# Amortized Communication Complexity of an Equality Predicate

Vladimir Nikishkin

Moscow Institute of Physics and Technology

**Abstract.** We study the communication complexity of the direct sum of independent copies of the equality predicate. We prove that the probabilistic communication complexity of this problem is equal to $O(N)$; the computational complexity of the proposed protocol is polynomial in the size of inputs. Our protocol improves the result achieved in 1991 by Feder et al. Our construction is based on two techniques: Nisan's pseudorandom generator (1992, Nisan) and Smith's string synchronization algorithm (2007, Smith).

## 1 Introduction

In this paper we study the amortized communication complexity of the equality predicate. We deal with the classic model of communication complexity with two participants (Alice and Bob), who want to compute some function of the data distributed between the participants. Alice and Bob can talk to each other via a communication channel. We measure the number of bits that must be transmitted between Alice and Bob to achieve the goal.

More specifically, let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function of two arguments. We assume that Alice is given the value of $x$, Bob is given the value of $y$, and Alice and Bob communicate with each other to compute the value $f(x,y)$.

We use three standard models of communication complexity: deterministic communication protocols, randomized communication protocols with public random bits, and randomized communication protocols with private random bits, see Kushilevitz and Nisan's textbook [1]. We denote communication complexities for these three models by $C_{det}$, $C^{\varepsilon}_{pub}$, and $C^{\varepsilon}_{priv}$, respectively. In the randomized versions, the superscript denotes the error probability and may be omitted if unnecessary.

Further, let us denote by $f^N$ the direct sum of $N$ independent copies of the initial function $f$. More precisely, the two arguments of $f$ are an $N$-tuple of values $(x_1, \ldots, x_N)$ and an $N$-tuple of values $(y_1, \ldots, y_N)$, and

$$f^N(x_1, \ldots, x_N, y_1, \ldots, y_N) = (f(x_1, y_1), \ldots, f(x_N, y_N)).$$

We assume that Alice is given all values of the $x_i$, and Bob is given all values of the $y_i$. Now Alice and Bob need to compute $f^N$, i.e., to get all the values $f(x_i, y_i)$ for $i = 1, \ldots, N$. It is natural to ask how the communication complexity of the problem $f^N$ grows as $N$ tends to infinity. The asymptotic behavior of this complexity is called the *amortized communication complexity* of $f$. More formally, the amortized communication complexity of $f$ is defined as

$$\mathrm{AC}(f) = \limsup_{N\to\infty} \frac{C(f^N)}{N}.$$

This definition makes sense for each model of communication, i.e., the value of $C$ in the definition above can be substituted with $C_{det}$, $C^{\varepsilon}_{pub}$, or $C^{\varepsilon}_{priv}$ (or by any other version of communication complexity known in the literature).

The question of communication complexity for the direct sum of several independent copies of the same problem is very natural, and it has been studied extensively for different models of communication complexity. This kind of questions were first raised by Karchmer *et al.* in [14]. Later, Feder *et al.* proved in [11] the first nontrivial lower bound for the deterministic model: $AC(f) = \Omega(\sqrt{C_{det}(f)} - \log n)$. Since then, several interesting results were achieved, mostly for more specific models of communication complexity (see [15–17]). However, the case of the classical randomized communication complexity remains not well understood. We only know that the gap between randomized communication complexity of one single instance of a problem and the corresponding amortized complexity can be rather large. E.g., such a gap was proven in [11] for the *equality predicate.*

The equality predicate $\mathrm{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as

$$\mathrm{EQ}_n(x,y) = \begin{cases} 1, \text{ if } x = y, \\ 0, \text{ if } x \neq y. \end{cases}$$

The communication complexity of the equality predicate is $C^{\varepsilon}_{priv} = \Theta(\log n)$ for any constant $\varepsilon > 0$, see [1]. On the other hand, the *amortized* randomized complexity of the equality predicate is only $O(1)$ with error probability $O(2^{-\sqrt{N}})$, [11]. In this paper we revisit the amortized communication complexity of the equality predicate. Our protocol achieves the same amortized communication complexity $O(1)$, and has a slightly better error probability $\varepsilon = O(2^{-\frac{N}{\log^2 N}})$. Besides, our protocol has "modular" structure; it consists of several independent gadgets, which makes the construction more flexible. So we hope that a similar technique can be applied to construct communication protocols with small amortized complexity for other functions.

