Oblivious Deletion Codes

Roni Con* , Ray Li†

June 24, 2025

Abstract

We construct deletion error-correcting codes in the *oblivious model*, where errors are adversarial but oblivious to the encoder's randomness. Oblivious errors bridge the gap between the adversarial and random error models, and are motivated by applications like DNA storage, where the noise is caused by hard-to-model physical phenomena, but not by an adversary.

- (Explicit oblivious) We construct t oblivious deletion codes, with redundancy $\sim 2t \log n$, matching the existential bound for adversarial deletions.
- (List decoding implies explicit oblivious) We show that explicit list-decodable codes yield explicit oblivious deletion codes with essentially the same parameters. By a work of Guruswami and Håstad (IEEE TIT, 2021), this gives 2 oblivious deletion codes with redundancy $\sim 3 \log n$, beating the existential redundancy for 2 adversarial deletions.
- (Randomized oblivious) We give a randomized construction of oblivious codes that, with probability at least $1-2^{-n}$, produces an efficiently encodable/decodable code correcting t oblivious deletions with redundancy $\sim (t+1) \log n$, beating the existential adversarial redundancy of $\sim 2t \log n$.
- (Randomized adversarial) Studying the oblivious model can inform better constructions of adversarial codes. The same technique produces, with probability at least $1 - 2^{-n}$, an efficiently encodable/decodable code correcting t adversarial deletions with redundancy $\sim (2t+1) \log n$, nearly matching the existential redundancy of $\sim 2t \log n$.

The common idea behind these results is to reduce the hash size by modding by a prime chosen (randomly) from a small subset, and including a small encoding of the prime in the hash.

*Department of Computer Science, Technion–Israel Institute of Technology. roni.con93@gmail.com. This work was supported by the European Union (DiDAX, 101115134). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

†Math & CS Department, Santa Clara University. Email: rli6@scu.edu. Research supported by NSF grant CCF-2347371

Contents

1 Introduction

Error-correcting codes (hereafter just codes) play a fundamental role in ensuring the reliability and integrity of data in modern communication systems. As information is transmitted or stored, it is often subject to noise, interference, or other disruptions that can lead to errors in the data. Codes add redundant information to the original data, allowing them to detect and, in many cases, correct errors without the need for retransmission.

Tremendous progress has been made over the decades — since the pioneering works of Shannon [Sha48] and Hamming [Ham50] — on the design of efficient codes that can recover from bit flips and erasures, under both probabilistic and adversarial noise models. Deletions — errors that remove a symbol from then transmitted word — are another important and well-studied type of error. Although there has been substantial advancement in recent years in understanding deletions errors, our comprehension of this model still lags behind that of codes for correcting erasures and flips.

The main challenge in correcting deletion errors lies in the loss of synchronization between the sender and the receiver. Consider the codeword 1001011. If deletions occur at positions 2 and 5, the received word becomes 10111. Unlike the erasure setting (where the received word in this case would be 1?01?11, explicitly indicating the missing positions), here, we *do not* know where the errors occurred. Moreover, due to the nature of deletions, most bits are shifted from their original positions, making it even harder to recover the original message.

The problem of correcting deletions has been primarily studied for two error models: adversarial and randomized. In the adversarial setting, the error channel is an adversary that sees the entire codeword, has unlimited computational power, and applies a fixed number of deletions. In the randomized setting, the channel deletes every bit of the codeword independently with some fixed probability. We refer the reader to the following excellent surveys for results in both settings [Mit08, MBT10, CR20, HS21].

In this work, we focus on the *oblivious model*, where t deletions are arbitrary but oblivious to the encoded codeword. Viewed another way, the deletions are applied by an adversary that knows the message but not the codeword. For deterministic codes, the oblivious model is equivalent to the adversarial model, but if the encoding may be randomized, the oblivious model may permit less redundancy and more error tolerance.

Definition 1.1 (Oblivious Deletions Code)**.** A stochastic code with randomized encoding function Enc and (deterministic or randomized) decoding function Dec is a t *oblivious deletion code with error* ε if for all messages m and deletion patterns τ applying at most t deletions, we have that

$$\mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) \neq m] \leq \varepsilon \ ,$$

The randomness of the encoder is private to the encoder and not known to the decoder.

In this paper, we study oblivious deletion-correcting codes in the regime where the number of deletions, t, is a constant independent of the code length, n. We believe that this study of oblivious deletions is important and timely for several reasons.

1. First, the oblivious model is a natural model that bridges the gap between the adversarial and random models. For more well-studied types of errors like bit-flips and

erasures, the oblivious model arises naturally and is well studied. For bit-flip and erasure errors, the oblivious channel is a specific instance of a broader class of channels known as arbitrarily varying channels (AVCs), which have been extensively studied in information theory (see the survey by Lapidoth and Narayan [LN98]). For bit-flip and erasure errors, we know their capacities for oblivious channels are $1 - H(p)$ and $1 - p$ [CN88, Lan08][1], matching the capacity for random channel, and there are explicit and efficient constructions [GS16]. Thus, even though oblivious errors represent a stronger error model than random errors, reliable communication is still achievable at the same rate in both cases. Our results, along with those in [GL20], demonstrate a similar story for oblivious deletions. Thus, while the errors in the oblivious model are applied adversarially, coding for oblivious errors is often more similar to coding for random errors. In this sense, the model bridges the adversarial and random error models.

2. Second, the oblivious model is well-motivated for DNA storage, one of the main motivations for deletion codes. In DNA storage systems, digital data is encoded and stored in DNA molecules sequences (with a process called synthesizing), utilizing the high density storage capacity, durability, and longevity of DNA for long-term data preservation. Deletions appear naturally in DNA-based storage and thus codes that can correct them are very much needed to ensure the reliability of the stored data. The nature of errors in DNA-based storage systems is highly dependent on elements such as the technology that was used to synthesize (write) and sequence (read) the data and the environment in which the molecules are stored. Thus, the error model, though not adversarial, is not easily predictable. The oblivious model is ideally suited for this context, when the noise arises from a complex or poorly understood physical process (where the exact error distribution is unknown), but not by an adversary that sees the codeword and selects the worst-case deletions accordingly. For a comprehensive survey about recent advancements regarding coding for DNA-based storage systems, we refer the reader to [SKSY24].

3. Third, studying the oblivious model is timely given the barriers to progress in the adversarial model. Designing explicit codes that can correct a constant number of adversarial deletions has received significant attention in recent years (see Table 1 for a list of works), but there appear to be barriers to progress. Adversarial t-deletion-codes need redundancy at least $\sim t \log n$, but existentially the best codes only achieve redundancy $\sim 2t \log n$. This is only beaten for the $t = 1$ deletion case, where the Varshamov–Tenengolts code [VT65] achieves optimal redundancy $\sim \log n$. For $t \geq 2$, this factor-of-2 redundancy gap is a well-known barrier.

   There is also a gap between the existential construction and the best known explicit constructions. For $t = 2$ deletions, Guruswami and Håstad [GH21], improving on [GS18, SRB19], gives a construction matching the existential redundancy $\sim 4 \log n$ and closing this gap. However, for more deletions, $t \geq 3$, a gap remains. Several constructions give an asymptotically optimal redundancy of $\Theta(t \log n)$, but the best construction only achieves redundancy $\sim 4t \log n$ [SGB20]. Given these barriers, studying the oblivious model is relevant and timely.

[1] $H(\cdot)$ denotes the binary entropy function

Codes	t	Redundancy	Explicit?	Polytime?	Model
Existential bound	all t	$\sim 2t \log n$	No	No	Adversarial
VT Code [VT65]	1	$\sim \log n$	Yes	Yes	Adversarial
[GS18]	2	$\sim 8 \log n$	Yes	Yes	Adversarial
[SRB19]	2	$\sim 7 \log n$	Yes	Yes	Adversarial
[GH21]	2	$\sim 4 \log n$	Yes	Yes	Adversarial
[Bel15]	all t	$O(t^2 + t \log n)$	Yes	Yes	Adversarial (D.E)
[BGZ17]	all t	$O(t^2 \log t \log n)$	Yes	Yes	Adversarial
[HSS18, CJLW22]	all t	$O(t \log^2(n/t))$	Yes	Yes	Adversarial (D.E)
[SB20]	all t	$\sim 8t \log n$	Yes	Yes	Adversarial
[SGB20]	all t	$\sim 4t \log n$	Yes	Yes	Adversarial
Theorem 1.8	all t	$\sim (2t+1) \log n$	Yes, Random	Yes	Adversarial
Existential bound	all t	$\sim t \log n$	No	No	Oblivious
[CGK16]	all t	$O(t^2 \log n)$	Yes	Yes	Oblivious (D.E)
[BZ16]	all t	$O(t \log^2 t + t \log n)$	Yes	Yes	Oblivious (D.E)
[Hae19]	all t	$O(t \log(n/t))$	Yes	Yes	Oblivious (D.E)
Corollary 1.6	2	$\sim 3 \log n$	Yes	Yes	Oblivious
Theorem 1.4	all t	$\sim 2t \log n$	Yes	Yes	Oblivious
Theorem 1.7	all t	$\sim (t+1) \log n$	Yes, Random	Yes	Oblivious
Lower bounds	all t	$\geq t \log n$			Adv/Obliv

Table 1: t deletion codes under adversarial and oblivious errors. For readability, multiplicative factors of $1 + o(1)$ are suppressed by the $\sim$ notation. The Polytime column means polynomial time encodable and decodable, assuming t is a constant. The (D.E.) in the Model column means a result that was originally presented for the respective document exchange variant.

We demonstrate that, while the oblivious model has practical relevance like the adversarial model, we can circumvent the barriers imposed by the adversarial models and construct less redundant codes (Theorem 1.4, Corollary 1.6, Theorem 1.7). As a basic illustration of this, the above factor-of-2 gap — between the best existential redundancy and lower bound for adversarial deletions — does not exist for oblivious deletions (see Proposition 1.2 and Proposition 1.3). Further, as we demonstrate, studying the oblivious model can be a stepping stone to better constructions in the adversarial model. We found better constructions for oblivious deletion codes with a randomized construction (Theorem 1.7), and the same techniques yield better constructions of adversarial deletion codes with randomized code constructions (Theorem 1.8).

1.1 Our results

As far as we know, the oblivious-deletion model has only been explicitly studied by Guruswami and Li [GL20], who focused on the regime where the number of deletions is a constant fraction of the codeword length (another work [HER18] considered a relaxed setting where the decoder succeeds with high probability over a *uniformly random* codeword).

Oblivious deletions have also been studied implicitly in the form of an equivalent problem called *randomized document exchange* [BZ16, CGK16, Hae19] (see Related Work, Section 1.2 and Preliminaries, Section 3.1). This work considers the regime in which the number of deletions is a constant independent of the codeword length. Our results, along with prior work, are summarized in Table 1.

Existential result and lower bound. We first show the optimal redundancy for t oblivious deletion codes is $\sim t \log n$ by giving matching constructions and impossibility results. This stands in contrast to the adversarial model, where there is a significant gap between the best existential construction and impossibility result. We first show the lower bound, that any code correcting t oblivious deletions must have redundancy of at least $\sim t \log n$. We note that our lower bound is stronger than any lower bounds that may be implicit from the document exchange literature, because our lower bound holds even for non-systematic codes.

