# Randomized Polynomial Time Protocol for Combinatorial Slepian-Wolf Problem

Daniyar Chumbalov[1] and Andrei Romashchenko[2]

[1] Ecole Polytechnique Federale de Lausanne, daniyar.chumbalov@epfl.ch
[2] Le Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), andrei.romashchenko@lirmm.fr

**Abstract.** We consider the following combinatorial version of the Slepian–Wolf coding scheme. Two isolated Senders are given binary strings $X$ and $Y$ respectively; the length of each string is equal to $n$, and the Hamming distance between the strings is at most $\alpha n$. The Senders compress their strings and communicate the results to the Receiver. Then the Receiver must reconstruct both strings $X$ and $Y$. The aim is to minimise the lengths of the transmitted messages.
The theoretical optimum of communication complexity for this scheme (with randomised parties) was found in [8], though effective protocols with optimal lengths of messages remained unknown. We close this gap and present for this communication problem a polynomial time randomised protocol that achieves the optimal communication complexity.

**Key words:** Slepian-Wolf coding, communication complexity, coding theory, randomized encoding, pseudo-random permutations

## 1 Introduction

The classic Slepian–Wolf coding theorem characterises the optimal rates for the lossless compression of two correlated data sources. In this theorem the correlated data sources (two sequences of correlated random variables) are encoded separately; then the compressed data are delivered to the receiver where all the data are jointly decoded, see the scheme in Fig. 1. The seminal paper [1] gives a very precise characterisation of the profile of accessible compression rates — Slepian and Wolf found a natural and intuitive characterisation in terms of Shannon's entropies of the sources.

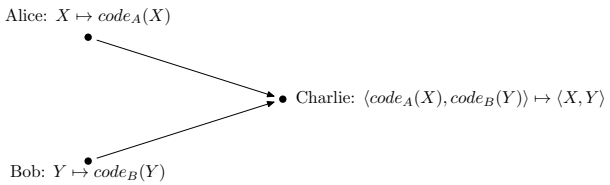


**Fig. 1.**

It seems instructive to view the Slepian–Wolf coding problem in the general context of information theory. One paper by Kolmogorov was entitled *Three approaches to the quantitative definition of information*, [12]. These three approaches where the *combinatorial* one (*cf.* Hartley's combinatorial definition of information, [13]), the *probabilistic* one (*cf.* Shannon's entropy), and the *algorithmic* one (*cf.* algorithmic complexity a.k.a. as Kolmogorov complexity). Many fundamental concepts and constructions in information theory have parallel implementations in all three approaches. An evident example of this parallelism is provided by the formal information inequalities: these inequalities can be equivalently represented as linear inequalities for Shannon's entropy, for Kolmogorov complexity, [14], or for (logs of) cardinalities of finites sets, [15], [16]. It is remarkable that some results known in one of these approaches look very similar to its homologues from the two other versions of information theory, while mathematical techniques and formal proofs behind them are fairly different.

Keeping in mind the idea of three parallel approaches, we notice that for the multi-source coding theory there exist two parallel versions: the probabilistic/Shannon's framework (the Slepian–Wolf coding theory and its generalisations) and the algorithmic/Kolmogorov's one (Muchnik's theorem on conditonal coding [17] and its generalisations, respectively). What is missing in this picture is the "combinatorial" version of these theorems. We try to fill this gap and start with formal definitions and some bounds for the combinatorial Slepian–Wolf coding scheme[3].

Thus, we investigate the combinatorial version of the Slepian–Wolf coding problem. To simplify the notation, we focus on the symmetric binary case of this problem. Formally, we consider a communication scheme with two senders (let us call them Alice and Bob) and one receiver (we call him Charlie). We assume Alice is given a string $X$ and Bob is given a string $Y$. Both strings are of length $n$, and the Hamming distance between $X$ and $Y$ is not greater than $\alpha n$. The senders prepare some messages for the receiver (Alice computes her message given $X$ and Bob computes his message given $Y$). When both messages are delivered to Charlie, he should decode them and reconstruct both strings $X$ and $Y$. Our aim is to characterise the optimal lengths of Alice's and Bob's messages.

This is the general scheme of the combinatorial version of the Slepian–Wolf coding problem. Let us place emphasis on the most important points of our setting: (a) the input data are distributed between two senders: Alice knows $X$ but not $Y$ and Bob knows $Y$ but not $X$; (b) one way comminiction: Alice and Bob send some messages to Charlie without feedback; (c) no communications between Alice and Bob; (d) parameters $n$ and $\alpha$ are known to all parties.

[3] I. Csiszar and J. Körner described the Slepian–Wolf theorem as "*the visible part of the iceberg*" of the multi-source coding theory; since the seminal paper by Slepian and Wolf, many parts of this "iceberg" were revealed and investigated, see a survey in [18]. Similarly, Muchnik's theorem has motivated numerous researches in the theory of Kolmogorov complexity. Apparently, a similar (probably even bigger) "iceberg" should also exist in the combinatorial version of information theory. However, before we explore this iceberg, we should understand first the very basic multi-source coding models, and the most natural starting point is the combinatorial version of the Slepian–Wolf coding scheme.

It is usual for the theory of communication complexity to consider two types of protocols: deterministic communication protocols (Alice's and Bob's messages are deterministic functions of $X$ and $Y$ respectively) and randomised communication protocol (encoding and decoding procedures are randomised, and for each pair $(X, Y)$ Charlie must get the right answer with only a small probability of error $\varepsilon$). We use the following standard *communication model with private sources of randomness*:

- each party (Alice, Bob, and Charlie) has her/his own "random coin" — a source of random bits,
- the coins are fair, i.e., produce independent and uniformly distributed random bits,
- the sources of randomness are private: each party can access only its own coins.

The last condition (the random coins are private) is crucial. In the model with public randomness, the problem of constructing an effective protocol is typically much simpler. We emphasise the difference between the classic probabilistic setting of the Slepian–Wolf coding and randomised protocols for combinatorial version of this problem. In the probabilistic approach, decoding should succeed for *most* pairs of inputs $(X, Y)$; in the combinatorial approach we require that for *each* pair of inputs $X, Y$ (with a given Hamming distance) the protocol succeeds with high probability.