Our construction is based on ideas similar to the protocol from [11]: (a) use random checksums to compare stings on Alice's and Bob's ends, (b) use a communication protocol with a public coin to compare Alice and Bob's checksums, and (c) use a pseudorandom generator to convert a public coin protocol to a private coin model. Still, the implementations of these ideas differ. The last idea (substitute public truly random bits by pseudorandom bits with a privately generated random seed) comes from the following classic theorem:

**Theorem 1 ([1]).** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function of two arguments. For every $\delta > 0$ and every $\varepsilon > 0$, it holds*

$$C^{\varepsilon+\delta}_{priv}(f) < C^{\varepsilon}_{pub}(f) + O(\log n + \log \delta^{-1}).$$

The proof of this theorem is not constructive, i.e., the pseudorandom generator used in the argument requires exponential (in the size of input) computations for Alice and Bob. The generators used in [11] and in our work run in polynomial (in $n$) time. However, the generator from [11] was created *ad hoc* for the EQ predicate problem, while the generator used in this paper is a well known general purpose generator, first described in [4].

Now let us formulate our main result.

**Theorem 2 (the main result).** *The randomized communication complexity (for the private coin model) of a direct sum of $N$ equality predicates* $\mathrm{EQ}_n$ *is equal to $O(N)$ if $n < N$, with error probability $P_{err} \leq O(2^{-c\frac{N}{\log^2 N}})$. Moreover, we explicitly construct a communication protocol that achieves this communication complexity and requires only polynomial time computations on Alice's and Bob's sides.*

In our construction, we use several classic tools (N. Nisan's pseudorandom generator, BCH codes, deterministic synchronization protocol by A. Orlitsky, [3]) and one relatively new construction (probabilistic synchronization protocol by A. Smith, [3]).

## 2 Classic Communication Protocols for EQ

### 2.1 Complexity of $\mathrm{EQ}_n$ for Different Types of Communication Protocols

The predicate $\mathrm{EQ}_n$ is very well studied. Let us remind the three different communication protocols for this predicate.

**Deterministic Model.** It is known that $C_{det}(\mathrm{EQ}_n) = n + 1$, see [1]. The bound is achieved by a trivial protocol: Alice transmits her string $x$ to Bob, Bob compares the two strings $x$ and $y$ and sends back one-bit response, 1 if the strings are equal and 0 otherwise. From the standard technique of fooling sets it follows that this bound is tight, i.e., there are no protocols with communication complexity less than $n + 1$.

**Private Coin Model.** For the randomized communication complexity with private sources of randomness $C^{\varepsilon}_{priv}(\mathrm{EQ}_n) = O(\log \frac{n}{\varepsilon})$. This bound is achieved by several classic communication protocols. Here we describe one of them. Alice and Bob view their inputs $x$ and $y$ as $n$-digit binary representations of integers (between 0 and $2^n - 1$ ). Alice chooses a prime number $p$ at random among the first $(n/\varepsilon)$ primes. She sends to Bob both $p$ and $x \mod p$. Bob verifies whether $x \mod p = y \mod p$. If $x$ and $y$ are equal modulo $p$, then Bob returns 1, otherwise he returns 0. If $x = y$, then this protocol always returns the correct result. If $x \neq y$, then the difference $(x - y)$ has at most $n$ prime factors; hence, the protocol returns the wrong answer with probability at most $\varepsilon$.

**Public Coin Model.** For randomized communication complexity with public source of randomness $C^{\varepsilon}_{pub}(\mathrm{EQ}_n) = O(\log \frac{1}{\varepsilon})$. This bound for communication complexity is achieved by the following protocol. Alice and Bob jointly choose a random $n$-bit string $r$. Then Alice computes the inner product $b = \langle x, r\rangle$ and transmits the result (a single bit) to Bob. Bob checks whether $b = \langle y, r\rangle$ and outputs "equal" if so and "not equal" otherwise. Obviously, if $x = y$, then the output is always "equal." On the other hand, if $x \neq y$, then by the properties of the inner product, $Pr[\langle x, r\rangle \neq \langle y, r\rangle] = \frac{1}{2}$. Thus, Bob outputs "not equal" with probability $\frac{1}{2}$. To decrease the probability of a wrong answer, Alice and Bob repeat these procedure several times with several independently chosen random strings $r$. If Alice and Bob repeat (in parallel or sequentially) $l$ times the described procedure, then the probability that $\langle x, r_i\rangle \neq \langle y, r_i\rangle$ for all $r_1, \dots, r_l$ is equal to $2^{-l}$. So, for $l = \lceil \log 1/\varepsilon \rceil$ we reduce the probability of error to $\varepsilon$, while communication complexity is $O(\log 1/\varepsilon)$.

