

Plotkin Attacks!

On the (Mis)use of Raptor Codes in Asynchronous Verifiable Information Dispersal

Victor Shoup
Category Labs
victor@shoup.net

October 4, 2026

Abstract

Some asynchronous verifiable information dispersal (AVID) and data-availability protocols certify that a disperser’s message is available once $2f + 1$ of $n = 3f + 1$ parties, at most f of them Byzantine, attest that they hold their erasure-coded fragments, and defer reconstruction and validation of the encoding until later. We call this design pattern *reconstruct-later AVID*. If the f Byzantine attestors later withhold their fragments, reconstruction must succeed from the $f + 1$ honest fragments that remain, and a Byzantine disperser chooses which $f + 1$ after seeing the encoding symbols assigned to the honest parties. We call a subset of fragments that does not determine the message *bad*. Raptor codes are a tempting choice here: with modest coding overhead they have very small decoding-failure probability under random erasures, and a preliminary Walrus design and an earlier Walrus testnet used RaptorQ in this way. We show that the pattern is insecure with the standardized Raptor codes R10 (RFC 5053) and RaptorQ (RFC 6330). For a message of K input symbols, every assignment of R10 encoding symbols to the honest parties contains a bad $(f + 1)$ -subset whenever $K > \lfloor \log_2(2f + 2) \rfloor$, and every RaptorQ assignment whenever $K > 7H_{\text{HDPC}} + \lfloor \log_2(2f + 2) \rfloor$, where H_{HDPC} is RFC 6330’s number of high-density parity-check symbols. Under slightly stronger conditions, randomized polynomial-time algorithms find such subsets, and an implementation of the RaptorQ attack defeats an RFC 6330 decoder on every adversarial subset tested. The key ingredient is the Plotkin bound, which forces a nonzero input whose encoded image has low Hamming weight. For RaptorQ at $n = 3f + 1$, an empirical attack over $\mathbb{F}_{256}$ succeeds at every tested f from 9 to 45 with $K = f + 1$, far below the $f = 77$ at which the bound above first applies, and empirical attacks over $\mathbb{F}_4$, $\mathbb{F}_{16}$, and $\mathbb{F}_{256}$ succeed at values of K well below that bound. For R10, a smaller K does not help: for every standardized $34 \leq K \leq 8192$ and every n with $3f + 1 \leq n \leq 4f \leq 65520$, explicit certificates give a bad subset for every assignment to the honest parties when the encoding-symbol identifiers (ESIs) are $0, 1, \dots, n - 1$, and with probability at least $1/2$ when the ESIs are uniformly random. Benchmarks of our implementations show that the MDS alternative costs little: in our parameter regime, Reed–Solomon is often faster than, and never more than a small constant factor slower than, either R10 or RaptorQ, for both encoding and decoding.

1 Introduction

In asynchronous verifiable information dispersal (AVID), a disperser seeks to make a message available to a set of parties for later retrieval. Typically, the disperser erasure-encodes the message

and sends an authenticated coded fragment to each party. Availability is established once enough parties attest that they hold their fragments. Cachin and Tessaro’s original AVID constructions check during dispersal that all fragments are consistent with a single valid encoding [13]. Some later protocols, such as Dumbo-MVBA’s dispersal mechanism and DispersedLedger’s AVID-M, instead defer reconstruction and validation until after availability is established [14, 15]. We call this design pattern **reconstruct-later AVID**. We call the contrasting design pattern, in which honest parties reconstruct the message before issuing the availability attestation, **reconstruct-first AVID**. The reconstruct-later design pattern is attractive because it keeps full-message reconstruction off the critical path for establishing availability. This can reduce both latency and communication complexity.

The reconstruct-later design pattern also appears in other erasure-coded dissemination protocols. In DispersedSimplex, a consensus protocol, enough coded fragments from honest parties may already be in flight for later reconstruction [16]. Likewise, in the compact broadcast and reliable message-distribution mechanisms of Shoup and Smart, once an honest party delivers, enough honest coded echoes are already in flight for every honest party eventually to reconstruct [17].

Raptor codes are tempting choices for erasure-coded dissemination because they provide very efficient erasure encoding and decoding and, with modest coding overhead, very small decoding-failure probabilities under random erasures and other probabilistic models. We consider the standardized R10 [4] and RaptorQ [5] codes, and study whether they are vulnerable when used in reconstruct-later dissemination. A preliminary design for Walrus [22], a decentralized storage network, proposed using RaptorQ in reconstruct-later dissemination, and an earlier Walrus testnet implementation used RaptorQ before the system switched to Reed–Solomon [25]. Monad’s RaptorCast dissemination protocol [26] uses R10 for block-proposal dissemination but follows the reconstruct-first pattern.

Consider the optimal-resilience setting with n parties, where

$$n = 3f + 1,$$

and exactly f parties are Byzantine in the worst case. There are then $2f + 1$ honest parties. In the reconstruct-later AVID design pattern considered here, data availability is established once $2f + 1$ parties attest that they hold their assigned fragments. As many as f of these parties may be Byzantine and later withhold their fragments, so eventual reconstruction may have to rely on the fragments held by only $f + 1$ honest attestors (this point is made explicitly in [15], for example). The security question is therefore adversarial: after the assignment of encoding symbols to the honest parties is known, can an efficient adversary select $f + 1$ honest fragments that do not determine the message?

