = 1/2$. One possible solution is to set $c_1 = 4$ and interleave the transmitted data, that is, define the set $\{\mathbf{z}^1, \mathbf{z}^2, \ldots\}$ where $\mathbf{z}^k = \mathbf{x}_1^k \mathbf{v}_1^k \mathbf{x}_2^k \mathbf{v}_2^k \ldots$, etc.

Corollary A.4. *Let c, ε be constants $0 \leq c < 1$, $0 < \varepsilon \leq (1-c)/2$, and let B be a Blueberry code with constant parameters determined by c, ε. For the strings $\{\mathbf{z}^1, \mathbf{z}^2, \ldots\}$ defined above, the concatenation of A_{eff} with B is an efficient $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-streaming authentication scheme.*

[4] To be more accurate, V is a substring of the string decoded by the scheme.