## 2.2 Trivial Generalizations for $\mathrm{EQ}_n^N$

The protocols from the previous section can be easily adapted to get some protocols for the direct sum of $N$ copies of $\mathrm{EQ}_n$, i.e., for the function $\mathrm{EQ}_n^N$.

**Adaptation of the Protocol from Paragraph 2.1.** We run the protocol independently for each pair of blocks$(x_i, y_i)$. The probability of a wrong answer *for at least one pair of blocks* must be bounded by $\varepsilon$. To this end we need to reduce the probability of an error for each of the $N$ pairs of blocks to be less than $\varepsilon' = \varepsilon/N$. This results in communication complexity $O(N(\log(n/\varepsilon'))) = O(N(\log n + \log N + \log 1/\varepsilon)))$. Thus, from the trivial adaptation of the protocol from paragraph 2.1 we get $C^{\varepsilon}_{priv}(\mathrm{EQ}^N) = O(N(\log n + \log N + \log 1/\varepsilon)))$.

**Adaptation of the Protocol from Paragraph 2.1.** We run the protocol from section 2.1 for each pair of blocks $(x_i, y_i)$ independently. To guarantee that the total probability of error is bounded by $\varepsilon$, we need to reduce the probability of error for each pair of blocks to $\varepsilon' = \varepsilon/N$. Then we get

$$C^{\varepsilon}_{pub}(\mathrm{EQ}^N) = O(N(\log N + \log 1/\varepsilon)).$$

**From Public to Private Randomness.** The last protocol above can be transformed into a protocol with private source of randomness. Indeed, from Theorem 1 we get immediately

$$C_{priv} = O(N \cdot \log N + N \log \frac{1}{\epsilon} + \log(n \cdot N) + \log \frac{1}{\delta}) = O(N \cdot \log N + \log \frac{1}{\varepsilon}).$$

Note that this communication protocol requires exponential computational complexity (at least for the standard proof of Theorem 1).

How to reduce the obtained (rather trivial) bound $O(N \cdot \log N)$, hopefully to $O(N)$? How to achieve this bound with a communication protocol that requires only poly-time computations? The construction of such a communication protocol is the main result of this paper. Loosely speaking, we plan to do it in two steps. In the first step, we construct a more effective communication protocol for the communication model with public randomness (this part of our construction is based on ideas of A. Smith). In the second step, we reduce the protocol with public randomness to a protocol with private randomness. In some sense, this idea is similar to the usual proof of Theorem 1: we substitute the sequence of random bits (shared by Alice and Bob) by a sequence of *pseudorandom* bits, which can be obtained as an output of a pseudorandom generator. A random seed of this generator is rather short. So, one of the participants can choose it at random and then send to another participant. E.g., Alice chooses a random seed and sends it to Bob; then Alice and Bob apply the pseudorandom generator to this same seed, and then both participants obtain the same long string of pseudorandom bits. The sharp difference between our construction and the standard general proof of Theorem 1 is that we use an explicit and effectively computable generator (the generator of N. Nisan).

Before explaining the details of our construction, we remind the technical tools used in our proof.

## 3 Pseudorandomness, Codes and String Synchronization

### 3.1 Pseudorandom Generator

In our construction we need a pseudorandom generator that fools tests with bounded memory. Technically, we assume that a generator is a mapping $G : \{0,1\}^m \to \{0,1\}^n$, and a test is a randomized Turing machine with working space of some size $S$. Technically, we define a test (for a pseudorandom generator) as follows.

**Definition.** A statistical test with memory $S$ is a deterministic Turing machine $M$ with three tapes: a finite working tape of size $S$, an auxiliary read-only tape with some binary string $a = (a_1, ...a_n, ...)$ (an advice string), and a one-way input tape with an $n$-bit input $x$ (the reading head on the input tape can move from the left to the right but cannot move back to the left). We always assume that the length of $a$ should not be greater than $exp(S)$. This machine returns 1(true) or 0(false). We denote machine's output by $M_a(x)$. Informally the output means that test accepts/rejects $x$ given an advice string $a$.