In terms of the theory of communication complexity, we are looking for an optimal one-round communication protocol. We are interested not only in the total communication complexity (the sum of the lengths of Alice's and Bob's messages) but also in the trade-off between the two sent messages. More formally, we want to characterise the set of *achievable pairs of rates*:

**Definition 1.** *We say that a pair of integers $(k_a, k_b)$ is* an achievable pair of rates *for the combinatorial Slepian–Wolf problem (in the communication model with private sources of randomness) with parameters $(n, \alpha, \varepsilon)$ if there exists a randomised communication protocol such that*

- *the length of Alice's message is equal to $k_a$,*
- *the length of Bob's message is equal to $k_b$,*
- *for each pair of inputs $x, y \in \{0,1\}^n$ such that $\operatorname{dist}(x, y) \le \alpha n$, the probability of the error is less than $\varepsilon$.*

A simple counting arguments gives very natural lower bounds for lengths of messages in this problem:

**Theorem 1 ([8]).** *For all $\varepsilon \ge 0$ and $0 < \alpha < 1/2$, a pair $(k_a, k_b)$ can be an achievable pair of rates for the combinatorial Slepian–Wolf problem with parameters $(n, \alpha, \varepsilon)$* only if *the following three inequalities are satisfied*

- $k_a \ge h(\alpha)n - o(n)$,
- $k_b \ge h(\alpha)n - o(n)$,
- $k_a + k_b \ge (1 + h(\alpha))n - o(n)$,

*where $h(\alpha) := -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ is Shannon's entropy function.*

Let us note that Theorem 1 holds also for the model with public randomness.

The asymptotic version of these conditions is shown in Fig. 2: the points in the red area below the dashed lines are *not achievable*. Notice that this bound is very similar to the Slepian–Wolf bound known for the classic probabilistic setting, [1]. The correspondence is quite straightforward: in Theorem 1 the sum of lengths of two messages is lower-bounded by the "combinatorial entropy of the pair" $(1 + h(\alpha))n$, which is basically the logarithm of the number of possible pairs $(X, Y)$ with the given Hamming distance; in the classic Slepian–Wolf theorem the sum of two channel capacities is bounded by the Shannon entropy of the pair. Similarly, in Theorem 1 the lengths of both messages are bounded by $h(\alpha)n$, which is the "combinatorial conditional entropy" of $X$ conditional on $Y$ or $Y$ conditional on $X$, i.e., the logarithm of the maximal number of $X$'s compatible with a fixed $Y$ and vice-versa; in the classic Slepian–Wolf theorem these quantities are bounded by the two conditional Shannon entropies.

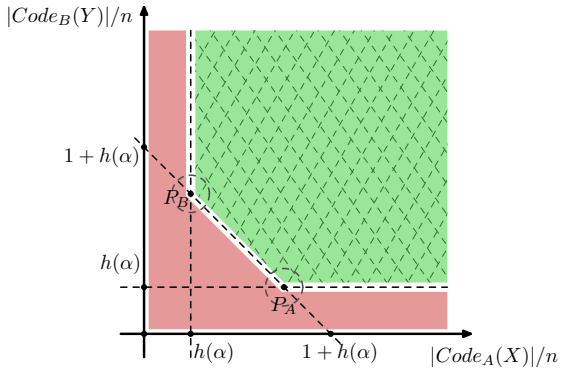


**Fig. 2.**

For the deterministic version of the combinatorial Slepian–Wolf encoding problem, the complete characterisation of the set of achievable pairs remains unknown. Only some partial (negative) results are proven in [11], see also a discussion in [8]. Namely, it is proven that in some $\Theta(n)$-neighborhood of points $(n, h(\alpha)n)$ and $(h(\alpha)n, n)$ (i.e., in the dashed circles around points $V_A$ and $V_B$ in Fig. 2) there are no achievable pairs. Hence, for the case $\varepsilon = 0$ the bound from Theorem 1 does not provide the exact characterisation of the set of achievable pairs. This result is in sharp contrast with the classic Slepian–Wolf coding in the probabilistic setting.

The case of randomised protocols for this communication problem is somewhat simpler. It is known that the sufficient conditions for achievable pairs are very close to the bound from Theorem 1. More precisely, for every $\varepsilon > 0$, all pairs in the hatched green area in Fig. 2 are achievable for the combinatorial Slepian–Wolf problem with parameters $(n, \alpha, \varepsilon)$, see [8]. The gap between known necessary and sufficient conditions (the red and green area in the figure) is negligibly small. This result is similar to the classic Slepian–Wolf theorem.

An annoying shortcoming of the result in [8] was computational complexity. The protocols in [8] require exponential computations on the senders and the receiver sides. In this paper we improve computational complexity without degrading communication complexity. We propose a communication protocol with (i) optimal trade-off between the lengths of senders messages and (ii) polynomial time algorithms for all parties. Technically, we prove the following theorem[4]:

**Theorem 2 (main result).** *There exists a real $d > 0$ and a function $\delta(n) = o(n)$ such that for all $0 < \alpha < 1/2$ and all integers $n$, every pair $(k_a, k_b)$ that satisfies three inequalities*

- $k_a \geq h(\alpha)n + \delta(n)$,
- $k_b \geq h(\alpha)n + \delta(n)$,
- $k_a + k_b \geq (1 + h(\alpha))n + \delta(n)$,

*is achievable for the combinatorial Slepian–Wolf coding problem with parameters $(n, \alpha, \varepsilon(n) = 2^{-\Omega(n^d)})$ (in the standard communication model with private sources of randomness). Moreover, all computations in the communication protocol can be done in polynomial time.*

Protocols achieving the marginal pairs $(n, h(\alpha)n + o(n))$ and $(h(\alpha)n + o(n), n)$ (even for poly-time protocols) were originally proposed in [5] and later in [7]. In our paper we generalise these results: we construct effective protocols for all points in hatched green area in Fig. 2. In fact, our construction follows the ideas proposed in [5] and later used in [6].

Our argument employs the following technical tools: reduction of one global coding problem with strings of length $n$ to many local problems with strings of length $\log n$ (similar to the classic technique of concatenated codes); Reed–Solomon checksums; pseudo-random permutations; universal hashing (see Appendix for more details).

In conclusion we discuss how to simplify the algorithms involved in the protocol and make the protocol more practical. The price for this simplification is a weaker bound for the probability of error.

## 2 Preliminaries

**Notation:**

- We denote by $\mathrm{dist}(v, w)$ the Hamming distance between binary strings $v$ and $w$.
- For an $n$-bits string $X = x_1 \dots x_n$ and a tuple of indices $I = \langle i_1, \dots, i_s \rangle$ we denote $X_I := x_{i_1} \dots x_{i_s}$.

**Pseudo-random permutations:** A distribution on the set $S_n$ of permutations of $\{1, \dots, n\}$ is called *almost $t$-wise independent* if for every tuple of indices $1 \leq i_1 < i_2 < \dots < i_t \leq n$, the distribution of $(\pi(i_1), \pi(i_2), \dots, \pi(i_t))$ for $\pi$ chosen according to this distribution has distance at most $2^{-t}$ from the uniform distribution on $t$-tuples of $t$ distinct elements from $\{1, \dots, n\}$.