Consider an erasure code whose encoding symbols are identified by indices (encoding-symbol identifiers, or ESIs, for Raptor codes; evaluation points for Reed–Solomon). To make the order of choices explicit, we use the following attack game against a specified challenger $\mathcal{C}$. Fix the coding parameters and $n = 3f + 1$.

- (1) The challenger $\mathcal{C}$ chooses a set E of n distinct valid indices and sends E to the adversary.
- (2) The adversary chooses a message m and sends m to $\mathcal{C}$; the challenger encodes m at the indices in E .
- (3) The challenger $\mathcal{C}$ chooses $H \subseteq E$ with $|H| = 2f + 1$ and sends H to the adversary.
- (4) The adversary chooses $S \subseteq H$ with $|S| = f + 1$.

A challenger $\mathcal{C}$ is **legal** if it respects this move order and the stated index and cardinality constraints. The challenger may be randomized. We place no computational restriction on it. The adversary wins if the encoding symbols indexed by S do not uniquely determine m . Equivalently, the adversary wins if there is another message $m' \neq m$ whose encoding agrees with that of m at every index in S . We call such a set S **bad**. We say that the code has **Byzantine erasure security against $\mathcal{C}$** if it is computationally infeasible for an adversary to win this game. The challenger specifies choices made by the protocol or environment outside the adversary’s control; it is not an adversarial party.

We now explain how winning the Byzantine erasure game yields an attack on reconstruct-later AVID. The AVID protocol assigns the n encoding indices in E to parties, possibly in a randomized way. The set of f Byzantine parties may be fixed before the protocol begins. The set H in the game is then the set of $2f + 1$ indices assigned to the honest parties. Thus the challenger’s choice of H models the assignment produced by the AVID protocol. The order $m \rightarrow H$ also permits protocols in which this assignment is determined only after the sender commits to the message, for example from randomness derived from a hash of the commitment. Suppose the adversary finds a bad set $S \subseteq H$. The f Byzantine parties together with the $f + 1$ honest parties whose indices lie in S can supply the $2f + 1$ attestations needed to establish data availability. The Byzantine parties may then withhold their fragments, leaving only the fragments indexed by S guaranteed to be available for reconstruction. Since S is bad, these fragments do not determine m . The protocol has thus certified as available a message that cannot be reconstructed. Neither the challenger’s choice of H nor the adversary’s choice of S requires adaptive corruptions. The Byzantine parties may remain fixed throughout the attack.

We study Byzantine erasure security against legal challengers for the standardized Raptor codes R10 [4] and RaptorQ [5]. Let K denote the number of input symbols in the message. For the linear codes studied here, whether S is bad is independent of the particular message: if the transmitted difference induced by a nonzero message difference x vanishes at every index in S , then m and $m + x$ agree on all encoding symbols indexed by S for every m .

For a fixed challenger $\mathcal{C}$, a stronger information-theoretic condition is that every $(f + 1)$ -subset of every H that $\mathcal{C}$ may output determines the message. That condition is sufficient for Byzantine erasure security against $\mathcal{C}$ but is not necessary: the mere existence of a bad subset does not by itself give an efficient attack. Probabilistic decoding guarantees for Raptor codes do not by themselves establish Byzantine erasure security against a given challenger, because they do not show that, after the realized H is known, it is hard to find a bad subset. Appendix D gives additional context on the relevant probability spaces and order of choices.

With a maximum-distance-separable (MDS) code, every $(f + 1)$ -subset reconstructs automatically. An MDS code of length $3f + 1$ and dimension $f + 1$ reconstructs from every $f + 1$ encoding symbols [3]. Thus such a code has Byzantine erasure security against every legal challenger. Once the protocol guarantees that $f + 1$ honest fragments are stored or in flight, eventual reconstruction follows even if every Byzantine party withholds its fragment.

Contributions. Write $K = f + 1 - \Delta$, where Δ is the coding overhead.

- (i) *Plotkin attacks.* For R10, every honest assignment H contains a bad $(f + 1)$ -subset whenever $K > \lfloor \log_2(2f + 2) \rfloor$, a condition that fails only when $K \leq \log_2(2f + 2)$. For RaptorQ, whose encoder first maps the K input symbols to intermediate symbols over $\mathbb{F}_{256}$, the argument applies to the intermediate codewords whose coordinates lie in $\mathbb{F}_2$ and gives the same conclusion whenever $K > 7H_{\text{HDPC}} + \lfloor \log_2(2f + 2) \rfloor$, where H_{HDPC} is RFC 6330’s number of high-density parity-check (HDPC) symbols; at $\Delta = 0$ this first applies at $f = 77$.