**Definition.** A function $G : \{0,1\}^m \to \{0,1\}^n$ is called a pseudorandom generator, $\varepsilon$-robust for tests with memory $S(n)$, if for every *statistical test* $A$ with $S$ bits of working memory

$$|Pr_{y \in_r \{0,1\}^n}[A \text{ accepts } y] - Pr_{x \in_r \{0,1\}^m}[A \text{ accepts } G(x)]| < \varepsilon.$$

Here the notation $x \in_r X$ means $x$ *chosen uniformly at random from* $X$. Note that this definition involves several parameters: $n$, $m$, $S$, $\varepsilon$. In general, these parameters can be chosen independnetly. But typically we use this definition when $m$, $S$, $\varepsilon$ are some functions of $n$.

Nisan suggested in [4] an explicit construction of a pseudorandom generator that fools tests with sufficiently small memory:

**Theorem 3 ([4]).** *There exists a constant $c > 0$ such that for any $R$ and $S$ there exists a pseudorandom generator $G : cS \log R \to R$ (computable in time poly($R$)) that is $2^{-S}$-robust for all statistical tests with $S$ bits of working memory.*

In Section 4.6 we construct some statistical test (with small working memory $S$) which is roughly equivalent to our communication protocol. Then, we use the standard argument: the protocol with high probability returns the correct answer when running on truly random public bits; futher, the generator of Nisan fools our test; hence, given pseudorandom bits instead of truly random ones, the communication protocol must also return the correct answer with high probability.

## 3.2 BCH Codes

Our construction involves implicitly the classic BCH-codes, see [12]. We do not employ any specific properties of the construction of the BCH codes. We use only the fact that $\forall m > 3$ and $t < 2^{m-1}$ there exists an explicit construction of a linear code with parameters $[n, k, d]$ such that the codeword length is $n = 2^m - 1$, the number of checksum bits is $n - k \leq mt$, and the minimal distance between codewords of the code is $d \geq 2t + 1$. We also use the fact that BCH codes can be decoded efficiently by the Berlekamp-Messy algorithm, [13].

The BCH codes are not used explicitly in our paper. However, we use a construction by Orlitsky from Section 3.3. This construction involves a linear error correcting code, which is not chosen explicitly in Section 3.3. In our applications, the BCH codes fits perfectly that construction. In what follows we refer to Orlitsky's protocol assuming that the codes used there are the BCH codes.

## 3.3 Strings Synchronization Protocols

In our communication protocol we will need to solve the following auxiliary problem. Let Alice and Bob each hold an $n$-bit string, $A$ and $B$, respectively. We assume that $A$ and $B$ differ from each other in at most $e$ positions. Alice and Bob want to exchange their inputs, i.e., Alice should get string $B$, and Bob should get string $A$. We will call this problem the *string synchronization problem* (Alice and Bob want to synchronize their inputs).

Orlitsky suggested in [3] a deterministic communication protocol for the problem of synchronization of a pair of $n$-bit strings with the Hamming distance from each other at most $e$. Communication complexity of this protocol is $O(e \log n)$. All computations of Alice and Bob in this protocol run in polynomial time in the length of the strings. More formally, the theorem (see [2]) is as follows:

**Theorem 4.** *Assume there exists a linear error-correcting code with parameters $(\alpha, R(\alpha))$, with a polynomial time decoding algorithms. Then there exists a one-round communication protocol solving the string synchronization problem with* *communication complexity $C = (1 - R(\alpha)) \cdot n$. Computational complexity of this protocol is polynomial.*

If the BCH code is used (noted in section 3.2), the communication complexity of this protocol is $O(e \log n)$. The protocol of Orlitsky makes sense if the distance $e$ between strings is very small. In case $e = \Omega(n)$, the communication complexity of Orlitsky's protocol is worse than the trivial upper bound $2n$.