Proposition 1.2. *Let n be a large enough integer. Let C be a code with block length n that can correct t oblivious deletions with error $\varepsilon \leq 1/16$. Then, the redundancy of C is at least $t \log n - O_t(\log \log n)$.*

We complement this result by showing that a random code construction gives an oblivious code with optimal redundancy (up to some lower-order terms).

Proposition 1.3. *Let t be a constant integer and let n be a large enough integer that does not depend on t. Let $\varepsilon \in (0, 1)$. Then, there exists a t oblivious deletion code with error ε, redundancy $t \log n + O_{t,\varepsilon}(\log \log n)$ and the amount of randomness the encoder uses is $\log \log n + O_{t,\varepsilon}(1)$.*

Explicit oblivious deletion codes. Next, we turn to explicit constructions.

Theorem 1.4. *There exists t oblivious deletion code with error ε and redundancy $2t \log n + O_t((\log \log n)^2 + \log \frac{1}{\varepsilon})$, encoding time $\tilde{O}(n)$ and decoding time $\tilde{O}(n^{t+1})$.*

This gives an explicit t oblivious deletion code that achieves a redundancy within a factor-of-2 of optimal. Our construction also matches (up to lower order terms) the existential redundancy $\sim 2t \log n$ of adversarial deletion codes, which is not known to be achievable by explicit adversarial codes for $t \geq 3$.

List-decoding implies oblivious. Our next result connects deletion list decodable codes with oblivious deletions, showing that an explicit deletion list-decodable code gives an explicit oblivious deletion code.

Theorem 1.5. *Given any code that is (t, L) list-decodable against deletions with encoding and decoding times T_{Enc} and T_{Dec}, and redundancy $r_{\text{list}} = r_{\text{list}}(n)$, we can construct in deterministic $\operatorname{poly} \log n$ time a t oblivious deletion code with error ε that has redundancy $r_{\text{list}}(n) + O_t(\log(L/\varepsilon) + \log \log n)$, is encodable in time $T_{\mathsf{Enc}} + \tilde{O}_t(n)$ and decodable in time $T_{\mathsf{Dec}} + \tilde{O}_t(Ln)$.*

We can combine this with a result of Guruswami and Håstad [GH21] that gives codes with redundancy $3\log n + O(\log\log n)$ that are list-decodable against 2 deletions with list size 2. This yields oblivious deletion codes against 2 deletions with similar redundancy.

Corollary 1.6. *There exists an explicit 2 oblivious deletion code with error ε, redundancy $3\log n + O(\log\log n)$, and encoding and decoding times $\tilde{O}(n)$.*

Observe that this 2 oblivious deletion code has redundancy that surpasses the existential bound in the adversarial case. Namely, while the best known (non-explicit) 2-deletion correcting code has redundancy $\sim 4\log n$, we achieve explicit and efficient 2 oblivious deletion codes with redundancy $\sim 3\log n$.

Theorem 1.5 further motivates studying list-decoding against deletions. We ask the obvious follow-up question, which would imply explicit oblivious deletion codes beyond the adversarial existential bound for all constant t:

Question. Are there explicit codes list-decodable against t deletions with constant list size and redundancy $\sim c\log n$ for $c < 2t$, or, ideally, $c = t$?[2]

Randomized Explicit constructions approaching the existential bound. Our next results benefit from the power of randomness when constructing the code. Specifically, we show that there is a randomized process that, with probability $1-2^{-\Omega(n)}$, produces codes that are t oblivious deletion codes with redundancy $\sim (t+1)\log n$, almost matching the optimal redundancy $\sim t\log n$, and substantially beating the adversarial existential redundancy $\sim 2t\log n$.

Theorem 1.7. *For all positive integers t and $\varepsilon \in (0,1)$, for n sufficiently large, there exists a randomized construction that, in time $\tilde{O}(n)$, with probability $1-2^{-8n}$, produces a t oblivious deletion code with decoding error ε, redundancy $(t+1)\log n + O(\log\log n) + O(\log(1/\varepsilon))$, encoding time $\tilde{O}(n)$ and decoding time $\tilde{O}(n^{t+1})$.*

Next, we show that the same randomized construction also gives a t-adversarial deletion code that almost matches the adversarial existential redundancy $\sim 2t\log n$.

Theorem 1.8. *For all positive integers t, and n sufficiently large, there exists a randomized construction that, in time $\tilde{O}(n)$, with probability $1-2^{-8n}$, produces a t-adversarial deletion code with redundancy $\sim (2t+1)\log n$, encoding time $\tilde{O}(n)$ and decoding time $\tilde{O}(n^{t+1})$.*

Remark 1.9. Our oblivious code constructions in Theorem 1.4 and Theorem 1.7 enjoy two additional properties. First, they are *systematic*, meaning that for every message m, the encoded codeword c begins with m in its first $|m|$ bits (see Definition 3.2 for a formal definition). Second, the decoding algorithm of these codes never outputs an incorrect message. Formally, for every message m and every deletion pattern τ, we have that

$$\mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) \in \{m, \perp\}] = 1 \quad \text{and} \quad \mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) = \perp] \leq \varepsilon\,.$$

We observe that the codes in Theorem 1.8 is also systematic, but the codes given in Proposition 1.3 and Corollary 1.6 are not systematic.

[2]We note that in [HER19, Theorem 2], the authors construct list-decodable codes against t deletions with list size $O_t(1)$ and redundancy $\sim 2t\log n$. Combining with Theorem 1.5, we get another construction of t oblivious deletion with redundancy $\sim 2t\log n$, as in Theorem 1.4.

1.2 Related work

We now describe additional related works.

Document exchange. Deletion codes are, up to lower order terms in the redundancy, equivalent to the (single-round) document exchange problem [Orl91]. In the document exchange problem Alice and Bob have documents represented by strings x and x', respectively. Neither knows the others' document, but they are promised the documents are "similar." Alice's goal is to send a *hash* (or *summary*) to Bob so that Bob can learn x. The main question is: how long does the hash need to be (as a function of the similarity) for Bob to learn x (efficiently).

Most relevant to deletion codes is the setting where Bob's document x' is a subsequence of Alice's document x obtained by t deletions, but the problem is studied more generally when x and x' are at bounded edit distance or hamming distance. The edit distance case is equivalent to our setting up to constant multiplicative factors in the hash size.

If Alice's hash is a deterministic function of her string x (this is *deterministic document exchange*), the optimal hash size is equal to, up to lower order terms in the redundancy, the optimal redundancy for correcting t adversarial deletions (see [Hae19, CJLW22]). If Alice's hash is a randomized function (with private randomness) and Bob recovers with high probability (this is *randomized document exchange*), the optimal hash size is equal to, up to lower order terms in the redundancy, the optimal redundancy for correcting t oblivious deletions. The randomized–oblivious equivalence holds for the same reason as the deterministic–adversarial equivalence, but we justify it for completeness in Section 3.1: in short, we can encode the hash itself in a t-adversarial-deletion code at negligible cost (when $t \ll n$).

The document exchange problem has been the subject of extensive research [BGM88, Orl91, BL91, AGEA94, CPSV00, SNT04, IMS05, Jow12, Bel15, BZ16, CGK16, Hae19, CJLW22]. Orlitsky showed that the hash needs to have length $\Omega(t \log(n/t))$ bits. He also showed that one can compute in *exponential time* a hash of $\Theta(t \log(n/t))$ bits. For deterministic document exchange, Belazzougui [Bel15] achieved a hash of size $\Theta(t^2 + t \log^2 n)$. Later, two independent works [Hae19, CJLW22] achieved a hash of size $\Theta(t \log^2(n/t))$ which is near optimal. The problem of designing an efficient deterministic document exchange protocol with optimal hash size $\Theta(t \log(n/t))$ remains open for $t = \Omega(n)$. For randomized document exchange, [CGK16, BZ16] achieved hash size $\Theta(t^2 \log n)$ and $\Theta(t \log^2 t + t \log n)$, respectively. Later, Haeupler [Hae19] achieved optimal (randomized) hash size $\Theta(t \log(n/t))$.

Codes against oblivious errors and stochastic codes Several works have studied codes that correct from oblivious substitution and erasure errors. Langberg [Lan08] was the first to explicitly define *oblivious* channels, though the oblivious channel is a specific instance of a broader class of channels known as arbitrarily varying channels (AVCs), which have been extensively studied in information theory (see the survey by Lapidoth and Narayan [LN98]). The capacities of the oblivious substitution channel and erasure channels are $1 - H(p)$ and $1 - p$, respectively [CN88]. In [GS16], Guruswami and Smith provided the first explicit and efficient codes for the oblivious substitution channel that achieve capacity.

We re-emphasize that codes for the oblivious errors must be stochastic to achieve better

rates than for adversarial errors. Stochastic codes have also found applications in the design of codes against channels that are computationally bounded. Lipton [Lip94], Micali et al. [MPSW05], Chen et al. [CJL15], Guruswami and Smith [GS16], and Shaltiel and Silbak [SS21a, SS21b, SS22, SS24] studied various computationally bounded channels: some defined by polynomial-size circuits, others by online or space-bounded constraints.

The binary deletion channel. The binary deletion channel with parameter p (BDC_p) is the most natural channel that models random deletions. More specifically, in this channel, each bit is deleted independently at random with probability p. Tremendous efforts were put on determining the capacity of the BDC_p and the reader is referred to the excellent surveys [Mit08, CR20] and references therein.

Since our interest in this paper is on correcting a constant number of oblivious deletions, a closely related question is correcting t random deletions. Thus, of particular interest to our question is the capacity of the binary deletion channel in the regime where the deletion probability goes to 0. This question was addressed in two independent papers [KMS10, KM13] where it was shown that when $p \to 0$, the capacity of the BDC_p is $1-(1-o(1))H(p)$ (where the $o(1)$ goes to 0 as $p \to 0$).

List decoding vs other error models. Theorem 1.5 shows a connection between list-decoding deletions and oblivious deletions and further motivates the study of list-decoding against deletions. Like oblivious errors, list-decoding can be seen as a bridge between the adversarial and random error models, and exploring the connections between these various error models is a fundamental question. Along these lines, one recent work [PSW24] answered a long-standing question by showing that list-decoding from adversarial substitutions is closely linked to random substitutions (known as the symmetric channel): any capacity-achieving list-decodable code with sufficiently large Hamming distance also achieves capacity on the symmetric channel.

1.3 Organization

In Section 2, we give an overview of the main technical ideas in our constructions. In Section 3, we give some preliminaries for our constructions. In Section 4, we show Proposition 1.2 and Proposition 1.3, that the optimal redundancy for t oblivious deletions is $\sim t \log n$. In Section 5, we show Theorem 1.4, giving explicit t oblivious deletion codes with redundancy $\sim 2t \log n$. In Section 6, we show Theorem 1.5, that list-decodable deletion codes yield oblivious deletion codes. In Section 7, we show Theorem 1.7 and Theorem 1.8, our randomized constructions of oblivious deletion codes and adversarial deletion codes.

2 Technical Overview

Our main idea for obtaining t oblivious deletion codes, which is based on [SGB20], is to take t-adversarial-deletion hashes, modulo a random prime. We first illustrate this with Theorem 1.4, giving explicit t oblivious deletion codes with redundancy $\sim 2t \log n$, then sketch our other results, which are variations on this idea.