[4] Not surprisingly, the statements of Theorem 1 and Theorem 2 are very similar. The gap between necessary and sufficient conditions for achievable pairs is only $o(n)$.

**Proposition 1 ([4]).** *For all $1 \le t \le n$, there exists $T = O(t \log n)$ and an explicit map $\Pi : \{0,1\}^T \to S_n$, computable in time $\mathrm{poly}(n)$, such that the distribution $\Pi(s)$ for random $s \in \{0,1\}^T$ is almost $t$-wise independent.*

## 3 Auxiliarely communication models: shared and imperfect randomness

The complete proof of Theorem 2 involves a few different technical tricks. To make the construction more modular and intuitive, we split it in several possibly independent parts. To this end, we introduce several auxiliary communication models. The first two models are somewhat artificial; they are of no independent interest make sense only as intermediate steps of the proof of the main theorem. Here is the list of our communication model:

**Model 1.** **The model with partially shared sources of perfect randomness:** Alice and Bob have their own sources of independent uniformly distributed random bits. Charlie has a free access to Alice's and Bob's sources of randomness (these random bits are not included in the communication complexity); but Alice and Bob cannot access the random bits of each other.

**Model 2.** **The model with partially shared sources of $T$-non-perfect randomness:** Alice and Bob have their own (independent of each other) sources of randomness. However these sources are not perfect: they can produce $T$-independent sequences of bits and *$T$-wise almost independent* permutations on $\{1, \dots, n\}$. Charlie has a free access to Alice's and Bob's sources of randomness, while Alice and Bob cannot access the random bits of each other.

**Model 3.** **The standard model with private sources of perfect randomness (our principal model).** In this model Alice and Bob have their own sources of independent uniformly distributed random bits. Charlie cannot access random bits of Alice and Bob unless they include these bits in their messages.

In all these models the profile of achievable pairs of rates is the same as in Theorem 1 (the green area in Fig. 2).

We start with an effective protocols for Model 1, and then extend it to Model 2, and at last to Model 3.

## 4 An effective protocol for Model 1 (partially shared sources of perfect randomness)

In this section we show that all pairs of rates from the green area in Fig. 2 are achievable for Model 1. Technically, we prove the following statement.

**Proposition 2.** *The version of Theorem 2 holds for the Communication Model 1.*

*Remark 1.* Our protocol involves random objects of different kinds: randomly chosen permutations and random hash functions from a universal family. In this section we

assume that the used randomness is perfect. This means that all permutations are chosen with the uniform distribution, and all hash functions are chosen independently.
*Remark 2.* Our protocol involves universal families of hash functions; see Appendix for the standard properties of universal hashing.

### 4.1 Parameters of the construction

Our construction has some "degrees of freedom"; it involves several parameters, and values of these parameters can be chosen in rather broad intervals. In what follows we list these parameters, with some short comments.

- $\lambda$ is any fixed number between 0 and 1 (this parameter controls the ratio between the lengths of messages sent by Alice and Bob);
- $\kappa_1, \kappa_2$ (some absolute constants that control the asymptotic of communication complexity hidden in the $o(\cdot)$-terms in the statements of Theorem 2 and Proposition 3);
- $k(n) = \log n$ (we will cut strings of Alice and Bob in "blocks" of length $k$; we can afford the brute force search over all binary strings of length $k$, since $2^k$ is polynomial in $n$);
- $m(n) = n/k(n)$ (when we split $n$-bits strings into blocks of length $k$, we get $m$ blocks);
- $r(n) = O(\log k) = O(\log\log n)$ (this parameter controls the chances to get a collision in hashing; we choose $r(n)$ so that $1 \ll r(n) \ll k$);
- $\delta(n) = k^{-0.49} = (\log n)^{-0.49}$ (the threshold for deviation of the relative frequency from the probability involved in the law of large numbers; notice that we choose $\delta(n)$ s.t. $\frac{1}{\sqrt{k}} \ll \delta(n) \ll k$);
- $\sigma = \Theta(\frac{1}{(\log n)^c})$ for some constant $c > 0$ ($\sigma n$ is the length of the Reed-Solomon checksum; we chose $\sigma$ such that $\sigma \to 0$);
- $t$ (this parameter characterise the quality of the random bits used by Alice and Bob; accordingly, this parameter is involved in the law(s) of large numbers used to bound the probability of the error; we let $t(n) = m^c$ for some $c > 0$).

### 4.2 The scheme of the protocol

**Alice's parts of the protocol:**

($1_A$) Select at random a tuple of $\lambda n$ indices $I = \{i_1, i_2, \ldots, i_{\lambda n}\} \subset \{1, \ldots, n\}$. Technically, we may assume that Alice chooses at random a permutation $\pi_I$ on the set $\{1, 2, \ldots, n\}$ and lets $I := \pi_I(\{1, 2, \ldots, \lambda n\})$.
($2_A$) Send to the receiver the bits $X_I = x_{i_1} \ldots x_{i_{\lambda n}}$.
($3_A$) Choose another random permutation $\pi_A : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and permute the bits of $X$, i.e., let[5] $X' = x'_1 \ldots x'_n := x_{\pi_A(1)} \ldots x_{\pi_A(n)}$. Further, divide $X'$ into blocks of length $k(n)$, i.e., represent $X'$ as a concatenation $X' = X'_1 \ldots X'_m$, where $X'_j := x'_{(j-1)k+1} x'_{(j-1)k+2} \ldots x'_{jk}$ for each $j$.

[5] In what follows we consider also the $\pi_A$-permutation of bits in $Y$ and denote it $Y' = y'_1 \ldots y'_n := y_{\pi_A(1)} \ldots y_{\pi_A(n)}$. Thus, the prime in notation (e.g., $X'$ and $Y'$) implies that we permuted the bits of the original strings by $\pi_A$.

($4_A$) Then Alice computes hash values of these blocks. More technically, we consider a universal family of hash functions

$$\mathrm{hash}_l^A \ : \ \{0,1\}^k \to \{0,1\}^{h(\alpha)(1-\lambda)k+\kappa_1\delta k+\kappa_2 \log k+r}.$$

We may assume that these hash functions are indexed by bit strings $l$ of length $O(k)$, see Proposition 5. Alice choses at random $m$ indices $l_1, \ldots, l_m$ of hash functions. (We may assume that the sequence of $l_i$ is $(T/k)$-independent). Then Alice applies each $\mathrm{hash}_{l_j}$ to the corresponding block $X'_j$ and sends to Charlie the resulting hash values

$$\mathrm{hash}^A_{l_1}(X'_1), \ldots, \mathrm{hash}^A_{l_m}(X'_m).$$

($5_A$) Compute the Reed-Solomon checksums of the sequence $X'_1, \ldots, X'_m$ that are enough to reconstruct all blocks $X'_j$ if most $\sigma m$ of them are corrupted, and send them to Charlie These checksums make a string of $O(\sigma mk)$ bits, see Proposition 4.