The parameters of the BCH code correspond to the ones of the synchronization protocol code in the following way: $\alpha = d/n, R(\alpha) = n - k$

Adam Smith suggested in [9] a randomized communication protocol for the problem of strings synchronization with an asymptotically optimal bound for communication complexity for the case $e = const \cdot n$. More precisely, Smith proved that for every $\delta(n) = \Omega(\frac{\log \log n}{\sqrt{\log n}})$ there exists an explicit communication protocol (with a private source of randomness) that solves the problem of synchronization of $n$ bit strings that differ in at most $e$ positions, with communication complexity $n(H(\frac{e}{n}) + \delta)$ and error $\varepsilon = 2^{-\Omega(\frac{\delta^3 n}{\log n})}$, where $H(p) = p \log_2 \frac{1}{p} + (1-p) \log \frac{1}{1-p}$. Algorithms of Alice and Bob in this protocol run in polynomial time.

## 4 Proof of the Main Theorem

In this section we present a protocol for $EQ_n^N$ and prove Theorem 2.

### 4.1 Overview of the Protocol

Our protocol runs as follows. First of all, Alice generates a string of truly random bits of length $O(N)$ and sends this string to Bob. They both use Nisan's generator and produce pseudorandom bits from this seed. In what follows, Alice and Bob use this long string of pseudorandom bits.

Then, Alice and Bob iteratively calculate "checksums" (inner products mod 2 with the pseudorandom string) for their $n$-bit blocks and synchronize strings of the resulting checksums using the probabilistic or deterministic protocol from Section 3.3. As soon as some pair of non-equal blocks $X^i$, $Y^i$ is revealed (if some checksums for these blocks are different), Alice and Bob remove these blocks from the list of their bit strings and never test them again. Thus, in every consecutive iteration the fraction of non-equal pairs of blocks (that are not discovered yet) becomes smaller and smaller.

in every consecutive iteration, we make the length of the checksums larger and larger, so for each pair of non-equal blocks the probability to be discovered becomes closer and closer to 1. Hence, the fraction of (non-discovered) pairs of non-equal blocks gradually reduces, and only pairs of equal blocks remain untouched at their places. This means that in every consecutive iteration the Hamming distance between arrays of checksums (obtained by Alice and Bob respectively) becomes smaller and smaller.

In each iteration Alice and Bob need to exchange the checksums computed for their blocks of bits (inner products with the same pseudorandom bits). For several initial iterations (technically, for $\log\log N$ iterations) we use the randomized synchronization protocol by Smith. Then we switch to the deterministic protocol by Orlitsky. In what follows we explain this protocol in more detail.

## 4.2 Generation Stage

Alice generates $r = \log((n \cdot N)^4) \cdot \log(2^{\frac{N}{\log(n \cdot N)}}) = O(N)$ random bits and sends them to Bob. Then Alice and Bob apply Nisan's pseudorandom generator from Section 3.1 and get $R = n^2N^2$ pseudorandom bits. The length of the seed $r$ is chosen so that the generator is $\varepsilon$-robust against tests with working memory of size $\frac{N}{\log(n \cdot N)}$.

## 4.3 Probabilistic Synchronization Stage (steps $i = 1, \dots, \log\log N$)

The input of Alice is a sequence of $N$ blocks $X = (X^1, \dots, X^N)$, and the input of Bob is a sequence of $N$ blocks $Y = (Y^1, \dots, Y^N)$. Each block $X^j$ or $Y^j$ is an $n$-bits string.

We start the discussion with a minor technical difficulty. In our construction we employ the synchronization protocols by Orlitsky and Smith; these protocols need to know in advance the distance between the strings that should be synchronized. As we may not know the initial distance between $X$ and $Y$, we will add $N$ dummy equal blocks to both $X$ and $Y$. This simple trick guarantees that in the very first stage of the protocol the fraction of equal pairs of blocks is no less than 1/2. This trick increases the total number of blocks from $N$ to $2N$, but it will not affect the asymptotic complexity of our protocol.