By the equivalence between randomized document exchange and systematic oblivious deletion codes (Lemma 3.4), it suffices to construct randomized document exchange protocols correcting t deletions with similar redundancy. Equivalently, it suffices to construct a systematic code — a code whose encoding is $m \mapsto m \circ h(m)$ for some *hash* $h(m)$ — for the easier setting that the hash $h(m)$ is transmitted noiselessly and only the message is corrupted. The equivalence holds because we can encode the hash in a t deletion code at negligible cost (see proof of Lemma 3.4).

Our randomized document exchange hash $h(\cdot)$ is computed as $h(m) = (h_{\text{uniq}}(m) \mod p, p)$, where $h_{\text{uniq}} : \{0,1\}^n \to \{0,1\}^{r_{\text{uniq}}}$ is a deterministic document exchange hash with reasonable (but not necessarily optimal) redundancy $r = \operatorname{poly}\log n$, and where p is a prime sampled uniformly at random from a set of $\tilde{O}(n^t)$ primes. By the prime number theorem, we can sample our prime from primes $p \le \tilde{O}(n^t)$, so each of $(h_{\text{uniq}}(m) \mod p)$ and p can be stored in $\sim t \log n$ bits, for a total redundancy of $\sim 2t \log n$. The decoder is the trivial decoder that, given a received word z and a hash (g, p), computes the hash $h_{\text{uniq}}(m') \mod p$ of each supersequence m' of the received word z, and outputs the (hopefully) unique supersequence m' with the matching hash mod p.

Now we see why this is correct. For any message m, and any deletion pattern τ, there are at most n^t supersequences of the received word $\tau(m)$. Importantly, the randomness of the hash is independent of τ. Identify the output of h_{uniq} with a number in $[2^{r_{\text{uniq}}}]$. Any of the n^t supersequences of $\tau(m)$ have pairwise distinct hashes by the definition of deterministic document exchange. Thus, by the Chinese Remainder Theorem, the hash $h(m)$ collides with any other hash on at most r_{uniq} primes; that is, for any $m' \neq m$, $h_{\text{uniq}}(m) - h_{\text{uniq}}(m') \equiv 0 \mod p$ for at most r_{uniq} distinct primes. Hence, there are at most $n^t \cdot r_{\text{uniq}}$ "bad" primes that cause a collision involving the hash $h_{\text{uniq}}(m)$. Thus, if we sample a prime from a set of at most $n^t \cdot r_{\text{uniq}}/\varepsilon$ primes, we avoid all bad primes with probability $1 - \varepsilon$, so our decoding correctly returns the message m with probability $1 - \varepsilon$. This is true for all m and τ, so we have our oblivious deletion code.

We can modify this for our other results as well.

- We now sketch the idea of Theorem 1.5, which derives an oblivious deletion code from list-decodable deletion codes. Assume for simplicity the list-decodable deletion code is systematic, with hash $h_{\text{list}} : \{0,1\}^n \to \{0,1\}^{r_{\text{list}}}$ (the non-systematic case is similar). Choose the hash $h(m) = (h_{\text{list}}(m), h_{\text{uniq}}(m) \mod p, p)$ for some p sampled randomly from a set of $\frac{L}{\varepsilon} \operatorname{polylog} n$ primes p. Now our decoder list-decodes using $h_{\text{list}}(m)$, then deduces the correct message from the list using $h_{\text{uniq}}(m) \mod p$ and p. Deducing the correct message from the list only requires distinguishing the correct message from L other messages with probability $1-\varepsilon$, which requires a negligible $O(\log(L/\varepsilon)+\log\log n)$ redundant bits.

- To obtain better redundancy with a randomized explicit construction (Theorem 1.7), we first sample a set P of $O(n/\varepsilon)$ primes from the set of all primes P_{all} smaller than $\tilde{O}(n^t)$, where ε is the decoding error.[3] This set P defines the code. Now, to encode, we sample from P rather than P_{all}. This yields redundancy $\sim (t+1)\log n$, rather than $\sim 2t \log n$, because it only takes $\log n$ bits (rather than $t \log n$ bits) to index the

[3] we actually sample from primes in $[M/2, M]$, where $M = \tilde{\Theta}(n^t)$.

prime. By concentration inequalities, for any message m and deletion pattern τ, with probability at least $1 - 2^{-10n}$, the set P contains at least $1 - \varepsilon$-fraction of primes that don't cause a collision. Union bounding over m and τ gives that the resulting code is an oblivious deletion code with overwhelming probability.

For a randomized construction of adversarial deletion codes (Theorem 1.8), we can use the same idea. This time, we sample a set of $O(n)$ primes from the set of all primes P_{all} that are at most $\tilde{O}(n^{2t})$, giving a redundancy of $\sim (2t+1)\log n$.

3 Preliminaries

All logs are base 2 unless otherwise specified. $\ln(x)$ is log base e. For an integer k, we denote $[k] = \{1, 2, \ldots, k\}$. Throughout, we identify integers with their binary representations and vise versa, so that, for example, we identify $[2^a]$ with $\{0,1\}^a$, and we may speak of $x \mod p$ for a binary string x and a prime p. We use Landau notation $O(\cdot), \Omega(\cdot), \Theta(\cdot), o(\cdot)$. We use the notation $\tilde{O}(\cdot), \tilde{\Omega}(\cdot), \tilde{\Theta}(\cdot)$ to indicate that polylog factors are suppressed, and we use the notation $O_a(\cdot), \Omega_a(\cdot), \Theta_a(\cdot)$ to indicate that multiplicative factors depending on variable a are suppressed. Throughout, the notation $\sim$ suppresses $1 \pm o(1)$ factors.

A *substring* of a string $s \in \{0,1\}^n$ is obtained by taking consecutive symbols from s while a *subsequence* of a string of s is obtained by deleting some (possibly none) of the symbols in s. A *supersequence* of a string s is another string s' which contain s as a subsequence. We write $x \sqsubseteq y$ to say that x is a subsequence of y. A *run* in a string s is a single-symbol substring of s of maximal length. Every string can be uniquely written as the concatenation of its runs. For example, if $s = 0111001$, then s is decomposed into $0 \circ 111 \circ 00 \circ 1$ where the symbol $\circ$ denotes concatenation of two strings. A *deletion pattern* is a function τ that removes a fixed subset of symbols from strings of a fixed length.

It is well known that for any $x \in \{0,1\}^{n-t}$, the number of supersequences of length n is exactly $\sum_{i=0}^{t} \binom{n}{i}$ (see, e.g., [Lev01, Equation 24]). For our purposes, we use the following crude upper bound.

Lemma 3.1. *Let n and t be integers[4] such that $2 \le t < n/2$ and let $z \in \{0,1\}^{n-t}$. The number of strings $s \in \{0,1\}^n$ such that z is a subsequence of s is at most n^t.*

Proof. We wish to show $\sum_{i=0}^{t} \binom{n}{i} \le n^t$. For $t = 2$, we can prove this bound directly, and for $t \ge 3$, we can use $\sum_{i=0}^{t} \binom{n}{i} \le (t+1)\binom{n}{t} \le (t+1)n^t/t! \le n^t$. □

3.1 Equivalence to Randomized Document Exchange

We show that systematic oblivious deletion codes are equivalent (up to lower order redundancy terms) to randomized document exchange hashes. First, we define a systematic code.

Definition 3.2 (Systematic code)**.** A code with encoding function $\text{Enc} : \{0,1\}^n \to \{0,1\}^{n+r}$ is *systematic* if, for all x, the first n bits of $\text{Enc}(x)$ are x (with probability 1).

[4]The bound is slightly incorrect for $t = 1$, but all our results are irrelevant anyway for $t = 1$, where the VT-code [VT65] achieves optimal redundancy even for adversarial errors, so we just state and use the $t \ge 2$ version in favor of a cleaner bound.

Randomized document exchange is similar to systematic codes for oblivious deletions, but we assume the redundant symbols are uncorrupted.

Definition 3.3 (Deterministic and Randomized document exchange)**.** A *deterministic* document exchange protocol for length n messages and t deletions is given by a hash function $h : \{0,1\}^n \to \{0,1\}^r$ and a decoding function $\mathrm{Dec} : \{0,1\}^{n-t} \times \{0,1\}^r \to \{0,1\}^n$ such that, for all x and all subsequences x' of length $n-t$, we can recover x from x' and $h(x)$. That is, $\mathrm{Dec}(x', h(x)) = x$.[5]

A *randomized* document exchange protocol with error ε is a document exchange protocol such that the hash function is randomized and we recover x from x' and $h(x)$ with high probability: $\mathbf{Pr}_{h(\cdot)}[\mathrm{Dec}(x', h(x)) = x] \geq 1 - \varepsilon$.

By encoding the redundant bits in a t-deletion code, we get that these two models are equivalent. This equivalence arises for the same reason that deterministic document exchange is equivalent to adversarial deletion codes (see, e.g., [Hae19, CJLW22]). We note this connection is implicit in [HER18]. For completeness, we state this equivalence in the next lemma and defer its proof to the appendix (Section 8.1).

Lemma 3.4. *[Randomized document exchange is equivalent to systematic oblivious deletions] The following hold:*

1. *Suppose we have a systematic oblivious t-deletion code with n message bits, r redundancy bits, encoding time T_E, and decoding time T_D. Then we can construct in $O(n)$ time a randomized document exchange protocol with length n, hash length r, distance t, error ε, encoding time $T_E + O(n)$, and decoding time $T_D + O(n)$.*

2. *Suppose we have a randomized document exchange protocol with length n, hash length r, distance t, error ε, encoding time T_E, and decoding time T_D. Then we can construct in $O(n)$ time a systematic oblivious t-deletion code with n message bits, $r + O(t \log(r))$ redundancy bits, encoding time $T_E + O(n + \mathrm{poly}\, r)$, and decoding time $T_D + O(n + \mathrm{poly}\, r)$.*

3.2 Concentration Inequalities

We use the following multiplicative version of the Chernoff bound.

Lemma 3.5 (Multiplicative Chernoff bound; see, e.g., [MU17])**.** *Suppose $X_1, \ldots, X_n$ are independent identically distributed random variables taking values in $\{0,1\}$. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then, for any $0 < \alpha \leq 1$:*

$$\mathbf{Pr}[X > (1+\alpha)\mu] \leq e^{-\frac{\mu\alpha^2}{2+\alpha}}$$

Also, we shall use Hoeffding's inequality.

Lemma 3.6 (Hoeffding's inequality; see, [Hoe94])**.** *Suppose that $X_1, \ldots, X_n$ are independent random variables with finite first and second moments and $a_i \leq X_i \leq b_i$ for $1 \leq i \leq n$. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then, for any $t > 0$ we have*

$$\mathbf{Pr}[X - \mu > t] < \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

[5] typically, the definition states that x' is any string with $ED(x, x') \leq t$, but this definition is equivalent up to a constant factor in t, and for our work this definition is more relevant.

3.3 Prime Number Theorem

We utilize the prime number theorem.

Theorem 3.7 ([DLVP96, Had96]). *Let $\pi(N)$ denote the number of primes less than N. Then $\lim_{n\to\infty} \frac{\pi(N)}{N/\ln N} = 1$. In other words, for all $\varepsilon > 0$, there exists N_0 such that, for all $N \geq N_0$, $\pi(N) \in [(1-\varepsilon)N/\ln N, (1+\varepsilon)N/\ln N]$.*

We use the following corollary of the prime number theorem that follows from plugging in $\varepsilon = 1/10$.

