Electronic Colloquium on Computational Complexity, Report No. 104 (2012)

Optimal Coding for Streaming Authentication and Interactive Communication

Matthew Franklin[*] Ran Gelles[†] Rafail Ostrovsky[‡] Leonard J. Schulman[§]

August 8, 2012

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 a shared randomness and shows how to recover (non-efficiently) a $(1-2c)$-fraction prefix of the stream sent so far, assuming the noise level is at most $c < 1/2$.

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. We prove that the decoded string is identical to the one sent with overwhelming probability, even when considering an adversarial noise model. 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, 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 are able to extend this result and construct an interactive protocol that withstands a noise rate up to $1/2$, and succeeds with overwhelming probability. 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.

[*]CS Department, UC Davis, `franklin@cs.ucdavis.edu`

[†]CS Department, UCLA, `gelles@cs.ucla.edu`

[‡]UCLA Department of Computer Science and Department Mathematics, 3732D Boelter Hall, Los Angeles CA 90095-1596, U.S. Email: `rafail@cs.ucla.edu`. 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.

[§]E&AS Division, Caltech. Email: `schulman@caltech.edu`. Supported in part by the NSF.

ISSN 1433-8092

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 consider only constant-rate scheme, that is, allowing the longer message to be at most constant-times longer than the original message.

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 may decode any other message if the noise exceeds that threshold). In authentication schemes, the receiver verifies that the decoded messages is indeed the one sent to him, and aborts if a different message was decoded. To see the relation between those two tasks note that if the corruption level of an adversary is guaranteed to be lower than the threshold, an error-correction scheme can also serve as an authentication scheme. 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, it is generally assumed that for performing authentication, the parties pre-share a secret key.

Standard error-correction and authentication methods do not apply for the model of data streams. The straightforward method is to cut the stream into chunks and perform error-correction and authentication separately on 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. Surprisingly, the bound for one-way communication is different from the interactive one.

Specifically, for one-way communication our results is an error correcting scheme for data-streams that withstands a noise rate of up to 1/2 of the transmission. 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 formal definition in Appendix E). For constant-rate schemes, it is clear that $c < 1/2$ is a hard limit and no scheme can succeed with higher noise level (for $c \geq 1/2$ the capacity of the channel in this case is 0). In order to achieve schemes that withstand higher noise rate 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 emphasis the fact the the parties are allowed to share a secret key, we refer schemes in this model as *authentication schemes* rather than error-correcting schemes, based on the relation between these two tasks as mentioned above.

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 a prefix of the stream of length at least $(1 - c)$-fraction of the stream sent so far, with high probability. The decoded prefix is authenticated, meaning that there is only negligible probability that the scheme outputs a different string than the one sent by Alice. Furthermore, our scheme is *efficient.* More formally (see formal theorems in Section 4), 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 two users perform Diffie-Hellman key-exchange [DH76] 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 for 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 [Sch92, Sch93, Sch96] 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 [BR11] 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 [BR11] 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 protocol exists for tasks which are interactive (that is, depend on the inputs of *both* the parties). Similarly to previous results for interactive communication with adversarial noise [Sch96, BR11, GMS11], our decoding scheme is inefficient. Very recently, Brakerski and Kalai [BK12] show how to augment previous results of interactive communication protocols and achieve *efficient* schemes that withstand also adversarial noise. Note that the bounds (on adversarial noise) obtained by [BK12] are improved by our work as well, since we improve the bounds of the underlying schemes used by [BK12].

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 constant probability (see [Gol04] for a formal definition of MAC). To this end, each symbol of the 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, according to the choice of $|\Gamma|$) turns into errors.

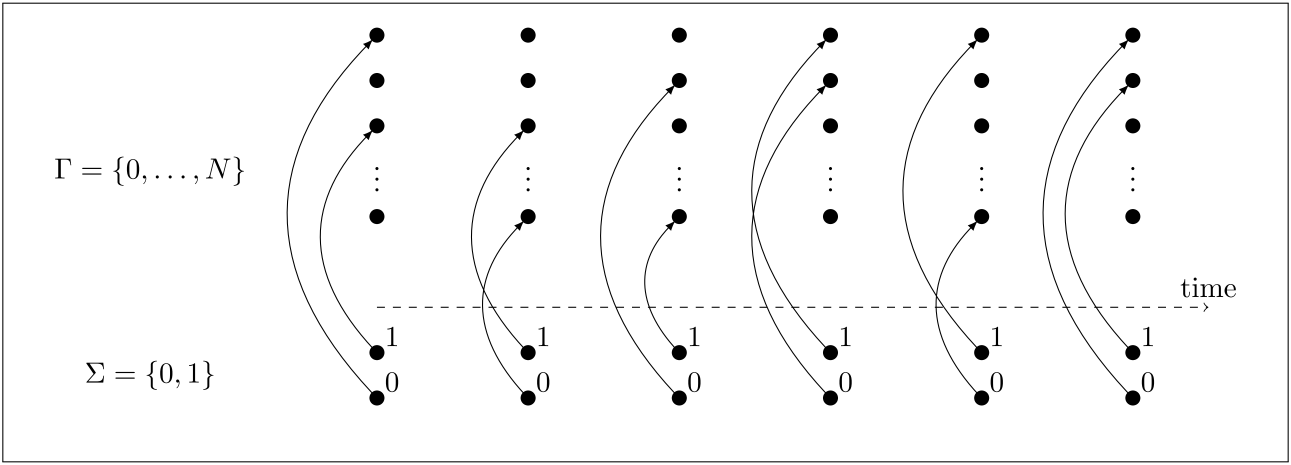

Figure 1: A demonstration of the Blueberry code: at any given time each symbol in Σ is randomly mapped to a symbol of Γ. Symbols of Γ with no incoming arrow are "booby-traps", which serve to detect corruptions.

The main insight that leads to our results is the different ways error correcting 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* [Sch96], an online encoding that has a "self-healing" property: when decoding a stream

[1]The name of the Blueberry code is inspired by the children's book "The case of the hungry stranger" [Bon63] 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.

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, we may also be able to decode the *entire* stream up to time n.

Combining tree codes encoding with the Blueberry code immediately gives a streaming authentication method: the Blueberry code prevents the adversary from corrupting too many transmissions without being noticed (causing the abortion of the protocol), and given that the protocol did not abort, the noise level is low enough for the tree code to correctly decode 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 exponential in the length of the received transmission in the worst case. To get an efficient authentication scheme we construct a specific randomized online coding, which repeatedly sends 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) received at the other side intact.

Roughly speaking, at any time n, we split the stream into words of logarithmic length in n. We encode each word using a tree code: each possible word corresponds to a node in the tree, and its encoding is the labels of the edges along the corresponding path (see formal definition below). At each time, we randomly select one of the $n/\log n$ trees and send the label of the next edge along the path. For most of the trees, the expected number of labels transmitted is $\Theta(\log n)$, and decoding of a specific word 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 exhaustive search.

In order to improve the authenticity of the received message, each word also contains a random hash of all the previous words, where the hash is of logarithmic length. The scheme aborts if any of the decoded hashes does not agree with the decoded stream. The event that the scheme does not abort and outputs an incorrect prefix of the stream received so far occurs with probability at most exponentially small in n.