Now we explain the main part of the protocol. For $i = 1, \dots, \log\log N$ we repeat the following procedure. We set $\lambda$ (a constant to be fixed later). Alice and Bob computes for each of their blocks (for all $X^j$ and $Y^j$) $\lambda$ random checksums. One checksum for each block $X^j$ or $Y^j$ is the inner products modulo 2 between this block and a new portion of pseudorandom bits generated in the previous stage, e.g., for $X^j = x_1 \dots x_n$ and $Y^j = y_1 \dots y_n$ the checksums are the bits

$$x_1r_1 + \dots + x_nr_n \pmod 2 \text{ and } y_1r_1 + \dots + y_nr_n \pmod 2,$$

where $r_1 \dots r_n$ is a block from the stream of pseudorandom bits, similar to the protocol in Section 2.1 (Alice and Bob share the same sequence of pseudorandom bits, so they can use the same bits $r_1 \dots r_n$ in the checksums for both $X^j$ and $Y^j$). Thus, the resulting string of checksums (for Alice and Bob) consists of $\lambda \cdot$ [number of blocks] bits. E.g., at the very first iteration it makes $\lambda \cdot 2N$ bits since Alice and Bob computes the checksums for all $2N$ blocks ($N$ original blocks and $N$ dummy blocks).

Then Alice and Bob exchange their checksums using the randomized string synchronization protocol by Smith. In the $i$-th step we run the synchronization protocol assuming that Alice's and Bob's checksums differ from each other in a fraction at most $2^{-i}$ of blocks (this threshold for the number of different blocks should be given to the synchronization protocol). When the checksums are exchanged, Alice and Bob remove from their lists all blocks $X^j$ and $Y^j$ whose checksums are not identical. Note that for a pair of equal blocks $X^j$, $Y^j$, the random checksums are always equal. If blocks are not equal to each other, then the probability to get $\lambda$ equal checksums is about $2^{-\lambda}$ (this probability is not *exactly* $2^{-\lambda}$ since Alice and Bob use pseudorandom bits $r_1 \dots r_n$ rather than truly random ones to compute the checksums).

Typically, on each step the number of non-discovered pairs of non-equal blocks $X^j$, $Y^j$ is reduced by a factor of about $2^{-\lambda}$. We say that the $i$-th step of the described procedure *fails*, if in this stage Alice and Bob discover less than 50% of the remaining pairs of non-equal blocks $X^j$, $Y^j$. If at least one step fails, we cannot guarantee the correctness of the result of the protocol. If no step fails, then on the $i$-th step the arrays of checksums of Alice and Bob differ from each other in a fraction at most $1/2^i$ of all computed inner products.

The communication complexity of this stage of the protocol is the sum of communication complexities of runs of Smith's protocol at steps $i = 1, \dots, \log\log N$:

$$\sum_{i=1}^{\log\log N} (H(1/2^i) + \delta)\lambda N = O(N),$$

where $\delta = \frac{\log\log nN}{\sqrt{\log nN}}$. The last equation follows from the property of Smith synchronization protocol and from the asymptotic $H(\alpha) = \alpha\log(1-\alpha) + O(1)$ as $\alpha$ tends to 0.

## 4.4 Deterministic Synchronization Stage ($i = \log\log N + 1, \dots, \log N$)

In this stage we continue essentially the same procedure as in the prevoius stage. There are two important differences: now we use checksums of variable size $\lambda_i$, and Alice and Bob apply the deterministic protocol of Orlitsky (instead or the randomized protocol of Smith) to exchange their checksums.

At each step $i = \log\log N, \dots, \log N$ Alice and Bob compute $\lambda_i = \lceil\frac{2^i}{\log^2 N}\rceil$ checksums for each remaining block $X^j$ and $Y^j$, respectively. Again, each checksum of a block is the inner product (mod 2) with a new portion of pseudorandom bits. Then Alice and Bob exchange the computed lists of checksums. Now they use the deterministic synchronization protocol by Orlitsky, see Section 3.3.

The communication complexity of the deterministic protocol is about $\log N$ times greater than the complexity of the protocol by Smith. But nevertheless we can use it since the Hamming distance between checksums is reasonably small. The communication complexity of this stage is

$$\sum_{i=\log\log N}^{\log N} \lceil\frac{N\lambda_i \log N}{2^i}\rceil \approx \sum_{i=\log\log N}^{\log N} \frac{\log N \cdot 2^i}{\log^2 N \cdot 2^i} \cdot N = O(N).$$

## 4.5 Summary

When the described stages are completed, we assume that Alice and Bob have discovered all pairs of non-equal blocks. All the remaining pairs $X^j$, $Y^j$ (all pairs of blocks whose checksums at all steps of the protocol remain equal to each other) are considered equal.