Corollary 3.8 (See also Corollary 5.2 of [Dus18]). *There exists an absolute constant N_0 such that, for all $N \geq N_0$, the number of primes between $N/2$ and N is at least $N/(10\ln N)$.*

Proof. Let N_0 be $\max(2N_0', 100)$, where N_0' is from the prime number theorem with $\varepsilon = 1/10$. Then the number of primes in $[N/2, N]$ is at least $0.9N/\ln N - 1.1(N/2)/\ln(N/2) > N/(10\ln N)$. □

3.4 The code of [Bel15]

We use as a black box the following construction of deterministic document exchange protocols. We note that this construction does not achieve the state-of-the-art hash size, however, it's decoding algorithm is linear. When the hash size of this ingredient is less important, we prefer this construction to get better decoding times.

Theorem 3.9 ([Bel15]). *There exists a deterministic document exchange protocol for t deletions with a hash of size $O(t^2 + t\log^2 n)$ and encoding and decoding time $\tilde{O}(n)$.*

4 Existential Result and Lower Bound

In this section, we prove Propositions 1.2 and 1.3. We first introduce a definition for a *deterministic* code that corrects *random deletions in the average-case*.

Definition 4.1. A deterministic code C is called *t-random deletion code (in the average-case) with error probability ε* if there exists a decoder $\mathsf{Dec} : \{0,1\}^{n-t} \to C$ such that

$$\Pr_{c,\tau}[\mathrm{Dec}(\tau(c)) \neq c] \leq \varepsilon$$

Here, the probability is over a uniformly random codeword and a uniformly random t deletion pattern.

For the sake of readability, we will write t-random deletion code to refer to t-random deletion code (in the average case).

This section is organized as follows. In Section 4.1, we show that if one has an oblivious deletion code, then there also exists a deterministic code that is a t-random deletion code. The lower bound on the redundancy of oblivious deletion codes (Proposition 1.2) is given in Section 4.2 with the assistance of a result by Kalai, Mitzenmacher, and Sudan [KMS10] that shows a lower bound on the redundancy of a t-random deletion code. In Section 4.3, we show that a random construction results in an oblivious deletion code.

4.1 Oblivious deletions $\implies$ Random deletions in the average-case

In this section, we show that if we have a t oblivious deletion code with error ε, then there exists a deterministic code C that can correct t-random deletions with error $\Theta(\sqrt{\varepsilon})$. Roughly speaking, we construct C by "sampling" the oblivious code: For each message m, include $c = \mathsf{Enc}(m)$ in the codebook. We then show that there exists such sampling for which the resulting code is a t-random deletion code.

To describe the "sampling" process, we give a formal definition of a stochastic code (recall that oblivious codes are stochastic codes, which means that their encoder is randomized).

Definition 4.2 (stochastic code)**.** A *stochastic binary code* with redundancy $r \in \mathbb{N}$, *randomness length* $b \in \mathbb{N}$ and *block length* $n \in \mathbb{N}$ is given by an encoder $\mathsf{Enc} : \{0,1\}^{n-r} \times \{0,1\}^b \to \{0,1\}^n$ and a decoder $\mathsf{Dec} : \{0,1\}^* \to \{0,1\}^{n-r} \cup \{\perp\}$.

Note that the encoder function Enc takes a message and a random string. However, we mostly write $\mathsf{Enc}(v)$ to refer to the process of sampling a random $u \leftarrow \{0,1\}^b$ and then encoding the message v with $\mathsf{Enc}(v,u)$. Nevertheless, there are some places where we explicitly write $\mathsf{Enc}(v,u)$ to indicate an encoding with a specific seed.

Claim 4.3. *Let t be a constant integer and n be a large enough integer. Assume that D is a stochastic code of length n, redundancy r, and assume that D is a t oblivious deletion code with error ε. Then, there exists a deterministic code C that can correct from t-random deletions with the same length and redundancy, and with error $\leq 2\sqrt{\varepsilon}$.*

Proof. Let D be an $(\mathsf{Enc}, \mathsf{Dec})$ stochastic code with redundancy r, length n. Define $C = \{\mathsf{Enc}(v) \mid v \in \{0,1\}^m\}$ to be a deterministic code obtained by sampling 2^m codewords via the encoder of D. Since C is a deterministic code, we shall abuse notation and refer to C as also being the encoding map from the space of messages to the codewords.

We shall define a new decoder $\mathsf{Dec}' : \{0,1\}^{n-t} \to \{0,1\}^n$ such that

$$\mathsf{Dec}'(\tau(c)) = C(\mathsf{Dec}(\tau(c))) \ .$$

Namely, the decoder of our deterministic code applies the decoder of the oblivious code to get a message and then encodes it using our sampled code C.

Enumerate all the message by $v_1, \ldots, v_{2^m}$ and their encoded codewords by $c_1, \ldots, c_{2^m} \in C$. For every deletion pattern τ, define $I_\tau = |\{i : \mathsf{Dec}'(\tau(c_i)) \neq c_i\}|$ to be the number of codewords that are incorrectly decoded by the decoder of our deterministic sampled code C.

For every τ, we have that

$$\begin{aligned}
\mathop{\mathbf{E}}_{c_1,\ldots,c_{2^m}}[I_\tau] &= \sum_{i=1}^{2^m} \mathop{\mathbf{Pr}}_{c_1,\cdots,c_{2^m}}[\mathsf{Dec}'(\tau(c_i)) \neq c_i] \\
&= \sum_{i=1}^{2^m} \mathop{\mathbf{Pr}}_{\mathsf{Enc}(v_1),\ldots,\mathsf{Enc}(v_{2^m})}[C(\mathsf{Dec}(\tau(\mathsf{Enc}(v_i)))) \neq C(v_i)] \\
&= \sum_{i=1}^{2^m} \mathop{\mathbf{Pr}}_{\mathsf{Enc}(v_1),\ldots,\mathsf{Enc}(v_{2^m})}[\mathsf{Dec}(\tau(\mathsf{Enc}(v_i)))) \notin \{v_j | C(v_j) = C(v_i)\}] \\
&\leq \sum_{i=1}^{2^m} \mathop{\mathbf{Pr}}_{\mathsf{Enc}(v_1),\ldots,\mathsf{Enc}(v_{2^m})}[\mathsf{Dec}(\tau(\mathsf{Enc}(v_i))) \neq v_i] \\
&= \sum_{i=1}^{2^m} \mathop{\mathbf{Pr}}_{\mathsf{Enc}(v_i)}[\mathsf{Dec}(\tau(\mathsf{Enc}(v_i))) \neq v_i] \\
&\leq \sum_{i=1}^{2^m} \varepsilon \\
&= \varepsilon \cdot 2^m \ ,
\end{aligned}$$

where the third equality follows since C might not be injective and therefore there can be a set of multiple messages that give the correct codeword. The first inequality follows by noting that v_i clearly belongs to that set. The fourth equality follows by noting that, according to Definition 1.1, the probability that the decoder fails is over the randomness of the encoding process of the specific message.

Thus, summing upon all deletion patterns, we get $\mathbf{E}_{c_1,\ldots,c_{2^m}}[\sum_\tau I_\tau] \leq \binom{n}{t}\varepsilon 2^m$, and thus, there exists a choice of $c_1,\ldots,c_{2^m}$ for which $\sum_\tau I_\tau \leq \binom{n}{t} \cdot \varepsilon 2^m$. Fix $c_1,\ldots,c_{2^m}$ to be such a choice. It must be that for at least $(1-\sqrt{\varepsilon})$ fraction of the deletion patterns, we have that $I_\tau \leq \sqrt{\varepsilon}2^m$. Indeed, otherwise, $\sum_{\tau: I_\tau > \sqrt{\varepsilon}} I_\tau > \sqrt{\varepsilon}\binom{n}{t} \cdot \sqrt{\varepsilon}2^m = \binom{n}{t}\varepsilon 2^m$, which is a contradiction. Now, we compute the probability that upon a random codeword and a random deletion pattern, the decoding fails. We have

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{i,\tau}[\mathsf{Dec}'(\tau(c_i)) \neq c_i] &= \mathop{\mathbf{Pr}}_{\tau}[I_\tau > \sqrt{\varepsilon}2^m] \cdot \mathop{\mathbf{Pr}}_{i}[\mathsf{Dec}'(\tau(c_i)) \neq c_i | I_\tau > \sqrt{\varepsilon}2^m] \\
&\quad + \mathop{\mathbf{Pr}}_{\tau}[I_\tau \leq \sqrt{\varepsilon}2^m] \cdot \mathop{\mathbf{Pr}}_{i}[\mathsf{Dec}'(\tau(c_i)) \neq c_i | I_\tau \leq \sqrt{\varepsilon}2^m] \\
&\leq \sqrt{\varepsilon} + (1-\sqrt{\varepsilon}) \cdot \sqrt{\varepsilon} \\
&\leq 2\sqrt{\varepsilon} \ .
\end{aligned}$$

□

4.2 Lower bound on the redundancy of oblivious deletion codes

In [KMS10], Kalai, Mitzenmacher, and Sudan studied the setting where the channel is the binary deletion channel with deletion probability p (BDC_p). In this case, every bit is deleted independently, with probability p. They showed that if C is a code that is robust against

the BDC_p, in the regime where p is small, then the rate of C is at most $1-(1-o(1))H(p)$ (where the $o(1)$ goes to 0 when $p \to 0$).

However, to prove their result, they first proved a slightly weaker theorem which contains two relaxations. First, they relax the channel to the case where there are *exactly* pn random deletions. Second, the decoding algorithm succeeds on a *random codeword* and not on every codeword. The statement of the theorem, rephrased to our setting, i.e., to the case where the number of deletions is constant (independent of the block length), is given next.

Theorem 4.4. *[KMS10, Theorem 2.2, rephrased] Let n be a large enough integer. Let C be a code with block length n that is a t-random deletion code in the average case with error probability ε. Then, the redundancy of C is at least*

$$\log \binom{n}{t} + t - \log(3t) - \log(2/(1-\varepsilon)) - O(t \log\log(n)) \ .$$

Remark 4.5. We remark that our statement is slightly different from [KMS10, Theorem 2.2]. Thus, for completeness, we provide a proof in the appendix.

We are now ready to prove Proposition 1.2 which is restated next.