**Bob's parts of the protocol:**

($1_B$) Choose at random permutation $\pi_B : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and use it to permute the bits of $Y$, i.e., let[6] $Y'' = y''_1 \ldots y''_n := y_{\pi_B(1)} \ldots y_{\pi_B(n)}$. Further, divide $Y''$ into blocks of length $k$, and represent $Y''$ as a concatenation $Y'' = Y''_1 \ldots Y''_m$, where $Y''_j := y''_{(j-1)k+1} y''_{(j-1)k+2} \ldots y''_{jk}$ for each $j$.

($2_B$) Then choose at random $m$ hash functions $\mathrm{hash}^B_{l_j}$ from a universal family of hash functions

$$\mathrm{hash}^B_l \ : \ \{0,1\}^k \to \{0,1\}^{(1-\lambda)k+h(\alpha)\lambda k+\kappa_1\delta\cdot k+\kappa_2 \log k+r}.$$

(we assume that $l_j$ are $(T/k)$-independent) and send to Charlie random hash values

$$\mathrm{hash}^B_{l_1}(Y''_1), \ldots, \mathrm{hash}^B_{l_m}(Y''_m).$$

Similarly to ($4_A$), we may assume that these hash functions are indexed by bit strings $l$ of length $O(k)$, see Proposition 5.

($3_B$) Compute the Reed-Solomon checksums of the sequence $Y''_1, \ldots, Y''_m$, that are enough to reconstruct all blocks $Y''_j$, if at most $\sigma m$ of them are corrupted, and send them to Charlie. These checksums should be a string of length $O(\sigma mk)$ bits, see Proposition 4.

**Charlie's parts of the protocol:**

[6] Similarly, in what follows we apply this permutation to the bits of $X$ and denote

$$X'' = x''_1 \ldots x''_n := x_{\pi_B(1)} \ldots x_{\pi_B(n)}.$$

Thus, the double prime in notation (e.g., $X''$ and $Y''$) implies that we permuted the bits of the original strings by $\pi_B$.

($1_C$) Apply Bob's permutation $\pi_B$ to the positions of bits selected by Alice, and denote the result by $I''$, i.e., $I'' = \{\pi_B(i_1), \dots, \pi_B(i_{\lambda n})\}$. Then split indices of $I''$ into $m$ disjoint parts corresponding to the different intervals $Int_j = \{(j-1)k+1, (j-1)k+2, \dots, jk\}$, and $I''_j := I'' \cap Int_j$. Further, for each $j = 1, \dots, m$ denote by $X_{I''_j}$ the bits sent by Alice, that appear in the interval $Int_j$ after permutation $\pi_B$.

($2_C$) For each $j = 1, \dots, m$ try to reconstruct $Y''_j$. To this end, find all bit strings $Z = z_1 \dots z_k$ that satisfy a pair of conditions ($\mathrm{Cond}_1$) and ($\mathrm{Cond}_2$) that we formulate below.

We abuse notation and denote by $Z_{I''_j}$ the subsequence of bits from $Z$ that appear at the positions determined by $I''_j$. That is, if $I''_j = \{(j-1)k+s_1, \cdots, (j-1)k+s_l\}$, where

$$(j-1)k + s_1 < (j-1)k + s_2 < \cdots < (j-1)k + s_l,$$

then $Z_{I''_j} = z_{s_1} z_{s_2} \dots z_{s_l}$. With this notation we can specify the required property of $Z$:

($\mathrm{Cond}_1$) $\mathrm{dist}(X_{I''_j}, Z_{I''_j}) \le (\alpha+\delta)|I''_j|$,

($\mathrm{Cond}_2$) $\mathrm{hash}^B_{l_j}(Z)$ must coincide with the hash value $\mathrm{hash}^B_{l_j}(Y''_j)$ received from Bob.

If there is a unique $Z$ that satisfies these two conditions, then take it as a candidate for $Y''_j$; otherwise (if there is no such $Z$ or if there exist more than one $Z$ that satisfy these conditions) we say that reconstruction of $Y''_j$ fails.

($3_C$) Use Reed-Solomon checksums received from Bob to correct the blocks $Y''_j$ that we failed to reconstruct or reconstructed incorrectly at step ($2_C$).

($4_C$) Apply permutation $\pi_B^{-1}$ to the bits of $Y''$ and obtain $Y$.

($5_C$) Permute bits of $Y$ and $X_I$ using permutation $\pi_A$.

($6_C$) For each $j = 1, \dots, m$ try to reconstruct $X'_j$. To this end, find all bit strings $W = w_1 \dots w_k$ such that

($\mathrm{Cond}_3$) at each position from $I' \cap Int_j$ the bit from $X'$ (in the $j$-th block) sent by Alice coincides with the corresponding bit in $W$,

($\mathrm{Cond}_4$) $\mathrm{dist}(Y'_{Int_j \setminus I'_j}, W_{Int_j \setminus I'_j}) \le (\alpha+\delta)|Int_j \setminus I'_j|$

($\mathrm{Cond}_5$) $\mathrm{hash}^A_{l_j}(W)$ coincides with the hash value $\mathrm{hash}^A_{l_j}(X'_j)$ received from Alice.

If there is a unique $W$ that satisfies these conditions, then take this string as a candidate for $X'_j$; otherwise (if there is no such $W$ or if there exist more than one $W$ satisfying these conditions) we say that reconstruction of $X'_j$ fails.

($7_C$) Use Reed-Solomon checksums received from Alice to correct the blocks $X'_j$ that were incorrectly decoded at step ($6_C$).

($8_C$) Apply permutation $\pi_A^{-1}$ to the positions of bits of $X'$ and obtain $X$.

**Lemma 1.** *In Communication Model 1, the protocol described above fails with probability at most $O(2^{-m^d})$ for some $d > 0$.*

(See the proof in Appendix.)