## 4.6 Probability of Error

We need to estimate the probability of error in our protocol. For simplicity, let us assume first that instead of $R$ pseudorandom bits Alice and Bob share $R$ independent and uniformly distributed random bits (so, we temporarily switch to the model with a public source of randomness). Then stages 4.3 and 4.4 make sense, and we can estimate the probability of error of the protocol.

The protocol may return a wrong answer because of the following reasons: (1) the probabilistic synchronization protocol of Smith fails at some stage; (2) some of the steps $i = 1, \dots, \log N$ fail since more than 50% of random checksums turn out to be equal for non-equal pairs of blocks $X^j$, $Y^j$. Let us estimate the probabilities of each of these two events.

**Error in the Probabilistic Synchronization Protocol.** Summing up the probabilities of error in Smith's synchronization in every step of our protocol we obtain (for some constant $c > 0$)

$$P(Err) = \sum_{i=1}^{\log\log N} O(2^{-(\frac{N}{\log N})}) \leq O(2^{-\frac{cN}{\log N}}).$$

**Failure of Checksum Verification.** A step $i = 1, \dots, \log N$ fails if for more than half of (not discovered yet) pairs of non-equal blocks $X^j$, $Y^j$ all the checksums turn out to be equal. We estimate the probability of this event using the Chernoff bound. We may assume that after the first $(i-1)$ steps there remain $N/2^i$ pairs of non-equal blocks.

For a pair of blocks $X^j, Y^j$ that are not equal, the probability that their inner products with a random string $r_1 \dots, r_n$ have the same parity, is equal to $1/2$. When we calculate $\lambda$ independent checksums, the probability that all pairs of checksums for $X^j$ and $Y^j$ are equal to each other, is $1/2^\lambda$. We say that the whole step of the protocol fails if the event *all checksums are equal* happens for more than 50% of pairs of non equal $X^j$, $Y^j$.

We assumed that after $(i-1)$ steps of the protocol the number of remaining non equal pairs of blocks $(X^j, Y^j)$ is $N_i = \frac{N}{2^i}$. For each of these pairs the probability *not to be discovered at step* $i$ is $2^{-\lambda}$. We estimate the probability of failure, i.e., the probability that more than $N_i/2$ pairs remain not revealed. By the Chernoff bound this probability is not greater than $2^{N_i D(q,p)}$, where

$$D(q,p) = q \ln(\frac{q}{p}) + (1-q)\ln(\frac{1-q}{1-p}) \text{ for } q = \frac{1}{2}, p = \frac{1}{2^\lambda}.$$

In the first steps $i = 1, \ldots, \log\log N$ steps of the protocol $\lambda = const$ (a large enough constant), and later for $i = \log\log N + 1, \ldots, \log N$ we have $\lambda_i = \frac{2^i}{\log^2 N}$. Hence, the probability of failure $P(Err_i) = O(2^{-\frac{N}{\log^2 N}})$. Sum up the error probabilities for all steps of stages 2 and 3:

$$\sum_{i=1}^{\log N} O(2^{-\frac{N}{\log^2 N}}) \approx \log N \cdot O(2^{-\frac{N}{\log^2 N}}) = O(2^{-c\frac{N}{\log^2 N}})$$

for some $c > 0$.

**Pseudorandom Generator.** In this section we construct a statistical test (see the definition in Section 3.1) that simulates one step of our protocol. In a sense, this test verifies that (pseudo)random bits are "suitable" for our communication protocol: they do not cause a failure of the protocol in the $i$-th iteration. The "advice strings" of this statistical test contain a sequence of pairs of blocks $X^j$, $Y^j$ from the inputs of Alice and Bob that have passed the checksum tests in the first $i-1$ rounds. The input $x$ is a string of (pseudo)random bits that should be accepted or rejected. The test rejects $x$ (for a given advice string $a$), if our communication protocol fails in the $i$-th round with random bits $x$ while Alice and Bob are given the blocks $X^j$, $Y^j$ corresponding to the advice $a$.

The algorithm of the test is straightforward: it computes the checksums for $X^j$ and $Y^j$ as it is done by our communication protocol in the $i$-th round, with random bits $x$ shared by Alice and Bob, and compares the corresponding checksums for Alice's and Bob's blocks. Note that the test does not simulate the synchronization procedure (the sub-protocols following the construction of Orlitsky and Smith).