Proposition 1.2. *Let n be a large enough integer. Let C be a code with block length n that can correct t oblivious deletions with error $\varepsilon \leq 1/16$. Then, the redundancy of C is at least $t \log n - O_t(\log\log n)$.*

Proof. Let C be a code with block length n that can correct t oblivious deletions with error ε. By Claim 4.3, we get that there exists a code C' with the same redundancy and block length that can correct t random deletions in the average case with error $2\sqrt{\varepsilon}$. Thus, by Theorem 4.4, the redundancy of C (and C') is at least $\log \binom{n}{t} + t - \log 3t - \log(2/(1-2\sqrt{\varepsilon})) - O(t \log\log n)$. The desired result is obtained by recalling that $\varepsilon \leq 1/16$. □

4.3 Random construction

In this section, we prove Proposition 1.3 which is stated here again, for convenience

Proposition 1.3. *Let t be a constant integer and let n be a large enough integer that does not depend on t. Let $\varepsilon \in (0,1)$. Then, there exists a t oblivious deletion code with error ε, redundancy $t \log n + O_{t,\varepsilon}(\log\log n)$ and the amount of randomness the encoder uses is $\log\log n + O_{t,\varepsilon}(1)$.*

Proof. Set $s = \varepsilon^{-2} \cdot 10t \cdot \log n$ and consider $|C|$ messages $m_1, \ldots, m_{|C|}$ where $|C| = \frac{\varepsilon}{2s} \cdot \frac{2^n}{n^t}$. For each message m_i, we associate a multi-set of codewords $E_i \in \{0,1\}^n$ of size s where every element in E_i is a uniform, random vector in $\{0,1\}^n$. The encoder and the decoder of our oblivious code are defined as follows

- $\mathsf{Enc} : \{0,1\}^{\log|C|} \to \{0,1\}^n$. A message m_i is mapped to a random element of E_i.
- $\mathsf{Dec} : \{0,1\}^{n-t} \to \{0,1\}^{\log|C|} \cup \{\perp\}$. For a received word z of length $n-t$, the decoder finds all messages m_i where z is a subsequence of a possible encoding of m_i ($z \sqsubset w$ for some $w \in E_i$). If there is only one such message, the decoder returns that message. Otherwise, the decoder return $\perp$.

A string $w \in E_i$ is called (τ, i) *bad*, if there exists $u \in E_j$ where $j \neq i$ such that $\tau(w) \sqsubseteq u$. By Lemma 3.1, the number of strings in $\{0,1\}^n$ that contain z as a subsequence is at most n^t, and thus, the probability that there exists w' in one of the E_js such that $\tau(w) \sqsubseteq w'$ is at most $|C| \cdot s \cdot \frac{n^t}{2^n} \leq \varepsilon/2$.

We say a message m_i is τ-*bad* if $|\{w \in E_i \mid w \text{ is } (\tau, i) \text{ bad}\}| > \varepsilon \cdot s$. Denote by X_w^τ the random variable indicating that w is (τ, i) bad. Observe that for $w, w' \in E_i$, X_w^τ and $X_{w'}^\tau$ are independent events due to the process that was used to choose the E_is. Thus, we can apply the Hoeffding's inequality when bounding the probability that a fixed message m_i is τ-bad for a given τ.

$$\begin{aligned}
\mathbf{Pr}[m_i \text{ is } \tau\text{-bad}] &= \mathop{\mathbf{Pr}}_{E_i}\left[\left|\{w \in E_i \mid \tau(w) \text{ is bad}\}\right| \geq \varepsilon s\right] \\
&= \mathop{\mathbf{Pr}}_{E_i}\left[\sum_{w \in E_i} X_w^\tau \geq (1+1)\frac{\varepsilon s}{2}\right] \\
&\leq e^{-2\varepsilon^2 s/4} \\
&\leq n^{-5t}
\end{aligned}$$

Let X_τ be the number of messages that are τ-bad and observe that $\mathbf{E}[X_\tau] \leq |C|/n^{5t}$. Thus, by linearity of expectation, we get that $\mathbf{E}[\sum_\tau X_\tau] = \binom{n}{t} \cdot |C|/n^{5t} \leq |C|/n^{4t}$. Therefore, there exists some choice of randomness for which the sets $E_1, \ldots E_{|C|}$ are such that $\sum_\tau X_\tau \leq |C|/n^{4t}$. For this choice of sets, we define the set of messages to be all the messages that are *not* τ bad for every τ. The number of such messages is at least $|C| - |C|/n^{4t} \geq |C|/2$, for large enough n. Therefore, the redundancy of our code is at most

$$\log(n^t) + \log 2s - \log \varepsilon + 1 \leq t \log n + \log\log n + \log t - 3\log \varepsilon + O(1) \,,$$

Now, we show that the probability of error is at most ε. In fact, we show something stronger: that the decoder never outputs a wrong message and that the probability that it outputs $\perp$ is $\leq \varepsilon$. Indeed, first observe that by the definition of the decoder, upon an input $\tau(\mathsf{Enc}(m_i)) \in \{0,1\}^{n-t}$, if there are $w \in E_i$ and $w' \in E_j$ such that $s \sqsubseteq w$ and $s \sqsubseteq w'$, then it outputs $\perp$. Since we always have that $\tau(\mathsf{Enc}(m_i)) \sqsubset w$ for some $w \in E_i$, we conclude that the decoder never outputs a wrong message. Second, for every message m_i and every t-deletion pattern τ, by our construction guarantees above, we have that $|\{w \in E_i \mid w \text{ is } (\tau, i) \text{ bad}\}| \leq \varepsilon |E_i|$. The claim follows by recalling that our encoder selects a uniform string in E_i. □

5 Explicit oblivious deletion codes with redundancy $\sim 2t \log n$

We now prove Theorem 1.4, which gives explicit oblivious deletion codes with redundancy $\sim 2t \log n$. The following lemma is the core of the construction, showing how to turn a "reasonable" deterministic document exchange protocol into a randomized document exchange protocol with redundancy $\sim 2t \log n$.

Lemma 5.1. *Suppose there exists hashes of length $f(n,t)$ for deterministic document-exchange for t deletions on words of length n with encoding time $E(n,t)$. Then there exist hashes of length $r = 2\log\binom{n}{t} + \log f(n,t) + \log\frac{100}{\varepsilon}$ for randomized document exchange with error ε. Furthermore, the encoding takes $E(n,t)+\operatorname{polylog} n$ time and the decoding takes $E(n,t)\cdot O(n^t)$ time.*

Applying this with the construction in [Bel15], which achieves $f(n,t) = O(t^2 + t\log^2 n)$ and $E(n,t) = \tilde{O}(n)$, and then using the equivalence between randomized document exchange and oblivious deletions (Lemma 3.4), we obtain Theorem 1.4.

Theorem 1.4. *There exists t oblivious deletion code with error ε and redundancy $2t\log n + O_t((\log\log n)^2 + \log\frac{1}{\varepsilon})$, encoding time $\tilde{O}(n)$ and decoding time $\tilde{O}(n^{t+1})$.*

Proof of Lemma 5.1. Let ε be the desired error. Let $M = 100n^t \cdot \frac{f(n,t)}{\varepsilon}$ and $r = 2\log M = 2t\log n + O_t(\log\log n)$. Let $h_{\text{uniq}} : \{0,1\}^n \to [2^{f(n,t)}]$ be the hash for deterministic document exchange.

Encoding. Define the hash $h : \{0,1\}^n \to \{0,1\}^r$ for our randomized document exchange protocol as follows:

- The encoding chooses a random prime number p between $M/2$ and M. The hash is $(h_{uniq}(m) \mod p, p)$.

Decoding. Suppose that z is the received word of length $n-t$ (if it is a longer subsequence, apply additional deletions arbitrarily) and (g,p) is the hash.

- By brute force search, find all supersequences m' of z such that $h_{\text{uniq}}(m') \mod p = g$. If there is exactly one such m', return m', else return $\perp$.

Correctness. Fix a message m of length n and a subsequence z of length $n-t$. Let $(h_{\text{uniq}}(m) \mod p, p)$ be the hash of m. We show that with probability $1-\varepsilon$ over the randomness of the encoder, there are no supersequences m' of z such that $h_{\text{uniq}}(m') \equiv h_{\text{uniq}}(m) \mod p$. Indeed, since $h_{\text{uniq}}(\cdot)$ is a hash for deterministic document exchange, then $h_{\text{uniq}}(m') \neq h_{\text{uniq}}(m)$ for all supersequences m' of z with $m \neq m'$. We also have $|h_{\text{uniq}}(m) - h_{\text{uniq}}(m')| \leq 2^{f(n,t)}$, which means this difference has at most $\log_{M/2} 2^{f(n,t)} \leq \frac{2f(n,t)}{\log M}$ prime factors greater than $M/2$. Since there are at least $\frac{M}{10\log M}$ prime factors between $M/2$ and M by the Prime Number Theorem (Corollary 3.8), the probability that $h_{\text{uniq}}(m) \equiv h_{\text{uniq}}(m') \mod p$ is at most $\frac{2f(n,t)/\log M}{M/(10\log M)} = \frac{20f(n,t)}{M}$. By a union bound over the n^t possible values of m' (Lemma 3.1), the probability that m is decoded incorrectly is at most $n^t \cdot \frac{20f(n,t)}{M} < \varepsilon$.

Runtime. The encoding takes time $E(n,t)$, plus the time to generate the prime p, which takes $\operatorname{poly}\log n$ if we generate the $O(M) \leq \tilde{O}(n^t)$ primes in advance. The decoding is dominated by the brute force search, which searches $\binom{n}{t}$ strings and checking each one takes $E(n,t)$ time. □

6 List decoding implies oblivious

In this section, we show that an explicit and efficient list-decodable deletion code yields an explicit and efficient oblivious code with effectively the same redundancy. We restate Theorem 1.5 for convenience

Theorem 1.5. *Given any code that is (t, L) list-decodable against deletions with encoding and decoding times T_{Enc} and T_{Dec}, and redundancy $r_{\text{list}} = r_{\text{list}}(n)$, we can construct in deterministic* $\operatorname{poly}\log n$ *time a t oblivious deletion code with error ε that has redundancy $r_{\text{list}}(n) + O_t(\log(L/\varepsilon) + \log\log n)$, is encodable in time $T_{\mathsf{Enc}} + \tilde{O}_t(n)$ and decodable in time $T_{\mathsf{Dec}} + \tilde{O}_t(Ln)$.*

Proof. We describe the encoding and decoding algorithms, then justify the correctness and runtime.

Encoding. Let $\text{Enc}_{\text{list}} : \{0,1\}^n \to \{0,1\}^{n+r_{\text{list}}}$ be the encoding of the (t, L)-list-decodable code. Let $h_{\text{uniq}} : \{0,1\}^{n+r_{\text{list}}} \to [2^{\alpha t \log^2 2n}]$ be the hash of the deterministic document exchange protocol in Theorem 3.9 on messages of length $n + r_{\text{list}}$, where $\alpha > 0$ is an absolute constant. Let $\text{Rep}_t(x) = x_1^t x_2^t \cdots x_{|x|}^t$ denote the string x where each bit is repeated t times. Note that Rep_t encodes a code that corrects $t-1$ deletions.