1.3 Other Related work

The works of Even, Goldreich and Micali [EGM90] and Gennaro and Rohatgi [GR97] 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 [PCTS00, MS01, GM01] pursues authentication of streams over *lossy* channels, usually in the multicast setting.

Error correction codes for *computationally bounded* noise models were first addressed by Lipton [Lip94], who constructs error-correction codes given pre-shared randomness and later considered by Micali, Peikert, Sudan and Wilson [MPSW05] who only assume sharing a short public-key, and recently by the surprising result of Guruswami and Smith [GS10] 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 [HO08] and later improved by Hemenway, Ostrovsky, Strauss and Wootters [HOSW11]. Langberg [Lan04] considers error-correction codes for *adversarial channels* assuming a shared randomness. The focus of [Lan04] is showing tight bounds on the length of the shared randomness and (existentially) constructing codes with rates that nearly reach the Shannon bounds.

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 $x_i \neq y_i$, and the Hamming weight of a string is its Hamming distance from the all-zero string. 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 model known as the *shared-randomness model*. The legitimate users have access to a random string R 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 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* [Sch96] 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 ith node is the x_ith child of the (i–1)st 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 $\mathsf{TCenc}(x\sigma)$ when $\mathsf{TCenc}(x)$ 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 $anc(x, y) = n - |\ell|$ the distance 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 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 [Sch96] 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 [Pec06] 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 [Bra12] gives a sub-exponential (in N) construction of a tree-code, and the work of Gelles, Moitra and Sahai [GMS11] provides 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.

[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.

3 The Blueberry Code

Definition 3.1. *For $i \geq 1$ let $B_i : [L+1] \to [L+1]$ be a random and independent 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 [Lan04]). Although B_i is a permutation on $[L+1]$, we actually use it to encode strings over a smaller alphabet $[S+1]$ with $S < L$; that is, we focus on the 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 will be marked as erasures, and only a small fraction will percolate through the Blueberry code and cause an error.

Lemma 3.4. *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 is at most c, then with probability $1 - 2^{-\Omega(n)}$ at least a $c(1 - 2\frac{S}{L})$-fraction of the corruptions are marked as erasures.*

Proof. Denote by z_i the random variable which is 1 if the ith 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. *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 "capacity" 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 Perpetual Authentication

Sending a datastream 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 schemes 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 4.1. *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 $y_1 = e(x_1, R)$, $y_2 = e(x_1 x_2, R)$, $\ldots$, $y_i = e(x_1 \cdots x_i, R)$, and a decoding $d : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ such that for any n, and for any adversary $Adv(x_1 \cdots x_n, y_1 \cdots y_n) = y'_1 \cdots y'_n$, it holds that 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,*

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 corruptions 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 4.2. *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 $c_{th} > c$, in both these schemes, if the adversarial corruption rate exceeds c_{th}, the scheme aborts with overwhelming probability over the shared randomness.

Theorem 4.3. *Assume that a bitstream $x_1, x_2, \ldots$ is communicated using some encoding protocol with constant rate, and assume that at time n the receiver side decodes the bitstring $x'_1, \ldots, x'_n$. If the rate of adversarial corruptions is $0 \leq c \leq 1$, then for any constant $\varepsilon > 0$,*

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

where the probability is over the coin tosses of the decoding algorithm (thus, over R).

We now prove Theorem 4.3 and then construct the protocols guaranteed by Theorem 4.2

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 all these 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, 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, demonstrate the power of the Blueberry code.

Proposition 4.4. *Let c, ε be constants $0 \le c < 1$, $0 < \varepsilon \le (1-c)$ 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 $anc(x, \tilde{x}) \ge (c+\varepsilon)n$. 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 correctly decode $x_1, \ldots, x_n$ 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 $(1 - \frac{2S}{L}) \ge \frac{c}{c_{th}}$ completes the proof for the non-efficient case of Theorem 4.2. $\square$

Using a similar argument we can construct a scheme that uses only the tree-code and doesn't use the Blueberry code. This yields a constant-rate error-correcting scheme for streams, see Appendix E for details.

We note that although in the above proof we require ε to be constant, for the case of $c = 0$ we can let ε be smaller. For instance, if we let $\varepsilon = \kappa/n$ for a security parameter κ, the scheme is comparable to the (non-streaming) authentication scheme with the same security parameter. This gives the perpetual authentication: at any given time, the user is assured that except with probability $2^{-\Omega(\kappa)}$, all but the last κ received bits are the bits sent by Alice. This implies that any prefix of length n is authenticated with communication $O(n + \kappa)$.

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.

[3]Note 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 which satisfies the required condition.

4.1 Efficient Streaming Authentication Protocol

We now complete the proof of Theorem 4.2 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 fact that the current time n changes throughout the encoding process means that the length of each word increases with time, however, since each word is encoded by a tree-code (rather than, say, a block code) this causes no problem—tree code's encoding is performed "online" and does not require to know the length of the word in advance. Decoding can be performed efficiently by exhaustive search since each word is of logarithmic length in the current time n.

Proposition 4.5. *For any set of infinite strings $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ there exist efficient constant-rate encoding and decoding schemes such that, for any constants $0 \le c < 1$, $0 < \varepsilon \le (1-c)/2$ and a constant $c_1 > 0$, the following holds for any sufficiently large time n except with polynomially small probability in n. If the corruption rate is at most c then the scheme correctly decodes a prefix of size $c_1 \log n$ 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 $k \le n/\log n$.*

Let $0 \le c < 1$ 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

$$\mathsf{TCenc}(\mathbf{x}^k_1 \cdots \mathbf{x}^k_{count_k}) = \mathsf{TCenc}(\mathbf{x}^k_1 \cdots \mathbf{x}^k_{count_k - 1}) \circ y_{n,j}.$$

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 $\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil \le k \le \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil$:
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. Set $x_i = \text{Majority}_{k,r}\{\hat{\mathbf{x}}^k_r\}$ for all the k, r such that $\mathbf{x}^k_r$ is x_i, with $\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil \le k \le \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil$ and $r < c_1 \log n$.

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

In Appendix A.1 we show that Protocol 1 concatenated with a Blueberry code $B : [S+1]^* \to [L+1]^*$ satisfies the requirements of Proposition 4.5. 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 4.4 to decode a prefix of length $O(\log n)$ of each of the codewords in K_n, with high probability.

The remaining hurdle is to split the stream $x_1, x_2, \ldots$ into words $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$. In Appendix A.2 we show how to construct the set $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ such that for any time n, the entire prefix $x_1, \ldots, x_{(1-c-\varepsilon)n}$

appears in a $c_1 \log n$-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 4.2.

4.2 Extensions

There are several possible extensions to the above results, which we briefly describe here.

Efficient streaming authentication scheme with exponentially small error. It is possible to improve the efficient scheme of Theorem 4.2 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 4.6. *For any $0 \le c < 1$, $0 < \varepsilon \le \frac{1}{2}(1-c)$ 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 $1 \le \ell \le (1-c-\varepsilon)n$ it holds that*

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

See proof in Appendix B.

Decoding a prefix longer than $(1-c)n$. Although in the worst case our scheme decodes a prefix of length at most $(1-c)n$, in many situations the (successfully decoded) prefix can be longer. The worst case, as demonstrated by Theorem 4.3, 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 4.4 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 4.7. *For any constant $0 \le \gamma < 1$, 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 4.8. *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 $(1 - c/\gamma)n \le N_\gamma(n) \le n$.