### 4.3 Communication complexity of the protocol.

Alice sends $\lambda n$ bits at step $(2_A)$, $h(\alpha)(1-\lambda)k + O(\delta)k + O(\log k) + r$ for each block $j = 1, \dots, m$ at step $(3_A)$, and $\sigma mk$ bits of the Reed-Solomon checksums at step $(4_A)$. So the total length of Alice's message is

$$\lambda n + (h(\alpha)(1-\lambda)k + O(\delta)k + O(\log k) + r) \cdot m + \sigma n.$$

For the values of parameters that we have chosen above, this sum can be estimated as $\lambda n + h(\alpha)(1-\lambda)n + o(n)$. Bob sends $(1-\lambda)k + h(\alpha)\lambda k + O(\delta)k + O(\log k) + r$ bits for each block $j = 1, \dots, m$ at step $(1_B)$ and $\sigma mk$ bits of the Reed-Solomon checksums at step $(2_B)$. This sums up to

$$((1-\lambda)k + h(\alpha)\lambda k + O(\delta)k + O(\log k) + r) \cdot m + \sigma n$$

bits. For the chosen values of parameters this sum is equal to $(1-\lambda)n + h(\alpha)\lambda n + o(n)$. When we vary parameter $\lambda$ between 0 and 1, we variate accordingly the lengths of both messages from $h(\alpha)n + o(n)$ to $(1+h(\alpha))n + o(n)$, while the sum of Alice's and Bob's messages always remains equal to $(1 + h(\alpha))n + o(n)$. Thus, varying $\lambda$ from 0 to 1, we move in Fig. 2 from $P_B$ to $P_A$.

It remains to notice that algorithms of all participants require only $\mathrm{poly}(n)$-time computations. Indeed, all manipulations with Reed-Solomon checksums (encoding and error-correction) can be done in time $\mathrm{poly}(n)$, with standard encoding and decoding algorithms. The brute force search used in the decoding procedure requires only the search over sets of size $2^k = \mathrm{poly}(n)$). Thus, Proposition 2 is proven.

## 5 An effective protocol for Model 2 (partially shared randomness)

In this section we prove that the pairs of rates from Fig. 2 are achievable for the Communication Model 2. Now the random sources of Alice and Bob are not perfect: the random permutations are $t$-wise almost independent and the chosen hash functions are $t$-independent (for a suitable $t$).

**Proposition 3.** *The version of Theorem 2 holds for Communication Model 2 (with parameter $T = \Theta(n^c \log n)$).*

To prove Proposition 3 we do not need a new communication protocol — in fact, the protocol for the Model 1 works for the Model 2 as well. The only difference between Proposition 2 and Proposition 3 is a more general statement about the estimation of the error probability:

**Lemma 2.** *For the Communication Model 2 with parameter $T = \Theta(n^c \log n)$ the communication protocol described in section 4 fails with probability at most $O(2^{-m^d})$ for some $d > 0$.*

(See the proof in Appendix.) Since the protocol remains the same, the bounds for the communication and computational complexity, proven in Proposition 2, remain valid in the new setting. With Lemma 2 we get the proof of Proposition 3.

## 6 The model with private sources of perfect randomness (the main model)

Proposition 3 claims that the protocol from Section 4 works well for the artificial Communication Model 2 (with non-perfect and partially private randomness). Now we want to modify this protocol and adapt it to the Communication Model 3.

Technically, we have to get rid of (partially) shared randomness. That is, in Model 3 we cannot assume that Charlie gets Alice's and Bob's random bits for free. Moreover, Alice and Bob cannot just send their random bits to Charlie (this would dramatically increase the communication complexity). However, we can use a standard idea: we require that Alice and Bob use pseudo-random bits instead of truly uniformly random bits. Alice and Bob take at random (with the truly uniform distribution) short seeds for pseudo-random generators and expand them to longer sequences of pseudo-random bits, and feed these *pseudo-random* bits in the protocol described in the previous sections. Then Alice and Bob send the random seeds of generators to Charlie (the seeds are rather short, so they do not increase communication complexity substantially); and Charlie (using the same generators) expands the seeds to the same long pseudo-random sequences and plug them in into his side of the protocol.

It remains to choose some specific pseudo-random generators that suits our plan: we need two different pseudo-random generators — one to generate indices of hash functions and another to generate permutations.

Constructing a suitable sequence of pseudo-random hash-functions is simple. Both Alice and Bob needs $m$ random indices $l_i$ of hash functions, and the size of each family of hash functions is $2^{O(k)} = 2^{O(\log n)}$. We need the property of $t$-independency of $l_i$ for $t = m^c$ (for a small enough $c$). So we can choose a random polynomial of degree $t-1$ over $\mathbb{F}_{2^{O(\log n)}}$ and take the values of this polynomial at $m$ different points of the field. Then the seed of this generator is just the tuple of all its coefficients (it consists of $O(t \log n) = o(n)$ bits).

A construction of a pseudo-random permutation is more involved. We need $t$-wise almost independent pseudo-random permutations. By Proposition 1 such a permutation can be effectively produced by a pseudo-random generator with a seed of length $O(t \log n)$ bits. Again, Alice and Bob chose seeds for random permutation at random, with the uniform distribution. The seeds of the generators involved in our protocol are much shorter than $n$, so Alice and Bob can send these seeds to Charlie without essentially increasing communication complexity.

The failure probability in Proposition 3 is bounded by $2^{-\Omega(n/\log n)^c}$, which is less than $2^{-\Omega(n^d)}$ for $d < c$. This concludes the proof of Theorem 2.

## 7 Conclusion

*Practical implementation.* The coding and decoding procedures in our protocol run in polynomial time. However, the protocol does not seem very practical (mostly due to the use of the KNR generator, which requires quite sophisticated computations). A simpler and more practical protocol can be implemented if we substitute $t$-wise almost independent permutations (KNR generator) by 2-independent permutation (e.g., a random

affine mapping). The price for this simplification is a weaker bound for the probability of error. Indeed, with 2-independent permutations we cannot employ the law of large numbers from Proposition 6; instead, we should use Chebyshev's inequality. (A similar technique was used in [7] to simplify the protocol from [5].) In this version of the protocol we can conclude that the probability of error $\varepsilon(n)$ tends to 0, but the convergence is rather slow.

*Open question*: to characterize the set of all achievable pairs of rates for deterministic communication protocols.