Let P be the set of primes in $[M/2, M]$, where $M = 100 \cdot \alpha t \cdot \frac{L}{\varepsilon} \cdot \log^2 2n$. By the Prime Number Theorem (Corollary 3.8), P has at least $\frac{M}{10 \ln M}$ primes. Define a new encoding function $\mathsf{Enc}' : \{0,1\}^n \to \{0,1\}^{n'}$, for $n' = n + r_{\text{list}} + r_{\text{rep}}$ for $r_{\text{rep}} \stackrel{\text{def}}{=} (t+1)\lceil \log |P| + \log M \rceil$ where Enc' chooses a uniformly random prime p from P, and then sets

$$\mathsf{Enc}'(m) = \mathsf{Enc}_{\text{list}}(m) \circ \text{Rep}_{t+1}(\langle p, h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) \mod p \rangle). \tag{1}$$

where $\langle p, h_{\text{uniq}}(m) \mod p \rangle$ denotes the binary representation of $(p, h_{\text{uniq}}(m) \mod p)$. Since p can be represented in $\lceil \log |P| \rceil$ bits and $h_{\text{uniq}}(m) \mod p$ can be represented in $\lceil \log M \rceil$ bits, the redundancy of this encoding is at most $r_{\text{list}} + r_{\text{rep}} \le r_{\text{list}} + O(t \log tL/\varepsilon + t \log\log n)$

Decoding. Suppose we are given a received word z. Let z_0 be the first $n + r_{\text{list}} - t$ bits of z, and let z_1 denote the last $r_{\text{rep}} - t$ bits of z. Run the list-decoding algorithm of C on z_0 to compute a list $\mathcal{L}$ of L messages. Run the repetition code decoder on z_1, and let the decoded word be $\langle p, g \rangle$ for some prime p and hash g. Then iterate through the list $\mathcal{L}$ to find messages m such that $h_{\text{uniq}}(m) \mod p = g$. If there is a unique m, return that m, else return $\perp$.

Correctness. Fix a message m and a deletion pattern τ, let $z = \tau(\mathsf{Enc}'(m))$ be the received word, and let z_0, z_1 be the substrings of z in the decoding algorithm. By construction, z_0 is a subsequence of $\mathsf{Enc}_{\text{list}}(m)$ obtained by applying t deletions. Hence, the list-decoding algorithm correctly determines a list $\mathcal{L}$ of L messages such that, (i) for each message m', the string z_0 is a subsequence of $\mathsf{Enc}(m')$, and (ii) the correct message m is in the list. Similarly, z_1 is a subsequence of $\text{Rep}_{t+1}(\langle p, h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) \mod p \rangle)$ obtained by applying t deletions, so the repetition decoder correctly determines p and $g = h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) \mod p$. For any $m' \neq m$ in the list $\mathcal{L}$, we must have $h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) \neq h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m'))$ as $\mathsf{Enc}_{\text{list}}(m)$

and $\mathsf{Enc}_{\text{list}}(m')$ are distinct length-$(|z_0|+t)$ supersequences of z_0. Further, their difference satisfies $|h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) - h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m'))| \le 2^{\alpha t \log^2 2n}$. This means they share at most $\log_{M/2} 2^{\alpha t \log^2 2n} \le \frac{\alpha t \log^2 2n}{\log(M/2)} < \frac{\varepsilon}{L} \cdot \frac{M}{10 \ln M} \le \frac{\varepsilon}{L} \cdot |P|$ common prime factors in P. The string z_0 is independent of the randomness of the encoding, so the list $\mathcal{L}$ is independent of the randomness of the encoding. Hence, a random prime from P fails to distinguish m and m' with probability at most $\frac{\varepsilon}{L}$, so the probability a random prime fails to distinguish m from all other list codewords $m' \in \mathcal{L}$ is at most ε by the union bound. Thus, we fail to recover m uniquely with probability at most ε, as desired.

Runtime. The construction takes time $\operatorname{polylog} n$ to compute the list of primes P. The encoding time is the time T_{enc} to encode in the list-decoding hash, plus the time $\tilde{O}(n)$ to encode in the document exchange hash of Theorem 3.9, plus the time $O(M)$ to choose a prime and compute the mod, for a total time of $T_{enc} + \tilde{O}(tn)$. The decoding time is the time T_{dec} to list-decode, plus the iteration to filter the list $O(Ltn) + \tilde{O}(L \cdot n)$, where we need to check for each m in the list, whether $h_{\text{uniq}}(\mathsf{Enc}_{\text{list}}(m)) \mod p = g$. □

Combining this with the 2-deletion codes of Guruswami and Håstad [GH21] gives oblivious 2-deletion codes with redundancy $\sim 3 \log n$, which is Corollary 1.6.

Theorem 6.1 ([GH21])**.** *There exist explicit codes encodable and decodable in linear time $O(n)$ with redundancy $3 \log n + O(\log \log n)$ that are list-decodable against 2 deletions with list size 2.*

Corollary 1.6. *There exists an explicit 2 oblivious deletion code with error ε, redundancy $3 \log n + O(\log \log n)$, and encoding and decoding times $\tilde{O}(n)$.*

7 Randomized explicit oblivious and adversarial codes approaching the existential bound

7.1 Randomized explicit: Oblivious with $\sim (t+1) \log n$ redundancy

We now prove Theorem 1.7, which gives a randomized construction of t oblivious deletion codes approaching the optimal lower bound and beating the adversarial existential bound.

Theorem 1.7. *For all positive integers t and $\varepsilon \in (0,1)$, for n sufficiently large, there exists a randomized construction that, in time $\tilde{O}(n)$, with probability $1 - 2^{-8n}$, produces a t oblivious deletion code with decoding error ε, redundancy $(t+1)\log n + O(\log \log n) + O(\log(1/\varepsilon))$, encoding time $\tilde{O}(n)$ and decoding time $\tilde{O}(n^{t+1})$.*

Proof. By Lemma 3.4, it suffices to give a protocol for randomized document exchange with the provided redundancy, encoding time, and decoding time. We describe the construction, encoding and decoding algorithms, then justify the correctness and runtime.

Construction. Let P_{all} be the set of all the primes in $[M/2, M]$ for $M = 100 M_0 \ln M_0$ for $M_0 = 4\alpha\varepsilon^{-1} n^t \log n$, where α is the absolute constant from Theorem 3.9. Randomly choose a multiset P of $100n/\varepsilon$ primes chosen independently at random from P_{all}. This set P specifies the code.

Encoding. Let $h_{\text{uniq}} : \{0,1\}^n \to \{0,1\}^r$ be a hash for deterministic document exchange for t deletions, where $r = \alpha t \log^2 n$ and α is an absolute constant given by Theorem 3.9 (we can omit the t^2 additive factor since we assume n is sufficiently large relative to t). Our randomized document exchange hash is $h(m,p) \stackrel{\text{def}}{=} (h_{\text{uniq}}(m) \mod p, p)$ for some p chosen uniformly at random from P. Since we know $p \in P$, we can store a description of p in $\log|P| = O(\log n) + O(\log(1/\varepsilon))$ bits, so we can store $h(m,p)$ in $\log M + \log|P| \le (t+1)\log n + O(\log\log n) + O(\log(1/\varepsilon))$ bits.

Decoding. Suppose z is a length $n-t$ received word, and suppose (g,p) is the hash. By brute force search, find all supersequences m' of z such that $h_{\text{uniq}}(m') \mod p = g$. If there is exactly one such m', return m'. Otherwise, return $\perp$.

Correctness. By the Prime Number Theorem (Corollary 3.8), P_{all} has at least $M/10\ln M > M_0$ primes.

Fix τ. We consider the probability (over the choice of P) that some message $m \in \{0,1\}^n$ is decoded incorrectly with probability at least ε (over the randomness of the encoder).

For any m, call a prime p *good* (with respect to m,τ) if our decoder recovers message m given $\tau(m)$ and $h(m,p)$ and *bad* otherwise. Since $h_{\text{uniq}}(\cdot)$ is a deterministic document exchange hash, for any of the at-most-n^t messages m' that have $\tau(m)$ as a subsequence, we have $h_{\text{uniq}}(m) \neq h_{\text{uniq}}(m')$. Since $|h_{\text{uniq}}(m) - h_{\text{uniq}}(m')| \le 2^{\alpha t \log^2 n}$, there are at most $\log_{M/2} 2^{\alpha t \log^2 n} \le \alpha \log n$ primes in P_{all} dividing $h_{\text{uniq}}(m) - h_{\text{uniq}}(m')$. By a union bound over the at-most-n^t supersequences of $\tau(m)$, at most $\frac{n^t \cdot \alpha \log n}{|P_{\text{all}}|} < \frac{\varepsilon}{2}$ fraction of the primes in P_{all} are bad (with respect to m,τ). Thus, the number of bad primes in P is distributed as a binomial distribution over $|P|$ elements with mean $\mu < \varepsilon|P|/2$. By the Chernoff bound (Lemma 3.5 with $\alpha = (\varepsilon|P| - \mu)/\mu \ge 1$), the probability that more than ε fraction of the primes in P are bad is at most $2^{-\alpha^2\mu/(2+\alpha)} \le 2^{-(\varepsilon|P|/2)\cdot\alpha/(2+\alpha)} \le 2^{-\varepsilon|P|/6} \le 2^{-10n}$ where we used $\alpha\mu \ge \varepsilon|P|/2$ and $\alpha \ge 1$. Thus, with probability at least $1 - 2^{-10n}$ over the choice of P, for a fixed deletion pattern τ and fixed m, string m is decoded correctly from $\tau(m)$ with probability at least $1-\varepsilon$. By a union bound over all messages m and all deletion patters τ, with probability at least $1 - 2^n \cdot \binom{n}{t} \cdot 2^{-10n} \ge 1 - 2^{-8n}$, for every m and τ, string m is decoded correctly from $\tau(m)$ with probability at least $1-\varepsilon$.

Runtime. We can construct the code (find P) with rejection sampling. The density of primes is at least $\Omega(\frac{1}{\log n})$ by the Prime Number Theorem (Theorem 3.7), so rejection sampling finds enough primes in time $\tilde{O}(n)$ with probability at least $1 - 2^{-\Omega(n)}$. Encoding takes time $O(\log n)$ to choose the prime P and $\tilde{O}(n)$ to encode into $h(\cdot,\cdot)$. Decoding takes time $O(n^{t+1})$ to brute force search over m and hash each one with h_{uniq}. □

7.2 Randomized Explicit: Adversarial with $\sim (2t+1)\log n$ redundancy

Theorem 1.8. *For all positive integers t, and n sufficiently large, there exists a randomized construction that, in time $\tilde{O}(n)$, with probability $1 - 2^{-8n}$, produces a t-adversarial deletion*

code with redundancy $\sim (2t+1)\log n$*, encoding time* $\tilde{O}(n)$ *and decoding time* $\tilde{O}(n^{t+1})$.

Proof. By the equivalence between adversarial deletion codes and deterministic document exchange (see e.g., [CJLW22, Hae19]), it suffices to give a deterministic document exchange protocol with the provided redundancy. We describe the encoding and decoding algorithms, then justify the correctness and runtime.

Call strings m and m' *confusable* if they share a length $n-t$ subsequence. We have the following straightforward lemma

Lemma 7.1. *Fix* $m \in \{0,1\}^n$*. The number of* $m' \in \{0,1\}^n$ *that are confusable with* m *is at most* n^{2t}*.*

Proof. The number of subsequences of m of length $n-t$ is at most $\binom{n}{t} \le n^t$ (choosing the t deleted symbols). Each one of these subsequences is contained in at most n^t strings in $\{0,1\}^n$ (Lemma 3.1). □

Let $h_{\text{uniq}} : \{0,1\}^n \to \{0,1\}^r$ be a hash for deterministic document exchange for t deletions, where $r = \alpha t \log^2 n$ for some absolute constant $\alpha > 0$ (Theorem 3.9). Call a message/prime pair (m,p) *good* — where m is a length n string and p is a prime — if, for all strings $m' \neq m$ confusable with m, we have $h_{\text{uniq}}(m) \mod p \neq h_{\text{uniq}}(m') \mod p$.