Lemma 4.9. *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.*

See proof in Appendix D.

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 it means that n did not satisfy the suffix condition for $\gamma = c/(c+\varepsilon)$ (see formal proof in Appendix D), but then, by Lemma 4.9, there must exists 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 amount of erasures marked by the Blueberry code.

Proposition 4.10. *Bob can efficiently compute a (lower-bound) estimation N'_γ for N_γ, such that $N'_\gamma > (1-c-\epsilon)$ and*

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

See appendix D for proof and discussion.

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 [DH76] 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 does not have any non-negligible information about the authentication key nor the generated randomness, and the authentication scheme remains secure. See Appendix C for more details and a proof sketch.

5 Interactive Computation

In this section we extend our discussion to the 2-way communication model of *interactive communication.* We show that for adversarial change rate 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. We begin by defining the the interactive communication model.

The Interactive-Communication Model. 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 $z_A = z_B = f(x, y)$. Without loss of generality we assume the output is a single bit, $|\mathcal{Z}| = 2$.

We show the following separation theorem,

Theorem 5.1. *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 to guess $f(x, y)$ given only the input x (or only the input y).*

Theorem 5.2. *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.*

The proofs follow methods from [BR11] extended to handle channel erasures.

Proof. (**Theorem 5.1**) Assume that the protocol takes T rounds. Furthermore, recall that in our model it is assumed that at each round both parties send exactly one message.[4] Hence and without loss of generality, Alice is the sender of at most $T/2$ of the transmissions. Eve corrupts all the

[4]The proof also holds for protocols for which there exists a function Next(i) which defines, for each round i, which of the parties sends a message, and is independent of the messages sent by now (these kind of protocols are called *oblivious* in [BR11]). In that case there exists one party that communicates at most $T/2$ messages at rounds known to Eve in advance.

On the other hand, the proof does not hold for the most general model, in which the protocol adaptively determines who is next to speak, possibly according to the noise observed so far (so that the party that suffers from higher noise rate wil get more transmission slots, etc.)

transmissions originated by Alice (causing at least an erasure in each one of these transmissions). Effectively, the unidirectional channel from Alice to Bob has a zero capacity, and it cannot be that the Bob correctly computes $f(x, y)$ with probability higher than guessing $f(x, y)$ given only y. □

It is interesting to note that if f only depends on *one* of its inputs, then only 1-way communication is required and $c = \frac{1}{2}$ is no longer a limit, as discussed in Section 4.

We now construct a protocol that correctly computes any $f(x, y)$ with overwhelming probability as long as the adversarial corruption rate is $\frac{1}{2} - \varepsilon$ for $\varepsilon > 0$. To this end, we concatenate an online protocol for computing $f(x, y)$ over an adversarial noisy channel [Sch96, BR11, GMS11] with a Blueberry code.

Let us recall how to construct a constant-rate protocol for computing $f(x, y)$ over a noisy channel out of an interactive protocol π for the same task that assumes a noiseless channel [BR11]. We assume that π consists of T rounds in which Alice and Bob send a single bit according to their input and previous transmissions. Without loss on generality, we assume that Alice sends her bits at odd rounds while Bob transmits at even rounds. We can view the computation of π as a root-leaf walk along a binary tree in which odd levels correspond to Alice's messages and even levels to Bob's, see Figure 2.

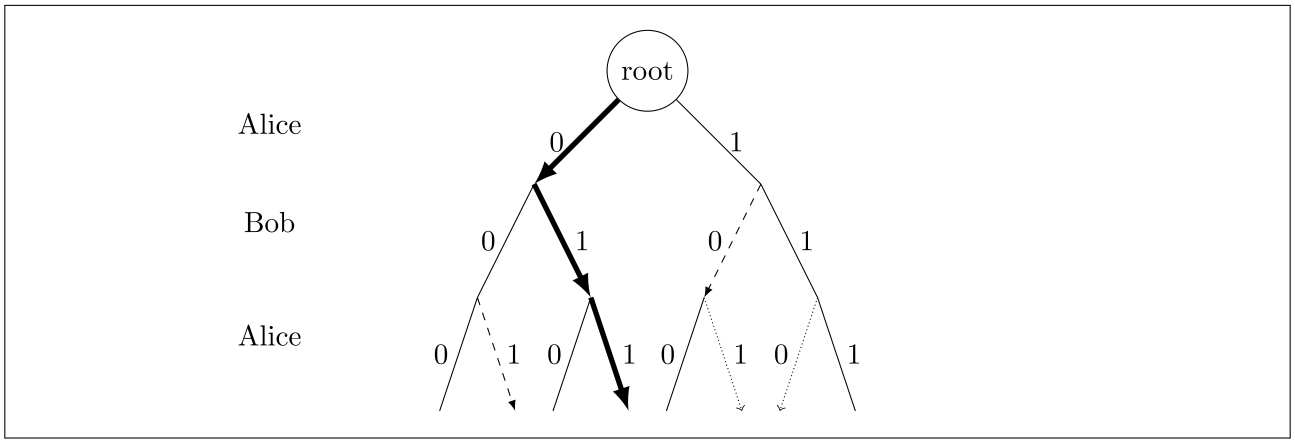

Figure 2: A π-tree showing the path P (bold edges) taken by Alice and Bob for computing $f(x, y)$. Dashed edges represents the hypothetical reply of Alice and Bob given that a different path P' was taken (when such replies are defined).

In order to obtain a protocol that withstands (a low rate of) channel noise, Alice and Bob *simulate* the construction of path P along the π-tree. The users transmit edges of P one by one, where each user transmits the next edge that extends the partial path transmitted so far.[5] This process is repeated for $N = \lceil T/(1 - \alpha) \rceil$ times, for some constant $\alpha < 1$ to be set later. In [BR11] it is shown that unless the noise rate exceeds $1/4$, after N rounds both parties will decode the entire path P with overwhelming probability. We refer the reader to [BR11] for a full description of the protocol and correctness proof. We now extend the analysis for the case of channel with noise and erasures.

To simplify the explanation, assume that each transmission is over the alphabet $\Sigma = \{0, \ldots, N\} \times \{0, 1\}^{\leq 2}$. Intuitively, the transmission $(e, s) \in \Sigma$ means "extend the path P by taking at most two steps defined by s starting at the child of the edge I have transmitted at transmission number e". Although this alphabet is not of constant size, it is easy to obtain a constant size alphabet by encoding each (e, s) into a delimited binary string (see Section 6 in [BR11]). Each symbol (e, s) is communicated to the other side via a $|\Sigma|$-ary tree code with distance α and alphabet Γ, that is at time n Alice sends $a_n \in \Gamma$, the last symbol of $\mathsf{TCenc}((e, s)_1, \ldots, (e, s)_n) = a_1 a_2 \cdots a_n$, and Bob receives $\tilde{a}_n \in \Gamma \cup \{\perp\}$, possibly with added noise or an erasure mark (similarly, Bob sends $b_n \in \Gamma$, and Alice

[5] The users transmit $\perp$ when they do not know how to extend P based on current information.

receives $\tilde{b}_n$). Let $\mathsf{TCdec}(\tilde{a}_n,\ldots,\tilde{a}_n)$ denote the string Bob decodes at time n (similarly, Alice decodes $\mathsf{TCdec}(\tilde{b}_1,\ldots,\tilde{b}_n)$). For every $i>0$, we denote with $m(i)$ the largest number such that the first $m(i)$ symbols of $\mathsf{TCdec}(\tilde{a}_1,\ldots,\tilde{a}_i)$ equal to $a_1,\ldots,a_{m(i)}$ and the first $m(i)$ symbols of $\mathsf{TCdec}(\tilde{b}_1,\ldots,\tilde{b}_i)$ equal to $b_1,\ldots,b_{m(i)}$.