Under slightly stronger conditions, a randomized adversary finds a bad subset in expected at most $4f + 4$ trials, for every legal choice of E and every subsequent legal choice of H , independently of any probability distribution used by $\mathcal{C}$ (Sections 2 and 3). The key combinatorial ingredient is the Plotkin bound, which forces a nonzero input whose encoded image has low Hamming weight.

- (ii) *Experimental verification.* An implementation of the RaptorQ attack against an RFC 6330 decoder fails to decode on every adversarial subset tested, while, for the same parameters, 200 uniformly random $(f + 1)$ -subsets all decode (Appendix C).
- (iii) *Smaller K for RaptorQ.* The restriction to binary coordinates discards up to $7H_{\text{HDPC}}$ dimensions. Working instead with the intermediate codewords whose coordinates lie in $\mathbb{F}_4$, $\mathbb{F}_{16}$, or $\mathbb{F}_{256}$ recovers much of the lost attack range at $n = 3f + 1$: over $\mathbb{F}_{256}$ a search finds attacks at every tested f from 9 to 45 with $\Delta = 0$, far below the $f = 77$ at which the bound of (i) first applies, and at positive overhead the three fields find attacks over broad ranges of f and Δ , sporadically as deep as $\Delta = 65$ (Appendix A).
- (iv) *Beyond $n = 3f + 1$, and R10 at every overhead.* Reduction to the $n = 3f + 1$ case gives generic bounds for every $n > 3f + 1$ (Section 4). For standard R10, explicit precomputed certificates show that, for every standardized $34 \leq K \leq 8192$ and every $3f + 1 \leq n \leq 4f$ with $4f \leq 65520$, a bad subset exists for every honest assignment when E is the ESI prefix $0, 1, \dots, n - 1$, and with probability at least $1/2$ when E is a uniformly random ESI set; in particular, at $n = 3f + 1$ no amount of overhead protects standard R10 in this range (Appendix B).
- (v) *Cost of the fix.* Single-threaded benchmarks of our implementations at $K = f + 1$ and $n = 3f + 1$ show that Reed–Solomon is often faster than, and never more than a small constant factor slower than, either R10 or RaptorQ, for both encoding and decoding (Appendix E).

A Byzantine disperser may also commit to encoding symbols that do not form a valid codeword. Reconstruct-later AVID can address this by decoding a candidate message, re-encoding it, and checking the resulting commitment or Merkle root [14, 15, 16, 22]. This is separate from decodability. The attack in this paper uses a valid codeword and valid authentication proofs; the selected honest fragments do not suffice to reconstruct the message.

R10 and RaptorQ are fountain codes. They can generate additional encoding symbols from the same K input symbols as needed, so a receiver in an ordinary object delivery application may collect symbols until decoding succeeds [6]. In the Byzantine erasure game, however, the adversary chooses S after seeing the realized honest assignment. The ability to generate additional symbols does not prevent such a subset from being bad.

The basic phenomenon that RaptorQ can fail under targeted erasures is not new. Lopes and Neves demonstrated it for standardized RaptorQ by an offline search at decoding overheads 0, 1, and 2, and subsequent work proposed cryptographic defenses against targeted erasures [9, 10]. In our setting, the $n = 3f + 1$ threshold gives a stronger conclusion. The Plotkin argument gives a bad subset for every realized honest assignment under the stated dimension bounds, and the randomized attack finds one efficiently under slightly stronger conditions.

The attacks studied here do not apply to reconstruct-first AVID. An honest party that issues an availability attestation already holds the complete message, so among any $2f + 1$ attesters at least $f + 1$ honest parties hold the message. Availability therefore does not depend on later reconstruction from a subset of coded fragments selected after the assignment is known. Monad’s RaptorCast [26], for example, follows the reconstruct-first pattern, so our attacks do not apply to it.

The rest of the paper. Section 2 develops the Plotkin attacks on R10, including the binary Plotkin bound, an efficient randomized algorithm for finding a bad subset, and the resulting Byzantine erasure attack. Section 3 extends the argument to RaptorQ by restricting to the intermediate codewords with binary coordinates. Section 4 extends the analysis beyond the optimal-resilience regime $n = 3f + 1$. Section 5 discusses related attacks, defenses, and applications. Section 6 concludes. Appendix A asks how much of the binary RaptorQ attack range can be recovered by working over larger subfields. Appendix B exploits standard R10 structure to extend the larger- n attack far beyond the generic Plotkin bound. Appendix C reports the end-to-end RaptorQ decoder experiments and the exhaustive standardized binary-subfield dimension check. Appendix D provides additional context on the probability spaces underlying fountain-code decoding guarantees and their relation to Byzantine erasure security. Appendix E measures the practical cost of replacing R10 or RaptorQ with MDS Reed–Solomon at the parameters studied here. Appendix F gives the algebraic and algorithmic details underlying the experimental Reed–Solomon implementations.