Construction. Let P_{all} be the set of primes in $[M/2, M]$ for $M = 100 M_0 \ln M_0$ for $M_0 = 2\alpha n^{2t} \log n$, where α is the absolute constant from Theorem 3.9. Randomly choose a multiset P of $10n$ primes chosen independently at random from the primes P_{all}. This set P specifies the code.

Encoding. Our deterministic document exchange hash is $h_2(m) = (h_{\text{uniq}}(m) \mod p, p)$ for some p chosen from P so that (m,p) is a good message/prime pair. We compute p by brute force search. Since we know $p \in P$, we can store a description of p in $\log |P| = O(\log n)$ bits, so we can store $h_2(m)$ in $\log |P| + \log M \le (2t+1)\log n + O(\log\log n)$ bits.

Decoding. Suppose z is a length $n-t$ received string, and suppose (g,p) is the hash. By brute force search, find all supersequences m' of z such that $h_{\text{uniq}}(m') \mod p = g$. If there is exactly one such m', return m', else return $\perp$.

Correctness. By the Prime Number Theorem (Corollary 3.8), P_{all} has at least $M/10 \ln M > M_0$ primes.

We consider the probability (over the choice of P) that some message $m \in \{0,1\}^n$ is decoded incorrectly. Since $h_{\text{uniq}}(\cdot)$ is a deterministic document exchange hash, for any of the at-most n^{2t} many messages m' that are confusable with m, we have $h_{\text{uniq}}(m) \neq h_{\text{uniq}}(m')$. Since $|h_{\text{uniq}}(m) - h_{\text{uniq}}(m')| \le 2^{\alpha t \log^2 n}$, there are at most $\log_{M/2} 2^{\alpha t \log^2 n} \le \alpha \log n$ primes in P_{all} dividing $h_{\text{uniq}}(m) - h_{\text{uniq}}(m')$. Thus, for at most $n^{2t} \cdot \alpha \log n$ primes $p \in P_{\text{all}}$, the pair (m,p) fails to be good. Thus, at least $1 - \frac{n^{2t} \cdot \alpha \log n}{|P_{\text{all}}|} > 1/2$ of the primes in P_{all} form a good pair with m. Thus, the probability that P fails to contain a good prime for m is at most $2^{-|P|} \le 2^{-10n}$. Thus, with probability at least $1 - 2^{-10n}$ over the choice of P, a fixed message

m is decoded correctly. By a union bound over the 2^n possible messages m, with probability at least $1 - 2^{-9n}$, every message m is decoded correctly.

Runtime. We can construct the code (find P) with rejection sampling. The density of primes is at least $\Omega(\frac{1}{\log n})$ by the Prime Number Theorem, so rejection sampling finds enough primes in time $\tilde{O}(n)$ with probability at least $1 - 2^{-\Omega(n)}$. Encoding is dominated by the time it takes to find a good prime for the message m. Indeed, we compute the n^{2t} hashes $h_{\text{uniq}}(m')$ of all messages m' confusable with m, each of which takes time $\tilde{O}(n)$, and then take all hashes modulo p for all $10n$ primes in P. Overall, the encoding is done in time $\tilde{O}(n^{2t+1})$. Decoding takes time $O(n^{t+1})$ to brute force search all the supersequences m' of $\tau(m)$ and hash each one with h_{uniq}. □

References

[AGEA94] Khaled AS Abdel-Ghaffar and Amr El Abbadi. An optimal strategy for comparing file copies. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):87–93, 1994.

[Bel15] Djamal Belazzougui. Efficient deterministic single round document exchange for edit distance. *arXiv preprint arXiv:1511.09229*, 2015.

[BGM88] Daniel Barbara and Hector Garcia-Molina. Exploiting symmetries for low-cost comparison of file copies. In *[1988] Proceedings. The 8th International Conference on Distributed*, pages 471–479. IEEE, 1988.

[BGZ17] Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. *IEEE Transactions on Information Theory*, 64(5):3403–3410, 2017.

[BL91] Daniel Barbara and Richard J. Lipton. A class of randomized strategies for low-cost comparison of file copies. *IEEE Transactions on Parallel & Distributed Systems*, 2(02):160–170, 1991.

[BZ16] Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 51–60. IEEE, 2016.

[CGK16] Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 712–725, 2016.

[CJL15] Zitan Chen, Sidharth Jaggi, and Michael Langberg. A characterization of the capacity of online (causal) binary channels. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 287–296, 2015.

[CJLW22] Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols and almost optimal binary codes for edit errors. *Journal of the ACM*, 69(6):1–39, 2022.

[CN88] Imre Csiszár and Prakash Narayan. The capacity of the arbitrarily varying channel revisited: Positivity, constraints. *IEEE transactions on Information Theory*, 34(2):181–193, 1988.

[CPSV00] Graham Cormode, Mike Paterson, Süleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 197–206, 2000.

[CR20] Mahdi Cheraghchi and João Ribeiro. An overview of capacity results for synchronization channels. *IEEE Transactions on Information Theory*, 67(6):3207–3232, 2020.

[DLVP96] Charles De La Vallee-Poussin. Recherches analytiques sur la théorie des nombres premiers. *Ann. Soc. Sc. Bruxelles*, 1896.

[Dus18] Pierre Dusart. Explicit estimates of some functions over primes. *The Ramanujan Journal*, 45:227–251, 2018.

[GH21] Venkatesan Guruswami and Johan Håstad. Explicit two-deletion codes with redundancy matching the existential bound. *IEEE Transactions on Information Theory*, 67(10):6384–6394, 2021.

[GL20] Venkatesan Guruswami and Ray Li. Coding against deletions in oblivious and online models. *IEEE Transactions on Information Theory*, 66(4):2352–2374, 2020.

[GS16] Venkatesan Guruswami and Adam Smith. Optimal rate code constructions for computationally simple channels. *Journal of the ACM (JACM)*, 63(4):1–37, 2016.

[GS18] Ryan Gabrys and Frederic Sala. Codes correcting two deletions. *IEEE Transactions on Information Theory*, 65(2):965–974, 2018.

[Had96] Jacques Hadamard. Sur la distribution des zéros de la fonction zeta(s) et ses conséquences arithmétiques. *Bulletin de la Societé mathematique de France*, 24:199–220, 1896.

[Hae19] Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 334–347. IEEE, 2019.

[Ham50] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[HER18] Serge Kas Hanna and Salim El Rouayheb. Guess & check codes for deletions, insertions, and synchronization. *IEEE Transactions on Information Theory*, 65(1):3–15, 2018.

[HER19] Serge Kas Hanna and Salim El Rouayheb. List decoding of deletions using guess & check codes. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2374–2378. IEEE, 2019.

[Hoe94] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[HS21] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings and codes for insertions and deletions—a survey. *IEEE Transactions on Information Theory*, 67(6):3190–3206, 2021.

[HSS18] Bernhard Haeupler, Amirbehshad Shahrasbi, and Madhu Sudan. Synchronization strings: List decoding for insertions and deletions. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 76–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[IMS05] Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1665–1676. IEEE, 2005.

[Jow12] Hossein Jowhari. Efficient communication protocols for deciding edit distance. In *European Symposium on Algorithms*, pages 648–658. Springer, 2012.

[KM13] Yashodhan Kanoria and Andrea Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Transactions on Information Theory*, 59(10):6192–6219, 2013.

[KMS10] Adam Kalai, Michael Mitzenmacher, and Madhu Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *2010 IEEE International Symposium on Information Theory*, pages 997–1001. IEEE, 2010.

[Lan08] Michael Langberg. Oblivious communication channels and their capacity. *IEEE Transactions on Information Theory*, 54(1):424–429, 2008.

[Lev01] Vladimir I Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory, Series A*, 93(2):310–332, 2001.

[Lip94] Richard J Lipton. A new approach to information theory. In *STACS 94: 11th Annual Symposium on Theoretical Aspects of Computer Science Caen, France, February 24–26, 1994 Proceedings 11*, pages 699–708. Springer, 1994.

[LN98] Amos Lapidoth and Prakash Narayan. Reliable communication under channel uncertainty. *IEEE transactions on Information Theory*, 44(6):2148–2177, 1998.

[MBT10] Hugues Mercier, Vijay K Bhargava, and Vahid Tarokh. A survey of error-correcting codes for channels with symbol synchronization errors. *IEEE Communications Surveys & Tutorials*, 12(1):87–96, 2010.

[Mit08] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. In Joachim Gudmundsson, editor, *Algorithm Theory - SWAT 2008, 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2-4, 2008, Proceedings*, volume 5124 of *Lecture Notes in Computer Science*, pages 1–3. Springer, 2008.

[MPSW05] Silvio Micali, Chris Peikert, Madhu Sudan, and David A Wilson. Optimal error correction against computationally bounded noise. In *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2*, pages 1–16. Springer, 2005.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[Orl91] Alon Orlitsky. Interactive communication: Balanced distributions, correlated files, and average-case complexity. In *FOCS*, pages 228–238, 1991.

[PSW24] Francisco Pernice, Oscar Sprumont, and Mary Wootters. List-Decoding Capacity Implies Capacity on the q-ary Symmetric Channel, October 2024. arXiv:2410.20020.

[SB20] Jin Sima and Jehoshua Bruck. On optimal k-deletion correcting codes. *IEEE Transactions on Information Theory*, 67(6):3360–3375, 2020.

[SGB20] Jin Sima, Ryan Gabrys, and Jehoshua Bruck. Optimal systematic t-deletion correcting codes. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 769–774. IEEE, 2020.

[Sha48] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[SKSY24] Omer Sabary, Han Mao Kiah, Paul H Siegel, and Eitan Yaakobi. Survey for a decade of coding for dna storage. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, 2024.

[SNT04] Torsten Suel, Patrick Noel, and Dimitre Trendafilov. Improved file synchronization techniques for maintaining large replicated collections over slow networks. In *Proceedings. 20th International Conference on Data Engineering*, pages 153–164. IEEE, 2004.

[SRB19] Jin Sima, Netanel Raviv, and Jehoshua Bruck. Two deletion correcting codes from indicator vectors. *IEEE transactions on information theory*, 66(4):2375–2391, 2019.

[SS21a] Ronen Shaltiel and Jad Silbak. Explicit list-decodable codes with optimal rate for computationally bounded channels. *computational complexity*, 30:1–70, 2021.

[SS21b] Ronen Shaltiel and Jad Silbak. Explicit uniquely decodable codes for space bounded channels that achieve list-decoding capacity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1516–1526, 2021.

[SS22] Ronen Shaltiel and Jad Silbak. Error correcting codes that achieve bsc capacity against channels that are poly-size circuits. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 13–23. IEEE, 2022.

[SS24] Ronen Shaltiel and Jad Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 2028–2038, 2024.

[VT65] RR Varshamov and GM Tenengolts. Codes which correct single asymmetric errors (in russian). *Automatika i Telemkhanika*, 161(3):288–292, 1965.

8 Appendix

8.1 Proof of Lemma 3.4

To prove Lemma 3.4, we use (in a black-box fashion) the following deterministic document exchange protocol, which achieves the optimal hash size and polynomial decoding time.