Define $\mathcal{N}(i,j)$ to be the "effective" number of adversarial corruptions in interval $[i,j]$: the number of erasures plus twice the number of errors in the $[i,j]$ interval of the simulation (for both users).

Definition 5.3. *Let* $\mathcal{N}_a(i,j)=|\{k\mid i\le k\le j,\tilde{a}_k=\perp\}|+2|\{k\mid i\le k\le j,\tilde{a}_k\notin\{a_k,\perp\}\}|$, *and similarly define* $\mathcal{N}_b(i,j)$. *The* effective number of corruptions *in interval* $[i,j]$ *is* $\mathcal{N}(i,j)=\mathcal{N}_a(i,j)+\mathcal{N}_b(i,j)$.

We begin by showing that if $m(i)<i$ then many corruptions must have happened in the interval $[m(i)+1,i]$.

Lemma 5.4. $\mathcal{N}(m(i)+1,i)\ge\alpha(i-m(i))$.

Proof. Assume that at time i Bob decodes the string $a'_1,\ldots,a'_i$. By the definition of $m(i)$, $a'_1,\ldots,a'_{m(i)}=a_1,\ldots,a_{m(i)}$, and assume without loss of generality that $a'_{m(i)+1}\neq a_{m(i)+1}$. Note that the Hamming distance between $\mathsf{TCenc}(a_1,\ldots,a_i)$ and $\mathsf{TCenc}(a'_1,\ldots,a'_i)$ must be at least $\alpha(i-m(i))$. It is immediate then that for Bob to make such a decoding error, $\mathcal{N}_a\ge\alpha(i-m(i))$. □

Lemma 5.5 ([BR11])**.** *Let $t(i)$ be the earliest time such that both users announced the first i edges of P within their transmissions. For $i\ge0$, $k\ge1$, if $t(k)>i+1$, then $t(k-1)>m(i)$.*

Proof. The proof is taken from [BR11]: Without loss of generality, assume that the kth edge of P describes Alice's move. Suppose $t(k-1)\le m(i)$ and $t(k)>i+1$. Then it must be the case that the first $k-1$ edges of P have already been announced within the first $m(i)$ transmissions of both parties, yet the kth edge has not. By the protocol definition, Alice will announce this edge at round $i+1$, in contradiction to our assumption that $t(k)>i+1$. □

Next we show that if at some time i the length of the proposed P is not long enough (less then k), then many transmissions must have been corrupted.

Lemma 5.6. *For $i\ge-1$, $k\ge0$, if $t(k)>i+1$, then $\mathcal{N}(1,i)\ge\alpha(i-k+1)$.*

Proof. We prove by induction. The claim vacuously holds for $k=0$ and trivially holds for $i\le0$ since $\mathcal{N}(1,i)$ is non-negative. Otherwise, we have

$$\mathcal{N}(1,i)=\mathcal{N}(1,m(i))+\mathcal{N}(m(i)+1,i).$$

The second term, by Lemma 5.4 gives $\mathcal{N}(m(i)+1,i)\ge\alpha(i-m(i))$. For the first term, Lemma 5.5 suggests that $t(k-1)>m(i)$ and we can use the inductive hypothesis with $i'=m(i)-1$ and $k'=k-1$ to get

$$\mathcal{N}(1,m(i))\ge\mathcal{N}(1,i')\ge\alpha(i'-k'+1)=\alpha(m(i)-k+1).$$

□

The above Lemmas allow us to complete the proof of Theorem 5.2 by showing that if the simulation of P failed, there must have been "too many" corruptions.

Proof. (**Theorem 5.2**) Assume an unsuccessful run of the simulation protocol. That is, the simulation of the path P has failed, $m(N)<t(T)$. The number of adversarial corruptions throughout the protocol

is given by $\mathcal{N}(1,N) = \mathcal{N}(1,m(N)) + \mathcal{N}(m(N)+1,N)$ which by Lemma 5.6 and Lemma 5.4 is lower bounded by

$$\mathcal{N}(1,N) \geq \alpha(m(N)-T+1) \ + \ \alpha(N-m(N)) \geq \alpha(N-T) \geq \alpha(N-(1-\alpha)N) = \alpha^2 N.$$

Yet, assume the adversary is restricted to corrupt at most $c = 1/2 - \varepsilon$ fraction of the $2N = 2\lceil \frac{T}{1-\alpha} \rceil$ transmissions, then Lemma 3.4 guarantees that with overwhelming probability there will be at least $2cN(1-2S/L)$ erasures, assuming a Blueberry code $B : [S+1]^* \to [L+1]^*$ (here, $S+1 = |\Gamma|$). This implies that with overwhelming probability $\mathcal{N}(1,N) \leq 2cN(1+2S/L)$. For any $0 < \varepsilon \leq 1/2$ we can choose constants $\alpha < 1$ and $L > S$ such that $\alpha^2 > (1-2\varepsilon)(1+2S/L)$ and conclude that the protocol succeeds with overwhelming probability over the shared randomness. $\square$

References

[BK12] Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. *Electronic Colloquium on Computational Complexity (ECCC)*, 2012. TR12-043.

[Bon63] C. Bonsall. *The case of the hungry stranger.* HarperCollins, 1963.

[BR11] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 159–166, New York, NY, USA, 2011. ACM.

[Bra12] Mark Braverman. Towards deterministic tree code constructions. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 161–167. ACM, 2012.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644 – 654, nov 1976.

[EGM90] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO '89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 263–275. Springer Berlin / Heidelberg, 1990.

[GM01] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. In *ISOC Network and Distributed System Security Symposium, NDSS'01*, 2001.

[GMS11] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 768–777, oct. 2011.

[Gol04] Oded Goldreich. *Foundations of cryptography. Vol II: Basic applications.* Cambridge University Press, New York, 2004.

[GR97] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In Burton Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197. Springer Berlin / Heidelberg, 1997.

[GS10] Venkatesan Guruswami and Adam Smith. Codes for computationally simple channels: Explicit constructions with optimal rate. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:723–732, 2010.

[HO08] Brett Hemenway and Rafail Ostrovsky. Public-key locally-decodable codes. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 126–143. Springer Berlin / Heidelberg, 2008.

[HOSW11] Brett Hemenway, Rafail Ostrovsky, Martin Strauss, and Mary Wootters. Public key locally decodable codes with short keys. In Leslie Goldberg, Klaus Jansen, R. Ravi, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 6845 of *Lecture Notes in Computer Science*, pages 605–615. Springer Berlin / Heidelberg, 2011.

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

[Lip94] Richard Lipton. A new approach to information theory. In Patrice Enjalbert, Ernst Mayr, and Klaus Wagner, editors, *STACS 94*, volume 775 of *Lecture Notes in Computer Science*, pages 699–708. Springer Berlin / Heidelberg, 1994.