## References


1. D. Slepian and J.K. Wolf. *Noiseless Coding of Correlated Information Sources.* IEEE Transactions on Information Theory, 19, 471–480 (1973).
2. Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In proc. TCC 2004, pp. 446–472, 2004.
3. J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. SIAM J. Discrete Math., 8(2):223–250, 1995.
4. E. Kaplan, M. Naor, and O. Reingold. *Derandomized construction of $k$-wise (almost) independent permutation.* Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. Springer, 354–365 (2005).
5. A. Smith, *Scrambling Adversarial Errors Using Few Random Bits*, Optimal information reconciliation, and better private codes. In Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA), 395–404 (2007).
6. V. Guruswami and A. Smith, Codes for Computationally Simple Channels: Explicit Constructions with Optimal Rate, In Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS), 723–732 (2010).
7. A. Chuklin, *Effective protocols for low-distance file synchronization*, arXiv:1102.4712 (2011).
8. D. Chumbalov, *Combinatorial Version of the Slepian-Wolf Coding Theorem for Binary Strings*, Siberian Electronic Mathematical Reports, 10, 656–665 ( 2013).
9. F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.
10. L. Carter, M. Wegman, *Universal Hash Functions*, Journal of Computer and System Science, Vol. 18, pp. 143-154, 1979.
11. A. Orlitsky Interactive communication of balanced distributions and of correlated files. SIAM Journal on Discrete Mathematics, 6, 548–564 (1993).
12. A. N. Kolmogorov. Three approaches to the quantitative definition of information. Problems of information transmission, 1(1), 1–7 (1965).
13. R. V. L. Hartley. Transmission of information. Bell System technical journal, 7(3), 535–563 (1928).
14. D. Hammer, A. Romashchenko, A. Shen, and N. Vereshchagin, Inequalities for Shannon entropy and Kolmogorov complexity. Journal of Computer and System Sciences, 60(2), 442–464 (2000).
15. A. Romanshchenko, A. Shen, and N. Vereshchagin. Combinatorial interpretation of Kolmogorov complexity. Theoretical Computer Science, 271(1-2), 111–123 (2002).
16. A combinatorial approach to information inequalities , Communications in Information and Systems, 1(3), 1–14 (2001).

17. An. Muchnik, Conditional complexity and codes, Theoretical Computer Science, 271(1-2), 97–109 (2002).
18. I. Csiszar and J. Körner. Information theory: coding theorems for discrete memoryless systems. 2nd ed. Cambridge University Press (2011).

# 8 Appendix A: Technical tools

In this section we summarise the technical tools employed in the papers. Some of them (e.g., Reed–Solomon codes) are classic and well known; the other (e.g., the KNR pseudo-random generator) are somewhat more exotic.

## 8.1 Error correcting codes

**Proposition 4 (Reed-Solomon codes).** *Assume $m + 2s < 2^k$. Then we can assign to every sequence of $m$ strings $X = \langle X^1, \ldots, X^m \rangle$ (where $X^j \in \{0,1\}^k$ for each $j$) a string of* checksums $Y = Y(X)$*,*

$$Y \ : \ \{0,1\}^{km} \to \{0,1\}^{(2s+1)k}$$

*with the following property. If at most $s$ strings $X^j$ are corrupted, the initial tuple $X$ can be uniquely reconstructed given the value of $Y(X)$. Moreover, encoding (computation $X \mapsto Y(X)$) end decoding (reconstruction of the initial values of $X$) can done in time* $\mathrm{poly}(2^k)$.

*Proof of Proposition 4:* The required construction can be obtained from a systematic Reed–Solomon code with suitable parameters (see, e.g., [9]). Indeed, we can think of $X = \langle X^1, \ldots, X^m \rangle$ as of a sequence of elements in a finite field $\mathbb{F} = \{q_1, q_2, \ldots, q_{2^k}\}$. Then, we interpolate a polynomial $P$ of degree at most $m-1$ such that $P(q_i) = X_i$ for $i = 1, \ldots, m$ and take the values of $P$ at some other points of the field as checksums:

$$Y(X) := \langle P(a_{m+1}), P(a_{m+2}), \ldots, P(a_{m+2s+1}) \rangle.$$

The tuple $\langle X^1, \ldots, X^m, P(a_{m+1}), P(a_{m+2}), \ldots, P(a_{m+2s+1}) \rangle$ is a codeword of the Reed–Solomon code, and we can recover it if at most $s$ items of the tuple are corrupted. It is well known that the error-correction procedure for Reed–Solomon codes can be implemented in polynomial time.

## 8.2 Universal hashing

**Proposition 5 (universal hashing family, [10]).** *There exists a family of poly-time computable functions* $\mathrm{hash}_i : \{0,1\}^n \to \{0,1\}^k$ *such that* $\forall x_1, x_2 \in \{0,1\}^n$, $x_1 \neq x_2$ *it holds*

$$\mathrm{prob}_i[\mathrm{hash}_i(x_1) = \mathrm{hash}_i(x_2)] = 1/2^k,$$

*where index $i$ ranges over $\{0,1\}^{O(n+k)}$ (i.e., each hash function from the family can be specified by a string of length $O(n+k)$ bits).*

Parameter $k$ in Proposition 5 is called *the length of the hash*.

The following claim is an (obvious) corollary of the definition of a universal hashing family. Let $\mathrm{hash}_i(x)$ be a family of functions satisfying Proposition 5. Then for every $S \subset \{0,1\}^n$, for each $x \in S$,

$$\mathrm{prob}_i[\exists x' \in S, \text{ s.t. } x' \neq x \text{ and } \mathrm{hash}_i(x) = \mathrm{hash}_i(x')] < \frac{|S|}{2^k}.$$

This property allows to identify an element in $S$ by its hash value.

## 9 Appendix B: The law of large numbers for $t$-independent sequences

The following version of the law of large numbers is suitable for our argument:

**Proposition 6 (see [2, 3, 5]).** *Assume $\xi_1, \ldots, \xi_m$ are random variables ranging over $\{0,1\}$, each with expectation at most $\mu$, and for some $c < 1$, for every set of $t = m^c$ indices $i_1, \ldots, i_t$ we have*

$$\mathrm{prob}[\xi_{i_1} = \ldots = \xi_{i_t} = 1] \leq \mu^t.$$

*If $t \ll \mu m$, then*

$$\mathrm{prob}\left[\sum_{i=1}^{m} \xi_i > 3\mu m\right] = 2^{-\Theta(m^c)}.$$

**Lemma 3.** *(a) Let $\rho$ be a positive constant, $k(n) = \log n$, and $\delta = \delta(n)$ some function of $n$. Then for each pair of subsets $\Delta, I \subset \{1, \ldots, k\}$ such that $|\Delta| = k$ and $|I| = \rho n$, for a uniformly random or $k$-wise amost independent permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$,*

$$\mu := \mathrm{prob}_\pi\big[\ \big|\ |\pi(I) \cap \Delta| - \rho k\ \big| > \delta k\ \big] = O\left(\frac{1}{\delta^2 k}\right).$$