The working space of our machine is $O(\frac{N}{\log^2 N})$. This is enough to simulate the computation of the checksums performed by our communication protocol. The test accepts $x$, if in the simulation at least 50% of non-equal pairs of blocks are successfully revealed, and rejects $x$ otherwise. In other words, a teststring $x$ is rejected if it causes a failure in the $i$-th round of the protocol.

Theorem 3.1 guarantees that Nisan's pseudorandom generator fools this test. Hence, for our protocol the probability of failure with pseudorandom bits is not much greater than the probability of failure for truly random bits. More precisely, the difference between the probabilities of failure for random and pseudorandom bits is at most

$$2^{-S(N)} = O(2^{-c\frac{N}{\log^2 N}}).$$

We sum up these values for all steps $i = 1, \ldots, \log N$ and get

$$P(Err) \leq \sum_{i=1}^{\log N} O(2^{-\frac{N}{\log^2 N}}) = O(2^{-c_3\frac{N}{\log^2 N}}).$$

**Summary.** The probability of error of our protocol consists of three parts: (1) the probability of error in Smith's protocol, (2) the probability of failure with truly random checksums in some step $i = 1, \ldots, \log N$, and (3) the additional probability of failure caused by the difference between truly random and pseudorandom checksums. Therefore,

$$P(Err) = O(2^{-c_1(\frac{N}{\log N})}) + O(2^{-c_2 \frac{N}{\log^2 N}}) + O(2^{-c_3 \frac{N}{\log N}}) = O(2^{-C \frac{N}{\log^2 N}}).$$

This concludes the proof of the correctness of our communication protocol.

## References

1. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge Univ. Press (1997)
2. Chuklin, A.: Effective protocols for low-distance file synchronization. arXiv:1102.4712 (2011)
3. Orlitsky, A.: Interactive communication of balanced distributions and of correlated files. SIAM Journal on Discrete Mathematics 6, 548–564 (1993)
4. Nisan, N.: Pseudorandom Generators for Spacebounded Computation. Combinatorica 12(4), 449–461 (1992)
5. Nisan, N., Widgerson, N.: Hardness vs. Randomness. Journal of Computer and System Sciences 49(2), 149–167 (1994)
6. Canetti, R., Goldreich, O.: Bounds on Tradeoffs between Randomness and Communication Complexity. Computational Complexity 3(2), 141–167 (1990)
7. Newman, L.: Private vs. Common Random Bits in Communication Complexity. Information Processing Letters 39(2), 67–71 (1991)
8. Impagliazzo, R., Nisan, N., Widgerson, A.: Pseudorandomness for Network Algorithms. In: Proc. of the 26th ACM Symposium on Theory of Computing, pp. 356–364 (1994)
9. Smith, A.: Scrambling Adversarial Errors Using Few Random Bits, Optimal Information Reconciliation, and Better Private Codes. In: Proc. of the 18th ACM-SIAM Symposium on Discrete Algorithms, pp. 395–404 (2007)
10. Nisan, N., Zukerman, D.: Randomness is Linear in Space. 1993 Journal of Computer and System Sciences 52(1), 43–52 (1996)
11. Feder, T., Kushilevitz, E., Naor, M., Nisan, N.: Amortized Communication Complexity. SIAM J. Comput. 24(4), 736–750 (1991)
12. Bose, R.C.M., Ray-Chaudhuri, D.K.: On A Class of Error Correcting Binary Group Codes. Information and Control 3(1), 68–79 (1960)
13. Berlekamp, E.R.: Nonbinary BCH decoding. IEEE Transactions on in Information Theory 14(2), 242 (1967)
14. Karchmer, M., Raz, R., Wigderson, A.: On Proving Super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity. In: Proc. of 6th IEEE Structure in Complexity Theory, pp. 299–304 (1991)
15. Parnafes, I., Raz, R., Wigderson, A.: Direct Product Results and the GCD Problem, in Old and New Communication Models. In: STOC 1997 Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, pp. 363–372 (1997)
16. Chakrabarti, A., Shi, Y., Wirth, A., Yao, A.: Informational Complexity and the Direct Sum Problem for Simultaneous Message Complexity. In: Proceedings of 42nd IEEE Symposium on Foundations of Computer Science (2001)
17. Sherstov, A.: Strong Direct Produce Theorems for Quantum Communication and Query Complexity. In: STOC 2011 Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 41–50 (2011)