[MPSW05] Silvio Micali, Chris Peikert, Madhu Sudan, and David Wilson. Optimal error correction against computationally bounded noise. In Joe Kilian, editor, *Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2005.

[MS01] S. Miner and J. Staddon. Graph-based authentication of digital streams. In *Security and Privacy, 2001, IEEE Symposium on*, pages 232 –246, 2001.

[PCTS00] A. Perrig, R. Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *Security and Privacy, 2000, IEEE Symposium on*, pages 56 –73, 2000.

[Pec06] Marcin Peczarski. An improvement of the tree code construction. *Information Processing Letters*, 99(3):92–95, 2006.

[Sch92] Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:724–733, 1992.

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 747–756, New York, NY, USA, 1993. ACM.

[Sch96] Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

APPENDIX

A Construction of an Efficient Authentication Scheme

A.1 Proof of Proposition 4.5

In this appendix we show that the concatenation of Protocol 1 with a Blueberry code $[S+1]^* \to [L+1]^*$ satisfies the conditions of Proposition 4.5. We begin by showing that for any time n, if we look at k's which are not too close to the start or to the end, that is, k's in $K_n = \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$, then every string $\mathbf{x}^k$ is selected by the encoding scheme $\Theta(\log n)$ times in expectation. In addition, a constant fraction of a specific tree's transmissions is received intact, while the the expected number of error is a smaller fraction, controlled by the choice of the constant L. Therefore, a logarithmic prefix of the string $\mathbf{x}^k$ can be decoded with high probability.

Lemma A.1. *Let c, ε be given. For a given time n and for every $k \in K_n$,*

1. *the expected number of transmissions (i,j) with $\mathsf{ID}(i,j) = k$ is $\Theta(c_0 \log n)$.*
2. *if the corruption rate is at most c, then the expected number of transmissions with $\mathsf{ID}(i,j) = k$ not corrupted by the adversary is $\Theta(c_0 \log n)$ and the expected number of errors is $\Theta(c_0 \log n/L)$.*

Proof. Fix a $k \in K_n$, and recall that $Y_k = \{(i,j) \mid \mathsf{ID}(i,j) = k\}$. It is easy to verify that

$$\Pr\big((i,j) \in Y_k\big) = \begin{cases} 0 & i/\log i < k \\ \frac{1}{\left\lfloor \frac{i}{\log i} \right\rfloor} & i/\log i \geq k \end{cases},$$

where the probability is over the shared randomness R. Assume that the channel's (Eve's) noise pattern is $P = p_1, \ldots, p_{cn}$, with $p_i \in [n] \times [c_0]$. First let us bound the number of erroneous transmissions of symbols from Y_k. For a specific instance of the scheme, let

$$\mathsf{ERR}_k = \big|\big\{(i,j) \in Y_k \mid \tilde{y}^{i,j} \text{ is an error}\big\}\big|.$$

We are interested in the (adversarial) corruption pattern that maximizes the expected number of these errors,

$$\max_P \mathbb{E}\left[\mathsf{ERR}_k\right].$$

Since the decoding process ignores the first $\varepsilon n/4$ transmissions, and since the expected number of errors is a $\frac{S}{L}$-fraction of the corrupted transmissions in Y_k, this equals to

$$\max_p \mathbb{E}\left[\tfrac{S}{L}\left|\left\{(i,j) \;\middle|\; \mathsf{ID}(i,j) = k,\ i > \varepsilon n/4 \ \text{ and } \ \tilde{y}^{i,j} \neq y^{i,j}\right\}\right|\right],$$

where the expectation is over the shared randomness R. Note that $\Pr((i,j) \in Y_k)$ is monotonically decreasing for $i \geq t(k)$, and zero otherwise. The pattern P that maximizes Eve's probability to hit transmissions in Y_k is $P_\circ \triangleq \{t(k), t(k)+1, \ldots, t(k)+cn\}$. However, the decoding algorithm ignores the first $\varepsilon n/4$ indices and Eve has no use in attacking them. Therefore, if $P_\circ \cap [\varepsilon n/4] \neq \emptyset$, Eve's best strategy is to shift her attack to the window $P = \{\lfloor \varepsilon n/4 \rfloor, \lfloor \varepsilon n/4 \rfloor + 1, \ldots, \lceil (\varepsilon/4 + c)n \rceil\}$.

$$\begin{aligned}\max_P \mathbb{E}\left[\mathsf{ERR}_k\right] &= \max_P \frac{S}{L} \sum_{(i,j)\in P} \Pr\big((i,j) \in Y_k\big) \leq \frac{c_0 S}{L} \sum_{i=t(k)}^{cn+t(k)} \frac{1}{\frac{i}{\log i} - 1} \leq \frac{c_0 S}{L} \sum_{i=t(k)}^{cn+t(k)} \frac{\log i}{i - \log i} \\ &< \frac{c_0 S \log n}{L} \sum_{i=\varepsilon n/4}^{n} \frac{1}{i - \log i} < \frac{c_0 S \log n}{L} \sum_{i=\varepsilon n/4}^{n} \frac{1}{c'' i} = \frac{c_0 S \log n}{c'' L}(H_n - H_{\varepsilon n/4})\end{aligned}$$

where the first inequality on the second line applies to both cases of empty and non-empty $P_\circ \cap [\varepsilon n/4]$, and c'' is some constant $c'' < 1$ such that $c'' i \le i - \log i$ for $i \ge \varepsilon n/4$, for a sufficiently large n. H_n is the nth Harmonic number, and it holds that $0 < H_n - \ln(n) < 1$. We get $\max_P \mathbb{E}[\mathsf{ERR}_k] = O(c_0 \log n/L)$.

On the other hand, we can lower bound the amount of uncorrupt transmissions in Y_k. In a similar way to the above we define $\mathsf{INTACT}_k = \left|\left\{(i,j) \in Y_k \middle|\ \tilde{y}^{i,j} = y^{i,j}\right\}\right|$, and wish to lower bound the quantity $\min_P \mathbb{E}[\mathsf{INTACT}_k]$. It is easy to verify that Eve's strategy from above is optimal for this case as well, thus

$$\begin{aligned}\min_P \mathbb{E}[\mathsf{INTACT}_k] \ge \min_P \sum_{\substack{(i,j)\notin P\\ i>\varepsilon n/4}} \Pr\big((i,j)\in Y_k\big) &\ge \sum_{i=(c+\varepsilon/4)n+t(k)}^{n} \frac{c_0 \log i}{i}\\ &\ge c_0\big((1-c-\varepsilon/4)n - t(k)\big)\frac{\log((c+\varepsilon/4)n+t(k))}{n},\end{aligned}$$

which is $\Omega(c_0 \log n)$ for $t(k) \le (1-c-\varepsilon)n$, hence the claim holds for $k \le (1-c-\varepsilon)n/\log(1-c-\varepsilon)n$.