2 Plotkin attacks on R10

Once its encoding parameters and ESIs are fixed, R10 maps the K input symbols linearly over $\mathbb{F}_2$ to the transmitted encoding symbols. Thus the $2f + 1$ encoding symbols indexed by an honest set H are represented by a binary matrix with K columns. We first prove a statement for any binary linear code, by which we mean a code whose encoding map is represented by a matrix over $\mathbb{F}_2$. If

$$K > \lfloor \log_2(2f + 2) \rfloor,$$

then every collection of $2f + 1$ binary encoding rows contains a bad $(f + 1)$ -subset. The proof reduces the problem to finding a nonzero input vector whose encoded image has Hamming weight at most f . The Plotkin bound shows that such a vector must exist. Equivalently, if every $(f + 1)$ -subset reconstructs, then

$$K \leq \lfloor \log_2(2f + 2) \rfloor.$$

If $K = f + 1 - \Delta$, where Δ is the coding overhead, this requires

$$\Delta \geq f + 1 - \lfloor \log_2(2f + 2) \rfloor,$$

so with constant overhead some $(f + 1)$ -subset fails to reconstruct for all sufficiently large f . Existence alone does not break Byzantine erasure security against a challenger, because the adversary must find a bad subset efficiently. Under the slightly stronger condition

$$K \geq \lceil \log_2(4f + 4) \rceil,$$

a randomized adversary finds one in expected at most $4f + 4$ trials for every realized H . The success bound holds conditionally on every realized H and is over the adversary’s own randomness. Subsection 2.1 proves the binary Plotkin bound. Subsection 2.2 gives the randomized attack. Subsection 2.3 states how the resulting bad subset wins the Byzantine erasure game.

2.1 Binary linear codes and the Plotkin bound

We now prove the general binary statement used for R10.

Let

$$A \in \mathbb{F}_2^{h \times K},$$

and suppose reconstruction is required from every set of s rows, where $1 \leq s \leq h$. Write

$$d := h - s + 1;$$

this is the minimum distance that every- s -subset reconstruction will force. Write $[h] := \{1, \dots, h\}$. For $S \subseteq [h]$, let A_S denote the submatrix formed from the rows indexed by S . For a binary vector y , write $\text{wt}(y)$ for its Hamming weight.

Definition 1. A set $S \subseteq [h]$ of size s is **good** if $\text{rank}(A_S) = K$ and **bad** otherwise.

A good set determines all K input symbols. A bad set does not. We call a nonzero x satisfying $\text{wt}(Ax) < d$ a **witness**. The next lemma shows exactly how such a vector certifies a bad subset.

Lemma 2 (Low-weight witnesses and bad row sets). *There exists a bad set $S \subseteq [h]$ of size s if and only if there exists a nonzero $x \in \mathbb{F}_2^{K \times 1}$ such that*

$$\text{wt}(Ax) < d.$$

Moreover, given any such x , any s zero coordinates of Ax index a bad set. Equivalently, every s -subset is good if and only if every nonzero $x \in \mathbb{F}_2^{K \times 1}$ satisfies

$$\text{wt}(Ax) \geq d.$$

Proof. Suppose first that $x \neq 0$ and $\text{wt}(Ax) < d$. Then

$$\text{wt}(Ax) \leq d - 1 = h - s,$$

so Ax has at least s zero coordinates. Let S index any s of them. Then $A_S x = 0$, so A_S does not have full column rank and S is bad.

Conversely, if S is bad, then there exists $x \neq 0$ with $A_S x = 0$. Thus Ax is zero on all s coordinates in S , and hence

$$\text{wt}(Ax) \leq h - s = d - 1 < d.$$

The final equivalence is the negation of this characterization. $\square$

The same condition has the standard coding-theoretic interpretation needed later.

Corollary 3 (Minimum-distance formulation). *Every s -subset is good if and only if the binary linear code*

$$\mathcal{C}_A := \{Ax : x \in \mathbb{F}_2^{K \times 1}\} \subseteq \mathbb{F}_2^{h \times 1}$$

has dimension K and minimum distance at least d .

Proof. If every s -subset is good, Lemma 2 gives $\text{wt}(Ax) \geq d$ for every $x \neq 0$. In particular, $Ax \neq 0$ for every $x \neq 0$, so $x \mapsto Ax$ is injective, $\mathcal{C}_A$ has dimension K , and its minimum distance is at least d .

Conversely, suppose $\mathcal{C}_A$ has dimension K and minimum distance at least d . Since $\mathcal{C}_A$ is the image of the K -dimensional space $\mathbb{F}_2^{K \times 1}$, the map $x \mapsto Ax$ is injective. Thus $x \neq 0$ implies $Ax \neq 0$, and the minimum-distance assumption gives $\text{wt}(Ax) \geq d$. Lemma 2 then shows that every s -subset is good. $\square$

The following elementary averaging fact drives both the Plotkin bound and the randomized attack below.