*(b) Let $\{1, \ldots, n\} = \Delta_1 \cup \ldots \cup \Delta_m$, where $\Delta_j$ are disjoint sets of cardinality $k$ (so $m = n/k$). Also we let $t = m^c$ (for some $c < 1$) and assume $\frac{t}{\mu m} \ll 1$. Then, for a uniformly random or a random almost $(tk)$-independent permutation $\pi$,*

$$\mathrm{prob}_\pi\Big[\ \Big|\pi(I) \cap \Delta_j\Big| > (\rho + \delta)k \text{ for at least } 3\mu m \text{ different } j\Big] = 2^{-\Theta(m^c)},$$
$$\mathrm{prob}_\pi\Big[\ \Big|\pi(I) \cap \Delta_j\Big| < (\rho - \delta)k \text{ for at least } 3\mu m \text{ different } j\Big] = 2^{-\Theta(m^c)}.$$

*Proof of Lemma 3 (a):* First we prove the statement for a uniformly independent permutations. Let $I = \{i_1, \ldots, i_{\rho n}\}$. We denote

$$\xi_s = \begin{cases} 1, \text{ if } \pi(i_s) \in \Delta, \\ 0, \text{ otherwise.} \end{cases}$$

The idea is that $\xi_s$ are almost independent. Since permutation $\pi$ is chosen uniformly, we have $\mathrm{prob}[\xi_s = 1] = |\Delta|/n = k/n$ for each $s$. Hence, $\mathbb{E}(\sum \xi_s) = \rho k$. Let us estimate the variance of this random sum.

For $s_1 \neq s_2$ we have

$$\mathrm{prob}[\xi_{s_1} = \xi_{s_2}] = \frac{k}{n} \cdot \frac{k-1}{n-1} = \left(\frac{k}{n}\right)^2 + O(k/n^2).$$

So, every two $\xi_s$ are "almost independent". We get

$$\begin{aligned}\mathrm{var}\left(\sum \xi_s\right) &= \mathbb{E}\left(\sum \xi_s\right)^2 - \left(\mathbb{E}\sum \xi_s\right)^2 \\ &= \sum_s \mathbb{E}\xi_s + \sum_{s_1 \neq s_2} \mathbb{E}\xi_{s_1}\xi_{s_2} - \left(\mathbb{E}\sum_s \xi_s\right)^2 = O(k).\end{aligned}$$

Now we apply Chebyshev's inequality

$$\mathrm{prob}_\pi\left[\, \left|\sum \xi_s - \rho k\right| > \delta k \,\right] < \frac{\mathrm{var}(\sum \xi_s)}{(\delta k)^2} = O\left(\frac{1}{\delta^2 k}\right), \tag{1}$$

and we are done.

For a $k$-wise almost independent permutation we should add to the right-hand side of (1) the term $O(2^{-k})$, which does not affect the asymptotics of the final result.

Before we prove Lemma 3 (b), let us formulate a corollary of Lemma 3 (a).

**Corollary 1.** *Let $\Delta_1, \ldots, \Delta_t$ be disjoint subsets in $\{1, \ldots, k\}$ such that $|\Delta_j| = k$ for each $j$. Then for a uniformly random or $(kt)$-wise almost independent permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$,*

$$\mathrm{prob}_\pi\left[\, \left|\pi(I) \cap \Delta_j\right| > (\rho + \delta)k \text{ for all } j\right] \leq \mu^t$$

*and*

$$\mathrm{prob}_\pi\left[\, \left|\pi(I) \cap \Delta_j\right| < (\rho - \delta)k \text{ for all } j\right] \leq \mu^t.$$

*Proof of Corollary:* (sketch) For uniform permutations it is enough to notice that that the events "*there are too few $\pi$-images of $I$ in $\Delta_j$*" are negatively correlated with each other. That is, if we denote

$$E_j := \left\{\pi \;\middle|\; \left|\pi(I) \cap \Delta_{j_1}\right| < (\rho - \delta(n))k\right\},$$

then

$$\mathrm{prob}_\pi\left[\, E_1 \,\right] > \mathrm{prob}_\pi\left[\, E_{j_1} \;\middle|\; E_2 \,\right] > \ldots \mathrm{prob}_\pi\left[\, E_{j_1} \;\middle|\; E_2 \text{ and } E_2 \,\right] > \ldots$$

It remains to use the bound from (a) for the unconditional probabilities.

Similarely to the proof of Lemma 3 (a), in the case of almost independent permutations the difference of probabilities is negligible.

*Proof of Lemma 3 (b):* Follows immediately from the Corollary 1 and Proposition 6.

## 10 Appendix C: proof of Lemma 1 and Lemma 2

We prove directly the statement of Lemma 2 (which implies of course Lemma 1).

Let us estimate probabilities of errors at each step of Charlie's part of the protocol.

Step ($1_C$): No errors.

Step ($2_C$): We should estimate probabilities of errors in reconstructing each block $Y''_j$.

*1st type error:* the number of Alice's bits $x_{i_1}, \dots, x_{i_{\lambda n}}$ that appear in the block $Y''_j$ is less than $(\lambda - \delta)k$. Technically, this event itself is not an error of decoding; but it is undesirable: we cannot guarantee success of reconstruction of $Y''_j$ if we get too few bits from Alice in this slot. Denote probability of this event (for a block $j = 1, \dots, m$) by $p_1^B$. By the law of large numbers, $p_1^B \to 0$ if $\delta \gg 1/\sqrt{k}$. This fact follows from Lemma 3(a).

*2nd type error:* 1st type error does not occur but $\mathrm{dist}(X_{I''_j}, Y_{I''_j}) > (\alpha + \delta)|I''_j|$. Denote probability of this event (for a block $j = 1, \dots, m$) by $p_2^B$. Again, by the law of large numbers, $p_2^B \to 0$ if $\delta \gg 1/\sqrt{k}$. Technically, we apply Lemma 3(a).

*3rd type error:* 1st and 2nd type errors do not occur but there exist at least two different strings $Z$ satisfying (1) and (2). We choose the length of hash values for $\mathrm{hash}_l^B$ so that this event happens with probability less than $p_3^B = 2^{-r}$. Let us explain this in more detail.

All the positions $Int_j$ are split into two sorts: the set $I''_j$ and its complement $Int_j \setminus I''_j$. For each position in $I''_j$ Charlie knows the corresponding bit from $X''$ sent by Alice. To get $Z$, we should

(i) invert at most $(\alpha + \delta)|I''_j|$ Alice's bits (here we use the fact that the 2nd type error does not occur), and
(ii) choose some bits for the positions $Int_j \setminus I''_j$ (we have no specific restrictions for these bits).