Finally, define $\mathsf{TOTAL}_k = |Y_k|$ to be the total amount of transmissions with $\mathsf{ID}(i,j) = k$ (erasures, errors, and intact). The expected amount of this quantity is at least

$$\mathbb{E}[\mathsf{TOTAL}_k] \le c_0 \sum_{i=\varepsilon n/4}^{n} \frac{\log i}{i - \log i} \le c_0 \sum_{\varepsilon n/4}^{n} \frac{\log i}{c'' i} \le \frac{c_0 \log n}{c''}(H_n - H_{\varepsilon n/4}) = O(c_0 \log n)$$

with some small constant $c'' < 1$ for a sufficiently large n. The sum begins from $\varepsilon n/4$ since $\mathbf{x}^k$ is declared only at time $t(k) \ge \varepsilon n/4$, if $k \ge \varepsilon n/4 \log(\varepsilon n/4)$. Since the number of intact transmissions is $\Omega(c_0 \log n)$, the total amount of transmission is lower-bounded by the same quantity, thus $\mathbb{E}[\mathsf{TOTAL}_k] = \Theta(c_0 \log n)$. □

Lemma A.2. *Let c, ε be given. If, for time n, at most cn of the transmissions were corrupted, then there exist constants c_0, L and a constant distance α such that for every constant $c_1 > 0$, the first $c_1 \log n$ elements of any $\mathbf{x}^k$ with $k \in \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$ are correctly decoded with polynomial computational effort, except with polynomially small probability over the shared randomness*

Proof. At time n, assume $\max_{k\in K_n} \mathbb{E}[\mathsf{TOTAL}_k] < C_T \log n$ and $\min_{k\in K_n} \mathbb{E}[\mathsf{INTACT}_k] > C_I \log n$, and define $\beta = C_I/C_T$. Note that β is independent of n and c_1. Denote by $\mathcal{BAD}_1$ the event that there were too many erasures and errors for the kth codeword, i.e. $\mathsf{INTACT}_k/\mathsf{TOTAL}_k < \beta/2$, and by $\mathcal{BAD}_2$ the event that there were not enough transmissions for the kth codeword, $\mathsf{TOTAL}_k < \frac{2c_1}{\beta}\log n$.

By an appropriate choice of $c_0 = O(c, \varepsilon, c_1, 1/\beta)$, we can bound the probability of any bad event to be polynomially small. For large enough c_0 we can assure that $\mathbb{E}[\mathsf{TOTAL}_k] > \frac{4c_1}{\beta}\log n$, and thus by Chernoff, $\Pr[\mathcal{BAD}_2] \le 2^{-\Omega(\log n)}$. Furthermore, a union bound gives

$$\Pr[\mathcal{BAD}_1] < \Pr[\mathsf{TOTAL}_k > \sqrt{2} C_T \log n] + \Pr[\mathsf{INTACT}_k < \frac{1}{\sqrt{2}} C_I \log n],$$

and by Chernoff inequality,

$$\Pr[\mathcal{BAD}_1] \le 2^{-\Omega(\mathbb{E}[\mathsf{TOTAL}_k])} + 2^{-\Omega(\mathbb{E}[\mathsf{INTACT}_k])} = 2^{-\Omega(\log n)}.$$

Conditioned on the fact that $\mathcal{BAD}_1$ and $\mathcal{BAD}_2$ did not occur, we know that $n^* > \frac{2c_1}{\beta}\log n$ symbols of $\mathsf{TCenc}(\mathbf{x}^k)$ were transmitted, and the adversary has corrupted at most $c^* < (1-\beta/2)$ fraction of these transmissions. Proposition 4.4 suggests that for an appropriate choice of constant L, we are able to decode a prefix of length at least $\approx (1-c^*)n^* = c_1 \log n$ except with probability $\exp(-\Omega(n^*)) = \exp(-\Omega(\log n))$. A union bound over all the possible $k \in K_n$ completes the proof.

As for the efficiency, since each codeword is of length $O(\log n)$, decoding via exhaustive search can be performed with polynomial computational effort. Hence, it is easy to verify that both the encoding and decoding can be done efficiently. □

A.2 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 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.3. *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.4. *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, only part of the stream, namely $x_{\varepsilon n/4}, \ldots, x_{(1-c-\varepsilon)n}$, is decoded by the above choice of $\mathbf{x}^k$s. In order to communicate the prefix $x_1, \ldots, x_{\varepsilon n/4}$ we run another instance of the scheme guaranteed by Proposition 4.5 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

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

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.5. *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 $V \triangleq x_1x_2\ldots x_{\lceil t(k_1)/2\rceil}x_1x_2\ldots x_{\lceil t(k_2)/2\rceil}x_1x_2\ldots$, and V is decoded by Protocol 1 with high probability.[6] 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 $\{\mathbf{v}^{(1-c-\varepsilon)n/4}, \ldots, \mathbf{v}^{(1-c-\varepsilon)n}\}$,

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

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 $\frac{3}{8}(1-c-\varepsilon) \le \ell \le \frac{1}{2}(1-c-\varepsilon)$ 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 $(1-c-\varepsilon)/4 > \varepsilon n/4$, the claim holds. □

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

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.6. *Let c, ε be constants $0 \le c < 1$, $0 < \varepsilon \le (1-c)/2$, 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.*

B Efficient Authentication Scheme with Exponentially Small Error

In this section we show how to change the scheme given by Proposition 4.5 such that the error probability is exponentially small. That is, for any time n the decoding scheme aborts with at most polynomially small probability in n, but on the event that the scheme did not abort, the decoded string matches the stream sent by Alice with overwhelming probability $1 - 2^{-\Omega(n)}$.

To this end, we add a parallel transmissions of random hash values of the entire stream (up to time n), where the hash length is logarithmic in n.[7] More formally, define an additional set of infinite strings $\{\mathbf{h}^1, \mathbf{h}^2, \ldots\}$. We identify a string $a = a_1 a_2 \cdots a_n$ with the n-dimensional vector $(a_1, a_2, \ldots, a_n)$, and define $\mathbf{h}^k$ in the following way. Randomly pick a matrix $R_k \in \{0,1\}^{\log t(k) \times t(k)}$ and a vector $V_k \in \{0,1\}^{\log t(k)}$ and set $\mathbf{h}^k = R_k \cdot (x_1, x_2, \ldots, x_{t(k)})^T + V_k$, concatenated with as many zeroes as needed. The strings $\{\mathbf{h}^k\}$ are interleaved with the strings $\{\mathbf{x}^k\}$ and $\{\mathbf{v}^k\}$, and c_1 increases as explained in Appendix A.2.

Proposition 4.5 guarantees that except with polynomially small probability in n all the hash values $\{\mathbf{h}^k\}_{k \in K_n}$ are correctly decoded. The decoding at time n aborts if any of the hash values $\{\mathbf{h}^k\}_{k \in K_n}$ mismatch the corresponding prefix $x_1 \cdots x_{t(k)}$.

Proposition B.1. *Given that Protocol 1 with hash testing did not abort, let x' be the decoded stream, then for every $\ell \le (1-c-\varepsilon)n$,*

$$\Pr_R[x'_\ell \neq x_\ell] < 2^{-\Omega(\max\{(1-c-\varepsilon)n-\ell\ ,\ \log n\})}.$$

Proof. Eve is oblivious of R_i and V_i, thus for *any* two vectors $\tilde{\mathbf{x}} \in \{0,1\}^{t(i)}$, $\tilde{\mathbf{h}} \in \{0,1\}^{\log t(i)}$ chosen by Eve, $\Pr_{R_i}[\tilde{\mathbf{h}} = R_i \cdot \tilde{\mathbf{x}}^T + V_i] < 2^{-\log t(i)}$.