Lemma 4 (Binary weight averaging). *Let X be uniform over $\mathbb{F}_2^{K \times 1}$ and put*

$$W := \text{wt}(AX).$$

Then

$$\mathbb{E}[W] \leq \frac{h}{2},$$

and

$$\Pr[W < d] \geq 1 - \frac{h}{2d}.$$

Proof. For each nonzero row a_i of A , the bit $a_i X$ is uniform in $\mathbb{F}_2$. For a zero row, it is always zero. Therefore each coordinate contributes at most $1/2$ to the expected weight, giving

$$\mathbb{E}[W] \leq \frac{h}{2}.$$

Since $d \geq 1$, Markov's inequality gives

$$\Pr[W \geq d] \leq \frac{\mathbb{E}[W]}{d} \leq \frac{h}{2d}.$$

Therefore

$$\Pr[W < d] \geq 1 - \frac{h}{2d}.$$

□

When $d > h/2$, which we call the **Plotkin regime**, Corollary 3 places the desired minimum distance in the range addressed by the Plotkin bound. Equivalently, since $d = h - s + 1$ and h, s are integers, $d > h/2$ is the condition $h \geq 2s - 1$. The usual Plotkin bound can therefore be obtained directly from the same averaging statement [1].

Theorem 5 (Plotkin bound for every- s -subset reconstruction). *Let $A \in \mathbb{F}_2^{h \times K}$ and suppose $d > h/2$. If every s rows of A have rank K , then*

$$2^K \leq \frac{2d}{2d - h}.$$

Proof. Let X be uniform over $\mathbb{F}_2^{K \times 1}$ and put $W := \text{wt}(AX)$. By Lemma 2, if every s -subset is good, then $W < d$ can occur only when $X = 0$. Conversely, $X = 0$ gives $W = 0 < d$ because $d \geq 1$. Hence

$$\Pr[W < d] = 2^{-K}.$$

Lemma 4 gives

$$2^{-K} \geq 1 - \frac{h}{2d} = \frac{2d - h}{2d}.$$

Since $2d - h > 0$, this is equivalent to the claimed bound. □

At the $n = 3f + 1$ parameters,

$$h = 2f + 1, \quad s = f + 1, \quad d = f + 1,$$

so $2d - h = 1$ and the general bound takes an especially simple form.

Theorem 6 (Plotkin bound at the $n = 3f + 1$ parameters). *Let $A \in \mathbb{F}_2^{(2f+1) \times K}$. If every $(f + 1)$ rows of A have rank K , then*

$$2^K \leq 2f + 2,$$

and therefore

$$K \leq \lfloor \log_2(2f + 2) \rfloor.$$

Equivalently, if

$$K > \lfloor \log_2(2f + 2) \rfloor,$$

then every collection of $2f + 1$ binary encoding rows contains a bad $(f + 1)$ -subset.

Proof. Substitute $h = 2f + 1$ and $d = f + 1$ into Theorem 5. □

Remark 7 (Tightness). The dimension bound is tight for infinitely many values of f . Suppose $2f + 1 = 2^m - 1$ and let the rows of $A \in \mathbb{F}_2^{(2^m-1) \times m}$ be all nonzero row vectors of $\mathbb{F}_2^{1 \times m}$. For every nonzero $x \in \mathbb{F}_2^{m \times 1}$, exactly 2^{m-1} rows have inner product 1 with x . Hence

$$\text{wt}(Ax) = 2^{m-1} = f + 1 = d.$$

Lemma 2 therefore shows that every $(f + 1)$ -subset is good, while

$$K = m = \log_2(2f + 2).$$

This is the binary simplex code and meets the Plotkin bound with equality.

Corollary 8 (Required overhead). *Suppose $K = f + 1 - \Delta$, where Δ is the coding overhead. If every $(f + 1)$ -subset of the $2f + 1$ honest encoding symbols reconstructs, then*

$$\Delta \geq f + 1 - \lfloor \log_2(2f + 2) \rfloor.$$

In particular, a constant Δ , or more generally any $\Delta = o(f)$, cannot ensure reconstruction from every $(f + 1)$ -subset as f grows.

Proof. Substitute $K = f + 1 - \Delta$ into Theorem 6. □

For R10, let A_H denote the $(2f + 1) \times K$ matrix of encoding rows corresponding to the encoding symbols in H . As noted in the introduction, Theorem 6 applies to every possible H , however H is determined. Thus, when $K > \lfloor \log_2(2f + 2) \rfloor$, every such H contains a bad $(f + 1)$ -subset. The next subsection shows that, under a slightly stronger condition, such a subset can be found efficiently.

It may seem that the same binary Plotkin argument should apply to Reed–Solomon over $\mathbb{F}_{256}$, since such a code is also $\mathbb{F}_2$ -linear after each field element is expanded into eight bits. It does not: this expansion gives a binary code of length eight times the number of transmitted symbols, and binary Hamming weight counts individual bits rather than party-held symbols. The relevant metric here is symbol Hamming weight, in which Reed–Solomon is MDS. After restriction to $2f + 1$ honest positions with dimension $f + 1$, its minimum symbol distance is $f + 1$, so no nonzero codeword is supported on at most f parties.

Remark 9 (Dependence on h and d). Theorem 5 makes the roles of the number h of available rows and the required minimum distance d explicit. The averaging argument gives a nontrivial bound only in the Plotkin regime $d > h/2$. The $n = 3f + 1$ parameters lie at the smallest positive margin, $2d - h = 1$. If $d \leq h/2$, Lemma 4 gives no positive lower bound of this form, and the direct Plotkin argument does not yield a nontrivial dimension restriction. Section 4 considers increasing n and reduces to the $n = 3f + 1$ case by forcing the $n - (3f + 1)$ surplus encoding symbols to zero.