Theorem 8.1 ([CJLW22])**.** *There exists a deterministic document exchange protocol for t deletions with hash size $O(t \log n)$ and encoding and decoding time $\operatorname{poly} n$, where the exponent in the polynomial is independent of t. There also exists a systematic adversarial t-deletion code with redundancy $O(t \log n)$ and encoding and decoding time $\operatorname{poly} n$, where the exponent in the polynomial is independent of t.*

We now prove Lemma 3.4 which is restated for convenience.

Lemma 3.4. *[Randomized document exchange is equivalent to systematic oblivious deletions] The following hold:*

1. *Suppose we have a systematic oblivious t-deletion code with n message bits, r redundancy bits, encoding time T_E, and decoding time T_D. Then we can construct in $O(n)$ time a randomized document exchange protocol with length n, hash length r, distance t, error ε, encoding time $T_E + O(n)$, and decoding time $T_D + O(n)$.*

2. *Suppose we have a randomized document exchange protocol with length n, hash length r, distance t, error ε, encoding time T_E, and decoding time T_D. Then we can construct in $O(n)$ time a systematic oblivious t-deletion code with n message bits, $r + O(t \log(r))$ redundancy bits, encoding time $T_E + O(n + \operatorname{poly} r)$, and decoding time $T_D + O(n + \operatorname{poly} r)$.*

Proof. For the first part, suppose we have a systematic oblivious t-deletion code with error ε with encoding $\mathrm{Enc}(x) = x \circ h(x)$ for some hash $h : \{0,1\}^n \to \{0,1\}^r$ and decoder $\mathrm{Dec} : \{0,1\}^{n+r-t} \to \{0,1\}^n$. Let $\mathrm{Dec}' : \{0,1\}^{n-t} \times \{0,1\}^r$ be the decoder $\mathrm{Dec}'(x,h) = \mathrm{Dec}(x \circ h)$. By definition, (h, Dec') is a randomized document exchange protocol with length n, distance t, hash length t, and error ε.

Now suppose we have a randomized document exchange protocol with length n, hash $h : \{0,1\}^n \to \{0,1\}^r$ of length r, distance t, error ε, and decoder $\mathrm{Dec}_h : \{0,1\}^{n-t} \times \{0,1\}^r \to \{0,1\}^n$. Let $\mathrm{Enc}_0 : \{0,1\}^r \to \{0,1\}^{r+g(r,t)}$ and $\mathrm{Dec}_0 : \{0,1\}^{r+g(r,t)-t} \to \{0,1\}^r$ be the encoding and decoding functions, respectively, for the deletion code given by Theorem 8.1, where $g(r,t) \le O(t \log r)$ and the encoding and decoding functions running in $\operatorname{poly} r$ time (the exponent in the polynomial is independent of t). We define our oblivious deletion code's encoding and decoding functions as follows.

Encoding. Let the encoding for our t oblivious deletion code $\mathrm{Enc} : \{0,1\}^n \to \{0,1\}^{n+r+g(r,t)}$ be defined as $\mathrm{Enc}(m) = m \circ \mathrm{Enc}_0(h(m))$.

Decoding. Suppose that z is a subsequence of $m \circ \mathrm{Enc}_0(h)$ of length $n + r + g(r,t) - t$ (if it is a longer subsequence, apply t deletions arbitrarily).

- Let z_0 be the first $n - t$ bits of z, and let z_1 be the last $r + g(r,t) - t$ bits of z.
- Compute $\mathrm{Dec}_h(z_0, \mathrm{Dec}_0(z_1))$.

Correctness. Fix a deletion pattern τ and a message m. Again, let z_0 be the first $n-t$ bits of z, and let z_1 be the last $r + g(r,t) - t$ bits of z. It's easy to see that z_0 is a subsequence of m and z_1 is a subsequence of $\mathrm{Enc}_0(h(m))$. Over the randomness of the encoder, m is fixed, so z_0 is fixed for a fixed τ. Since $(\mathrm{Enc}_0, \mathrm{Dec}_0)$ corrects t adversarial deletions, we know that the $\mathrm{Dec}_0(z_1) = h(m)$. Further, since (h, Dec_h) gives a randomized document exchange protocol, $\mathrm{Dec}_h(z_0, h(m))$ returns m with probability at least $1 - \varepsilon$, as desired.

Runtime. Encoder Enc_0 and decoder Dec_0 runs in $\operatorname{poly} r$ time (because we chose the code from [CJLW22]), hash h runs in T_E time, and Dec_h runs in T_D time, so the total encoding time is $T_E + O(n) + \operatorname{poly} r$ and decoding time is $T_D + O(n) + \operatorname{poly} r$. □

8.2 Proof of Theorem 4.4

As discussed above, Theorem 4.4 is merely a reformulation of [KMS10, Theorem 2.2]. Thus, for completeness, we reproduce their original proof here, altering only the parameters slightly.

Here, we will define a deletion pattern $\tau \in [n]$ that represents t deletions to be the set of indices that are *not* deleted. The set of all t-deletion patterns is denoted by $P_{t,n} = \{\{a_1, a_2, \ldots, a_{n-t}\} \mid a_1 < a_2 < \cdots < a_{n-t}\}$. For a string $x \in \{0,1\}^n$, $\tau(x)$ denotes x after applying the deletion pattern to it. We abuse notation and denote by τ the set of indices that are not affected by the deletion pattern and also the function that applies the deletions.

The first definition presents a *distance* function between two deletion patterns.

Definition 8.2. Let $\tau = \{a_1, \ldots, a_{n-t}\}$ and $\tau' = \{b_1, \ldots, b_{n-t}\}$ be two t-deletion patterns. Denote by $\Delta(\tau, \tau')$, the number of disagreements between τ and τ':

$$\Delta(\tau, \tau') = |\{i \mid a_i \neq b_i\}|$$

Next, we give the definition of an (ℓ, t)-*bad string*. This is a string in $\{0,1\}^n$ for which there are two "ℓ-far apart" deletion patterns τ, τ' such that $\tau(x) = \tau'(x)$. Formally,

Definition 8.3. Let $\ell \geq 1$. A string $x \in \{0,1\}^n$ is (ℓ, t)-bad if there are two distinct t-deletion patterns such that $\tau(x) = \tau'(x)$ and $\Delta(\tau, \tau') \geq \ell$.

In [KMS10], the authors showed that the number of (ℓ, t) bad strings is not big. Specifically,

Lemma 8.4. *[KMS10, Lemma 2.3] Let $\ell \geq 1$. There are at most $\binom{n}{t}^2 2^{n-\ell}$ different (ℓ, t)-bad strings of length n.*

An important step in their proof was to bound for a given t-deletion pattern τ, how many other t-deletion patterns are ℓ-close to it. Formally,

Lemma 8.5. *[KMS10, Lemma 2.2] Let $\ell > 0$ be an integer. For any t-deletion pattern τ, the number of t-deletion pattern τ' such that $\Delta(\tau, \tau') \leq \ell$ is at most*

$$(\ell + 1)\binom{2t + \ell + 1}{2t + 1}\binom{t + \ell}{t} .$$

Another useful lemma that [KMS10] used was the following

Lemma 8.6. *Let ρ be a joint distribution over $S \times T$ for finite sets S and T such that the marginal distribution over S is uniform. Let $g : T \to S$ be a function. Then, $\mathbf{Pr}_{(a,b)\sim\rho}[g(b) = a] \leq |T|/|S|$.*

We are now ready to present the proof of Theorem 4.4. The theorem is restated for convenience.

Theorem 4.4. *[KMS10, Theorem 2.2, rephrased] Let n be a large enough integer. Let C be a code with block length n that is a t-random deletion code in the average case with error probability ε. Then, the redundancy of C is at least*

$$\log \binom{n}{t} + t - \log(3t) - \log(2/(1 - \varepsilon)) - O(t \log\log(n)) .$$

Proof. Set $\ell = 3t \log n$. The number of (ℓ, t)-bad strings is at most

$$\binom{n}{t}^2 \cdot 2^{n - 3t\log n} \leq \left(\frac{n^t}{t!}\right)^2 \cdot \left(\frac{1}{n}\right)^{3t} \cdot 2^n = \frac{1}{(t!)^2} \cdot \frac{2^n}{n^t} ,$$

where the inequality follows by using the upper bound $\binom{n}{t} \leq n^t/t!$. Let C be the code guaranteed by the theorem, and let $\mathcal{A}$ be the decoding algorithm of the code.

We will next define an algorithm $\mathcal{G}$ that gets as input $\tau(c)$ for some random codeword c and random t-deletion pattern τ and outputs (c, τ) with nonnegligible probability. On input s, $\mathcal{G}$ runs the decoder $\mathcal{A}$ on s and then returns $\mathcal{G}(s) = (\mathcal{A}(s), \tau')$ where τ' is the lexicographically first deletion pattern for which $s = \tau'(\mathcal{A}(c))$. Now, the probability that the decoder $\mathcal{A}$ succeeds and that the codeword is *not* an (ℓ, t)-bad string is at least

$$(1-\delta) - \frac{2^n}{(t!)^2 \cdot n^t} \cdot \frac{1}{|C|} \geq \frac{1-\delta}{2} \ .$$

Indeed, at the worst-case scenario, all the (ℓ, t)-bad strings are codewords. The inequality above holds since otherwise we would have that $|C| \leq \frac{2^n}{(t!)^2 \cdot n^t} \cdot \frac{2}{1-\delta}$ which implies that the redundancy of C' is at least $t \log n + 2\log(t!) - \log(2/(1-\delta)) > \log \binom{n}{t} + t - \log(3t) - \log(2/(1-\delta))$ for all t and so the claim holds.

Fix τ. Conditioned on c not being (ℓ, t)-bad, for all τ' such that $\tau'(c) = \tau(c)$, we have that $\Delta(\tau, \tau') \leq \ell - 1$. By Lemma 8.5, the number of such close patterns is at most

$$\ell \cdot \binom{2t+\ell}{2t+1} \cdot \binom{t+\ell-1}{t} \leq \ell \cdot \left(e \cdot \frac{2t+\ell}{2t+1}\right)^{2t+1} \left(e \cdot \frac{t+\ell}{t}\right)^{t} \leq \ell \cdot \left(\frac{6\ell}{t}\right)^{3t+1} \ ,$$

where the first inequality follows by the upper bound $\binom{n}{t} \leq (en/t)^t$ which holds for $t < n/2$ and by noting that $\ell > 2t$. Now, since each deletion pattern is equally likely, the probability that $\mathcal{G}(\tau(c)) = (c, \tau)$ is at least $(1-\delta)/(2\alpha)$ where $\alpha = \ell \cdot \left(\frac{6\ell}{t}\right)^{3t+1}$.

Now, using Lemma 8.6 with the sets $S = C \times P_{t,n}$ and $T = \{0,1\}^{n-t}$, we get the probability that $g(\tau(c)) = (c, \tau)$ is at most $\frac{2^{n-t}}{|C| \cdot \binom{n}{t}}$. Therefore, we have

$$\frac{2^{n-t}}{|C| \cdot \binom{n}{t}} \geq \frac{1-\delta}{2\alpha}$$

and by taking logarithm on both sides and rearranging terms, we find that

$$\log(|C|) \leq n - t - \log \binom{n}{t} - \log((1-\delta)/2) + \log \alpha$$

and the claim follows by noting that $\log(\alpha) = \log(3t \log n) + (3t+1)\log(18 \log n) = O(t \log \log n)$. □