The number of all strings that satisfy (i) and (ii) is equal to

$$S_B := \sum_{s=0}^{(\alpha+\delta)|I''_j|} \binom{|I''_j|}{s} \cdot 2^{|Int_j \setminus I''_j|} = 2^{h(\alpha)\lambda k + (1-\lambda)k + O(\delta)k + O(\log k)}.$$

(In the last equality we use the assumption that the 1st type error does not occur, so $|Int_j \setminus I''_j| \le (1 - \alpha + \delta)k$.) We set the length of the hash function $\mathrm{hash}_l^B$ to

$$L_B = \log S_B + r = (1 - \lambda)k + h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r$$

(here we choose suitable values of parameters $\kappa_1$ and $\kappa_2$). Then, from Proposition 5 it follows that the probability of the 3d type error is at most $1/2^r$.

We say that block $Y_j$ is *reconstructible*, if the errors of type 1, 2, and 3 do not occur for this $j$. For each block $Y''_j$, probability to be non reconstructible is at most $p_1^B + p_2^B + p_3^B$. This sum can be bounded by some threshold $\mu_B = \mu_B(n)$, where

$\mu_B(n) \to 0$. We chose parameters $\delta(n)$ and $r(n)$ so that $\mu_B(n) = 1/(\log n)^c$ for some $c > 0$.

Since for each $j = 1, \ldots, m$ probability that $Y_j''$ is non reconstructible is less than $\mu$, we conclude that the expected number of non reconstructible blocks is less than $\mu m$. This is already good news, but we need a stronger statement — we want to conclude that with high probability the number of non reconstructible blocks is not far above the expected value.

Since random permutation in the construction are $(m^c \cdot k)$-wise almost independent and the indices of hash functions are $m^c$-independent, we can apply Proposition 6 and Lemma 3(b). We obtain

$$\mathrm{prob}[\text{the fraction of non-reconstructible blocks is greater than } 3\mu_B] = O(2^{-m^c})$$

for some $c > 0$. Note we can apply Lemma 3(b) even though the random objects in the construction (random permutations and random indices of hash functions) are chosen not quite uniformly. It is enough to require that all random permutation in the construction are $(m^c \cdot k)$-wise almost independent and the indices of hash functions are $m^c$-independent.

We conclude that on stage $(2_C)$ with probability $1 - O(2^{-m^c})$ Charlie decodes all blocks of $Y_j''$ except for at most $3\mu_B(n) \cdot m$ of them.

$(3_C)$ Here Charlie reconstructs the string $Y''$ if the number of non-reconstructible blocks $Y_j''$ (at the previous step) is less than $3\mu_B(n) \cdot m$. Indeed, $3\mu_B(n) \cdot m$ is just the number of errors that can be corrected by the Reed-Solomon checksums. Hence, probability of failure at this step is less than $O(2^{-m^c})$. Here we choose the value of $\sigma$: we let $\sigma = 3\mu$.

Steps $(4_C)$ and $(5_C)$: No errors.

Step $(6_C)$ is similar to step $(2_C)$. We need to estimate probabilities of errors in reconstructing each block $X_j'$.

*1st type error:* the number of Alice's bits $x_{i_1}, \ldots, x_{i_{\lambda n}}$ is less than $(\lambda - \delta)k$. (We cannot guarantee reconstruction of a block $X_j'$ if there are too few bits from Alice in this slot). We denote probability of this event by $p_1^A$. From Lemma 3(a) it follows that $p_1^A \to 0$ since $\delta = 1/k^{0.49} \gg 1/\sqrt{k}$.

*2nd type error:* 1st type error does not occur but $\mathrm{dist}(X'_{Int_j \setminus I_j'}, Y'_{Int_j \setminus I_j'}) > (\alpha + \delta)|Int_j \setminus I_j'|$. Denote probability of this event by $p_2^A$. Again, from Lemma 3(a) it follows that $p_2^A \to 0$ since $\delta \gg 1/\sqrt{k}$.

*3rd type error:* 1st and 2nd type errors do not occur but there exist at least two different strings $W$ satisfying $(\mathrm{Cond}_3)$ and $(\mathrm{Cond}_4)$. All the positions $Int_j$ are split into two sorts: the set $I_j'$ and its complement $Int_j \setminus I_j'$. For each position in $I_j'$ Charlie knows the corresponding bit from $X'$ sent by Alice. For the other bits Carlie already knows the bits of $Y_j'$, but not the bits of $X_j'$. To obtain $Z$, we should invert at most $(\alpha + \delta) \cdot |Int_j \setminus I_j'|$ bits of $Y'_{Int_j \setminus I_j'}$. The number of such candidates is equal to

$$S_A := \sum_{s=0}^{(\alpha+\delta)|Int_j \setminus I_j'|} \binom{|Int_j \setminus I_j'|}{s} \cdot 2^{|Int_j \setminus I_j''|} = 2^{h(\alpha)(1-\lambda)h(\alpha)k + O(\delta)k + O(\log k)}.$$

We set the length of the hash function $\mathrm{hash}_l^A$ to

$$L_A = \log S_A + r = (1-\lambda)h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r$$

From Proposition 5 it follows that the probability of the 2nd type error $p_3^A \le 1/2^r$.

We say that block $X'_j$ is *reconstructible*, if the errors of type 1, 2, and 3 do not happen. For each block $X'_j$, probability to be non-reconstructible is at most $p_1^A(j) + p_2^A(j) + p_3^A(j)$. This sum is less than some threshold $\mu_A = \mu_A(n)$, where $\mu_A(n) \to 0$. With our choice of parameters $\delta(n)$ and $r(n)$ we may assume $\mu_A(n) = 1/(\log n)^c$ for some $c > 0$.

Since the random permutations in the construction are $(m^c \cdot k)$-wise almost independent and the indices of hash functions are $m^c$-independent, we get from Proposition 6 and Lemma 3

$$\mathrm{prob}[\text{the fraction of non-reconstructible blocks is greater than } 3\mu_A] = O(2^{-m^c}).$$

Thus, with probability $1 - O(2^{-m^c})$ Charlie decodes on this stage all blocks of $X'_j$ except for at most $3\mu_A \cdot m$ of them.

Step $(7_C)$ is similar to step $(3_C)$. At this step Charlie can reconstructs $X'$ if the number of non-reconstructible blocks $X'_j$ (at the previous step) is less than $3\mu_A \cdot m$ (this is the number of errors that can be corrected by the Reed-Solomon checksums). Hence, the probability of failure at this step is less than $O(2^{-m^c})$.

Step $(8_C)$: No errors at this step.

Thus, with probability $1 - O(2^{-m^d})$ (for some $d < c$) Charlie successfully reconstructs strings $X$ and $Y$.