2.2 Efficient randomized attack

The preceding result guarantees that a bad subset exists but does not show how to find one. We now give a randomized algorithm that finds one efficiently for every realized honest set H , under a slightly stronger condition on K .

Given any matrix $A \in \mathbb{F}_2^{(2f+1) \times K}$, repeat the following experiment. Choose a uniformly random nonzero vector

$$x \in \mathbb{F}_2^{K \times 1}.$$

Compute

$$y := Ax.$$

If

$$\text{wt}(y) \leq f,$$

then Lemma 2 applies: y has at least $f + 1$ zero coordinates, and any $f + 1$ of those coordinates index a bad set S . Thus a successful sample produces both a low-weight witness and the adversarial row set itself.

The next proposition gives a generic success bound for this attack.

Proposition 10 (Sampling a bad witness). *Let $A \in \mathbb{F}_2^{(2f+1) \times K}$ be arbitrary. Let X be uniform over $\mathbb{F}_2^{K \times 1}$ and put*

$$W := \text{wt}(AX).$$

Then

$$\Pr[W \leq f] \geq \frac{1}{2f+2}.$$

Consequently, if

$$K \geq \lceil \log_2(4f+4) \rceil,$$

then for a uniformly random nonzero X ,

$$\Pr[\text{wt}(AX) \leq f] \geq \frac{1}{4f+4}.$$

Proof. The first bound is Lemma 4 with $h = 2f + 1$ and $d = f + 1$. The parameter K enters when the trivial sample $X = 0$ is excluded. This event has probability 2^{-K} and always satisfies $W \leq f$, but it is not a useful attack witness. Thus

$$\Pr[W \leq f \text{ and } X \neq 0] \geq \frac{1}{2f+2} - 2^{-K}.$$

If $K \geq \lceil \log_2(4f+4) \rceil$, then $2^{-K} \leq 1/(4f+4)$, so

$$\Pr[W \leq f \text{ and } X \neq 0] \geq \frac{1}{4f+4}.$$

Conditioning on $X \neq 0$ can only increase this lower bound because $\Pr[X \neq 0] \leq 1$. □

Theorem 11 (Efficient subset attack). *Suppose*

$$K \geq \lceil \log_2(4f+4) \rceil.$$

For every matrix $A \in \mathbb{F}_2^{(2f+1) \times K}$, a randomized adversary can find a bad $(f+1)$ -subset in expected at most $4f+4$ trials. For any integer $T \geq 0$, after T independent trials, its probability of not yet finding a bad subset is at most

$$\exp\left(-\frac{T}{4f+4}\right).$$

Proof. Apply Proposition 10 to the arbitrary matrix A . Each independent nonzero sample succeeds with probability at least $1/(4f + 4)$. The expected number of trials is therefore at most $4f + 4$. The probability that all T trials fail is at most

$$\left(1 - \frac{1}{4f + 4}\right)^T \leq \exp\left(-\frac{T}{4f + 4}\right).$$

□

A naive trial computes one binary matrix-vector product with $(2f + 1)K$ matrix entries and then counts the nonzero output coordinates. Thus a trial takes $O(fK)$ bit operations in a straightforward implementation. The expected total work is $O(f^2K)$ bit operations, which is $O(f^3)$ when $K = \Theta(f)$. Bit packing and word-level XOR operations can make the practical cost substantially smaller.

The assumption $K \geq \lceil \log_2(4f + 4) \rceil$ is mild in the intended regime. For example, $K \geq f/2$ implies it for all $f \geq 11$. When K is a constant fraction of f , the attack is efficient for every possible H . The only probability in Theorem 11 is the adversary’s own internal randomness used to search for a witness. The bound holds for every realized H and therefore against every legal challenger.

2.3 Byzantine erasure attack

Under the condition of Theorem 11, the adversary runs the randomized search after H is revealed and obtains a bad subset S . It wins the Byzantine erasure game by outputting S . Since the attack works for every realized H , it breaks Byzantine erasure security against every legal challenger.

3 Plotkin attacks on RaptorQ

We next transfer the binary Plotkin argument to standardized RaptorQ. The reduction is less immediate than for R10 because the RaptorQ precode includes nonbinary $\mathbb{F}_{256}$ relations, so intermediate codewords need not be binary. We first recall the part of the RaptorQ construction needed for the argument.