Clearly, the smaller ℓ is, the more hash values that are checked to be consistent with the decoded x'_ℓ. For $\ell > \varepsilon n/4$ there are at least $((1-c-\varepsilon)n - \ell)/2\log n$ independent hash values of stream prefixes longer than ℓ, where the smallest hash length is $\approx \log(\varepsilon n/4)$. Hence, the probability that $x'_\ell \neq x_\ell$ yet the decoding procedure did not abort is at most

$$2^{-\Omega\left(\log(\varepsilon n/4)\frac{(1-c-\varepsilon)n-\ell}{2\log n}\right)} = 2^{-\Omega((1-c-\varepsilon)n-\ell))}.$$

Clearly, for $\ell < \varepsilon n/4$ there are as many hash tests as for $\ell = \varepsilon n/4$, thus the probability to incorrectly decode x_ℓ with $\ell < \varepsilon n/4$ is exponentially small in n as well. Finally, for the case where $((1-c-\varepsilon)n-\ell) < \log n$ we note that at least one hash value must be consistent, $\mathbf{h}^{\lceil (1-c-\varepsilon)n/\log(1-c-\varepsilon)n \rceil}$. The probability to incorrectly decode x_ℓ and pass the hash check is at most $2^{-\Omega(\log n)}$, which completes the proof. $\square$

[7]This method is somewhat equivalent to the classic authentication method of splitting a stream into chunks of size $\log n$ and adding a MAC of logarithmic size after each chunk.

The proof of Theorem 4.6 is immediate form the above Proposition.

Proof. (**Theorem 4.6**) Let c, ε be fixed. Perform Protocol 1 with hash testing with parameters $c, \varepsilon' = \varepsilon/2$. By Proposition B.1, every decoded x'_ℓ with $\ell \leq (1-c-\varepsilon)n$ satisfies

$$\Pr[x'_\ell \neq x_\ell] < 2^{-\Omega((1-c-\varepsilon')n-\ell)} \leq 2^{-\Omega(\varepsilon n/2)}.$$

□

C Computationally Secure Perpetual Authentication

For this section we assume basic knowledge of cryptographic primitives and assumptions.

Multi-Party Shared-Key Setting: The m-party shared-key setting for $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication is as follows. There are m parties, and parties i and j have shared random string $R_{i,j}$, for all $1 \leq i, j \leq m$. The adversary may corrupt any number of parties, and learn all of their shared random strings. The adversary may invoke any sequence of streaming authentication sessions involving parties of his choice (for sender and receiver), messages of his choice, and transmission lengths of his choice. If the adversary has corrupted the sender, then he may violate the encoding procedure in an arbitrary manner. The actions of the adversary may be adaptive (e.g., corrupt some additional parties after invoking some streaming authentication sessions, etc.), and interleaved (e.g., corrupt some additional parties before some streaming authentication sessions have finished).

Definition C.1. *A $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication Scheme is secure in the m-party shared-key setting if the adversary cannot induce any decoding error (as described in Definition 4.1) for any streaming authentication session for which he has not corrupted one of the parties, except with probability at most $s \cdot \kappa(n)$, where s is the total number of sessions initiated.*

Claim C.2. *The efficient scheme from the proof of Theorem 4.2 is secure in the m-party shared-key setting if the R_{ij} are independent (proof by union bound).*

Multi-Party PKI Setting: The m-party PKI setting for $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication is as follows. There are public parameters $params \leftarrow ParamGen(1^\lambda)$. There are m parties, and each party i has generated a public key pair $(e_i, d_i) \leftarrow KeyGen(params)$. The adversary can see all of the public keys, and can learn any private key by corrupting that party. Otherwise, the adversary can perform any attack as described for the multi-party shared-key setting, with the following two differences: (1) If party i is in a streaming authentication session with party j then he uses d_i and e_j to compute the shared random string, and (2) the adversary is limited to computation that can be performed in probabilistic polynomial time in λ, n, m, s, where s is the total number of streaming authentication sessions initiated.

Definition C.3. *A $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication Scheme is computationally secure in the m-party PKI setting if the adversary cannot induce any decoding error (as described in Definition 4.1) for any streaming authentication session for which he has not corrupted one of the parties, except with probability at most $s \cdot \kappa(n)$, where s is the total number of sessions initiated.*

Theorem C.4. *The efficient scheme from the proof of Theorem 4.2 is computationally secure in the m-party PKI setting under the Decision Diffie Hellman (DDH) assumption if $ParamGen(1^\lambda) \to g, q$ where g is a generator of prime order $q > 2^\lambda$; $KeyGen(g, q) \to (e_i, d_i)$ where $d_i \leftarrow [1 \ldots q-1]$ and $e_i = g^{d_i}$; For their ℓth streaming authentication session, parties i and j use $R_{ij} = G_{ij\ell}(g^{x_i x_j})$, where $\{G_{ij\ell}\}_{i,j,\ell}$ is a family of pseudorandom generators.*

Proof. (sketch) Let A be a probabilistic polynomial-time adversary that can break the $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication Scheme. Without loss of generality, A corrupts all but two of the m parties during its attack. We use A to construct a ppt adversary B for the DDH problem. Let (g, u, v, w) be a DDH challenge tuple. That is, g is a generator of order q for suitably large prime q, $u = g^x$ for $x \leftarrow [1 \ldots q-1]$, $v = g^y$ for $y \leftarrow [1 \ldots q-1]$, and $w = g^z$ (where $z = xy \bmod q$ with probability 1/2, and $z \leftarrow [1 \ldots q-1]$ with probability 1/2). Given a DDH challenge tuple, adversary B proceeds as follows:

1. Choose $i^*, j^* \leftarrow [1 \ldots m]$ (distinguished parties that B hopes will be targeted for attack).
2. For every party $k \neq i^*, j^*$, generate its public key and private key in the usual way. That is, the private key of party k is $x_k \leftarrow [1 \ldots q-1]$, and its public key is g^{x_k}.
3. For party i^*, its public key will be u (and its private key will be unknown). For party j^*, its public key will be v (and its private key will be unknown).
4. If party i^* and party j^* start their ℓth streaming authentication session, then run the efficient scheme from the proof of Theorem 4.2 using the output of $G_{i^* j^* \ell}(w)$ as the shared random string.
5. If party i^* and party $j \neq i^*$ start their ℓth streaming authentication session, then run the efficient scheme from the proof of Theorem 4.2 using the output of $G_{i^* j \ell}(u^{x_j})$ as the shared random string.
6. If party j^* and party $i \neq i^*$ start their ℓth streaming authentication session, then run the efficient scheme from the proof of Theorem 4.2 using the output of $G_{i j^* \ell}(v^{x_i})$ as the shared random string.
7. If party $i \neq i^*$ and party $j \neq j^*$ start a streaming authentication session, then run the efficient scheme from the proof of Theorem 4.2 using the output of $G_{ij\ell}(g^{x_i x_j})$ as the shared random string.
8. If adversary A corrupts party $k \neq i^*, j^*$, then give x_k to the adversary.
9. If adversary A ever attempts to corrupt party i^* or party j^* then halt and flip a coin to determine if (g, u, v, w) is a valid DDH tuple.
10. If adversary A succeeds in violating the decoding condition for any streaming authentication session involving uncorrupted parties, then determine that (g, u, v, w) is a valid DDH tuple.
11. If adversary A fails to violate the decoding condition for any streaming authentication involving uncorrupted parties, then determine that (g, u, v, w) is not a valid DDH tuple.