RaptorQ structure. RFC 6330 [5] calls the collection of K input symbols to the encoding algorithm a **source block**; its elements are **source symbols**. These are RFC 6330’s names for the message input at this layer. It appends $K' - K$ all-zero padding symbols, where K' is the smallest supported extended source-block size at least K . The padding symbols are internal to the encoder and decoder and are not transmitted. Let S_{LDPC} and H_{HDPC} denote RFC 6330’s numbers of low-density parity-check (LDPC) and high-density parity-check (HDPC) symbols, respectively. RaptorQ then constructs

$$L = K' + S_{\text{LDPC}} + H_{\text{HDPC}}$$

intermediate symbols. The extra intermediate coordinates provide internal precode redundancy, but they are constrained by $S_{\text{LDPC}} + H_{\text{HDPC}}$ precode equations and are not themselves transmitted. RaptorQ uses two kinds of precode relations: sparse binary low-density parity-check (LDPC) relations and high-density parity-check (HDPC) relations, whose coefficients lie in $\mathbb{F}_{256}$ and need not lie in $\mathbb{F}_2$. Once the intermediate symbols have been constructed, both source and repair symbols are obtained from them by XORing selected intermediate symbols; this outer encoding layer is the Luby transform (LT) stage.

RaptorQ arithmetic acts independently at each octet position of a symbol. Fix one octet position for the rest of the argument. Thus a pair of source blocks that differ at this octet position and agree at every other octet position is a valid pair of full source blocks. We regard encoding maps as row vectors and source and intermediate vectors as column vectors. Define the source-symbol and intermediate-symbol spaces

$$\mathcal{S} := \mathbb{F}_{256}^{K \times 1}, \quad \mathcal{V} := \mathbb{F}_{256}^{L \times 1}.$$

Let $u \in \mathcal{S}$ be the vector of source-symbol octets at the chosen position. Our next goal is to define a map

$$C : \mathcal{S} \longrightarrow \mathcal{V}$$

that sends u to its corresponding vector $C(u)$ of intermediate-symbol octets. To this end, RFC 6330, Section 5.3.3.4.2, defines an invertible $L \times L$ matrix. We write it as

$$A = \begin{pmatrix} P \\ G_{\text{SRC}} \\ G_{\text{PAD}} \end{pmatrix} \in \mathbb{F}_{256}^{L \times L}.$$

The first block

$$P \in \mathbb{F}_{256}^{(S_{\text{LDPC}} + H_{\text{HDPC}}) \times L}$$

contains the precode equations. Its first S_{LDPC} rows are binary LDPC rows, and its final H_{HDPC} rows are the possibly nonbinary HDPC rows. RFC 6330 calls the final K' rows of A the matrix G_{ENC} . We split those rows as

$$G_{\text{ENC}} = \begin{pmatrix} G_{\text{SRC}} \\ G_{\text{PAD}} \end{pmatrix}, \quad G_{\text{SRC}} \in \mathbb{F}_2^{K \times L}, \quad G_{\text{PAD}} \in \mathbb{F}_2^{(K' - K) \times L}.$$

The rows of G_{SRC} are the binary encoding rows for the K actual source symbols, and the rows of G_{PAD} are the corresponding rows for the $K' - K$ prescribed zero-padding symbols. In particular, G_{SRC} is not an identity block: the source symbols are recovered as fixed XORs of the intermediate symbols. For fixed K and K' , all of these matrices are fixed by the RaptorQ specification and do not depend on the source data.

Using A , define

$$C(u) := A^{-1} \begin{pmatrix} 0_{(S_{\text{LDPC}} + H_{\text{HDPC}}) \times 1} \\ u \\ 0_{(K' - K) \times 1} \end{pmatrix}.$$

Define the **intermediate code**

$$\mathcal{W} := \text{im } C \subseteq \mathcal{V}.$$

Every source or repair encoding symbol is obtained from the intermediate symbols by a fixed binary row

$$g_X \in \mathbb{F}_2^{1 \times L},$$

because the encoding-symbol generator only adds selected intermediate symbols (RFC 6330, Sections 5.3.3.4.2 and 5.3.5.3). At the chosen octet position, the encoding symbol with identifier X therefore has value

$$g_X C(u).$$

Roadmap for the reduction. Because C is linear, $\mathcal{W}$ is both the set of valid intermediate codewords and the set of differences between valid intermediate codewords:

$$\mathcal{W} = \{C(u') - C(u) : u, u' \in \mathcal{S}\}.$$

Take $u, u' \in \mathcal{S}$ and set

$$z := C(u') - C(u) = C(u' - u) \in \mathcal{W}.$$

For any transmitted encoding row g_X , the difference between the two transmitted octets is

$$g_X C(u') - g_X C(u) = g_X z.$$

Thus a nonzero intermediate codeword $z \in \mathcal{W}$ for which $g_X z = 0$ for every row g_X in a set S of rows exhibits two distinct valid source blocks that agree on every encoding symbol indexed by S . The Plotkin argument is binary, whereas a general element of $\mathcal{W}$ has coordinates in $\mathbb{F}_{256}$. We therefore define the binary subfield subcode

$$\mathcal{W}_2 := \mathcal{W} \cap \mathbb{F}_2^{L \times 1}, \quad \kappa_2 := \dim_{\mathbb{F}_2} \mathcal{W}_2,$$

using the natural embedding $\mathbb{F}_2 = \{0, 1\} \subseteq \mathbb{F}_{256}$. Since every transmitted row g_X is binary,

$$g_X z \in \mathbb{F}_2$$

for every $z \in \mathcal{W}_2$. Hence the transmitted codewords produced by intermediate codewords in $\mathcal{W}_2$ form a binary linear code, to which the Plotkin argument applies. The next lemma shows that $\kappa_2 \geq K - 7H_{\text{HDPC}}$ and that a basis of $\mathcal{W}_2$ can be computed efficiently. We will then represent this binary map in coordinates and apply Theorem 6.