Let δ be the maximum success probability of any attack against the scheme in the multi-party shared-key setting (with independent keys). Let $\epsilon > \delta$ be the success probability of A against the scheme in the multi-party PKI setting. Here are the cases (and probabilities) where adversary B makes a correct determination of whether or not (g, u, v, w) is a valid DDH tuple:

Case 1: The adversary A never corrupts i^* or j^*, and the DDH challenge tuple is invalid. If the attack by A fails, then B makes the correct determination about (g, u, v, w). The probability of this case occurring is at least $(2/n(n-1))(1/2)(1-\delta) = (1-\delta)/((n(n-1))$.

Case 2: The adversary A never corrupts i^* or j^*, and the DDH challenge tuple is valid. If the attack by A succeeds, then B makes the correct determination The probability of this case occurring is at least $(2/n(n-1))(1/2)(\epsilon) = \epsilon/(n(n-1))$.

Case 3: The adversary A attempts to corrupt i^* or j^*, and B halts and flips a coin. If this coin flip matches the validity of the DDH challenge tuple then B makes the correct determination. The probability of this case occurring is $(1-(2/n(n-1)))(1/2) = 1/2-(1/(n(n-1))$.

These cases are independent, so the total probability that B makes the correct determination is the sum of their probabilities, which is at least $1/2+(\epsilon-\delta)/(n(n-1))$. If ϵ is non-negligibly greater than δ, then B distinguishes valid from invalid DDH challenge tuples with non-negligible advantage, which is a contradiction. This completes the proof sketch. □

D The Suffix Condition

Lemma D.1. *Let c and ϵ be given and let $\gamma = c/(c+\epsilon)$, then for the protocol given in Proposition 4.4, and for every noise level Bob's guessed string $x'_1, .., x'_{N_\gamma}$ is identical to the transmitted stream $x_1, .., x'_{N_\gamma}$*

Proof. Assume towards contradiction that Bob recovers the stream correctly only until time $t < N_\gamma$. Thus it follows that the errors and erasures in the $[t, N_\gamma]$ suffix (e and d respectively), satisfy $\alpha(N_\gamma - t) < 2e + d$. However, we know that N_γ satisfies the γ-suffix condition so $e + d < \gamma(N - t)$. Except with probability $2^{-\Omega(N-t)}$ it holds that $2e+d < (1+2\frac{S}{L})\cdot\gamma(N-t)$, thus except with the same probability, $\alpha < (1+2\frac{S}{L})\frac{c}{c+\epsilon}$ which is a contradiction to the way we choose α. □

However Bob does not know N_γ. We now show how to estimate this value.

D.1 Proof of Proposition 4.10 and Discussion

Consider the following procedure for estimating N_γ.
Let $c \in [0, 1)$ be given. For an input $\gamma \in (c, 1)$, at time n, Bob tries to find the longest suffix that satisfies the γ-suffix condition. To this end Bob performs the following.

1. Set $i = n$.
2. Check all the suffixes $x_t, \ldots, x_i$, with $t < i$.
   (a) If, in all such suffixes, the number of erasures is less than $\gamma(i-t)(1-2\frac{S}{L})$, output $N'_\gamma = i$.
   (b) Otherwise set $i \leftarrow i - 1$ and repeat. If $i < (1-c/\gamma)n$ break and output $N'_\gamma = \lfloor(1-c/\gamma)n\rfloor$.

Proof. **(Proposition 4.10).** Assume that Bob outputs $N'_\gamma > N_\gamma$. Since N_γ is the latest index that satisfies the γ-suffix condition, there must exist some time t such that $[t, N'_\gamma]$ has more than $\gamma(N'-t)$ corruption, yet the number of erasures in that interval is less than $\gamma(N'_\gamma - t)(1-2\frac{S}{L})$. This happens with probability exponentially small in $N'_\gamma - t$. If $t < N_\gamma$ then $N'_\gamma - t > N'_\gamma - N_\gamma$ which proves the claim for this case.

For the case where $t > N_\gamma$, we note that the time t does not satisfies the γ-suffix condition, therefore there must exists some time $t_1 < t$, such that the number of corruptions in the interval $[t_1, t]$ is more than $\gamma(t - t_1)$. If $t_1 > N_\gamma$, then there must exists time t_2 and interval $[t_2, t_1]$ that doesn't satisfy the γ-suffix condition. We repeat this reasoning until we find the first interval $[t_j, t_{j-1}]$ such that $t_j < N$. By considering the union of all these interval, it follows that the number of corruptions in $[t_j, N'_\gamma]$ is more than $\gamma(N'_\gamma - t_j) > \gamma(N'_\gamma - N_\gamma)$. However, Bob estimation process found at most $\gamma(N'_\gamma - N_\gamma)(1-2\frac{S}{L})$ erasures when it examined the suffix interval $[t_j, N'_\gamma]$ (otherwise, it would have failed the check in Step 2a). By Lemma 3.4/Corollary 3.5, this happens with probability exponentially small in $(N'_\gamma - N_\gamma)$. □

Bob learns a lower-bound on N_γ and it is guaranteed that $x'_1,...,x'_{N'_\gamma}$ is the same as $x_1,...,x_{N'_\gamma}$, maybe except for the last bits of the stream, in case N'_γ exceeds N_γ.

Bob can repeat the same procedure and compute a value N''_γ which usually upper-bounds N_γ, by finding the latest time i whose every suffix $[t,i]$ has less than $\gamma(i-t)(1-\frac{1}{2}\frac{S}{L})$. As above, the probability of the bad event that $N''_\gamma < N_\gamma$ is exponentially small in $(N_\gamma - N''_\gamma)$.

D.2 Proof of Lemma 4.9

Proof. Look at a suffix $y_{t+1},\dots,y_n$ for which the number of corruptions is strictly larger than $c\xi(n-t)$. If no such suffix exists then the lemma is true for $y_1,\dots,y_n$. Otherwise, discard $y_{t+1},\dots,y_n$, and repeat the process with $y_1,\dots,y_t$. At each iteration we remove more than $c\xi(n-t)$ corrupted transmissions and shorten the string by $n-t$ symbols. Assume that the process stops with some prefix of length $L < (1-\frac{1}{\xi})n$, then we have removed at least $c\xi(n-L) > c\xi(n-n+n/\xi) > cn$ corruptions which is a contradiction. Therefore, the entire process must stop with some prefix of length at least $(1-\frac{1}{\xi})n$. $\square$

E Error Correction of Datastreams

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

$$\Pr[x_1, x_2, \dots, x_{(1-2c)n} \neq x'_1, x'_2, \dots, x'_{(1-2c)n}] < 2^{-\Omega(n)},$$

that is, the receiver with overwhelming probability correctly decodes a prefix of the stream of length at least $(1-2c)n$.

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 correctly decode $x_1,\dots,x_n$ 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 $\tilde{x}$ such that $anc(x,\tilde{x}) < (c+\varepsilon)n$ with overwhelming probability. $\square$