Lemma 12 (Size of the binary subfield subcode). *The binary subfield subcode $\mathcal{W}_2$ satisfies*

$$\kappa_2 \geq K - 7H_{\text{HDPC}}.$$

Moreover, a basis of $\mathcal{W}_2$ can be computed efficiently by binary Gaussian elimination.

Proof. Write P_{LDPC} and P_{HDPC} for the first S_{LDPC} and final H_{HDPC} rows of P . Fix an $\mathbb{F}_2$ -linear isomorphism

$$\iota : \mathbb{F}_{256} \longrightarrow \mathbb{F}_2^{8 \times 1},$$

and extend it coordinatewise to an $\mathbb{F}_2$ -linear isomorphism

$$\iota_H : \mathbb{F}_{256}^{H_{\text{HDPC}} \times 1} \longrightarrow \mathbb{F}_2^{8H_{\text{HDPC}} \times 1}.$$

Define the $\mathbb{F}_2$ -linear map

$$\Phi : \mathbb{F}_2^{L \times 1} \longrightarrow \mathbb{F}_2^{(S_{\text{LDPC}} + 8H_{\text{HDPC}} + K' - K) \times 1}$$

by

$$\Phi(z) := \begin{pmatrix} P_{\text{LDPC}} z \\ \iota_H(P_{\text{HDPC}} z) \\ G_{\text{PAD}} z \end{pmatrix}.$$

We claim that

$$\mathcal{W}_2 = \ker \Phi.$$

First take $z \in \mathcal{W}_2$. Since $z \in \mathcal{W}$, there is a $u \in \mathcal{S}$ such that $z = C(u)$. By the definition of C ,

$$Az = AC(u) = \begin{pmatrix} 0 \\ u \\ 0 \end{pmatrix}.$$

Hence

$$Pz = 0, \quad G_{\text{PAD}}z = 0,$$

and therefore $\Phi(z) = 0$.

Conversely, take $z \in \ker \Phi$. Since ι_H is injective,

$$Pz = 0, \quad G_{\text{PAD}}z = 0.$$

Set

$$u := G_{\text{SRC}}z.$$

Then

$$Az = \begin{pmatrix} 0 \\ u \\ 0 \end{pmatrix}.$$

By the definition of C ,

$$z = C(u) \in \mathcal{W}.$$

Since z is binary,

$$z \in \mathcal{W}_2.$$

This proves $\mathcal{W}_2 = \ker \Phi$.

Rank-nullity gives

$$\kappa_2 = L - \text{rank}_{\mathbb{F}_2}(\Phi).$$

Since the rank of Φ is at most the dimension of its codomain,

$$\begin{aligned} \kappa_2 &\geq L - S_{\text{LDPC}} - 8H_{\text{HDPC}} - (K' - K) \\ &= K - 7H_{\text{HDPC}}. \end{aligned}$$

The matrix of Φ is explicit. The LDPC and padding equations are binary, and applying ι to each HDPC equation gives eight binary equations. Thus binary Gaussian elimination computes a basis of

$$\ker \Phi = \mathcal{W}_2.$$

□

The cancellation of S_{LDPC} is worth making explicit. RaptorQ introduces S_{LDPC} extra intermediate coordinates and also imposes S_{LDPC} binary LDPC constraints, so the LDPC layer has zero net cost in this lower bound. Likewise, passing from K source symbols to K' extended source symbols adds $K' - K$ coordinates, while fixing the $K' - K$ padding symbols to zero imposes the same number of binary constraints. Only the nonbinary HDPC layer incurs an extension-field penalty: each HDPC symbol contributes one extra binary coordinate, while its $\mathbb{F}_{256}$ equation can cost as many as eight binary dimensions, for a net loss of at most seven dimensions per HDPC symbol. Thus the reduction loses at most $7H_{\text{HDPC}}$ dimensions before applying Theorem 6.

The induced binary matrix. We now represent in coordinates the map from intermediate codewords in $\mathcal{W}_2$ to their transmitted codewords. Choose a basis $z_1, \dots, z_{\kappa_2}$ of $\mathcal{W}_2$ and place the basis vectors in the columns of

$$Z_0 := \begin{pmatrix} | & & | \\ z_1 & \cdots & z_{\kappa_2} \\ | & & | \end{pmatrix} \in \mathbb{F}_2^{L \times \kappa_2}.$$

For any collection of h RaptorQ encoding rows $g_{X_1}, \dots, g_{X_h} \in \mathbb{F}_2^{1 \times L}$, let

$$G := \begin{pmatrix} g_{X_1} \\ \vdots \\ g_{X_h} \end{pmatrix} \in \mathbb{F}_2^{h \times L}$$

and define

$$B := GZ_0 \in \mathbb{F}_2^{h \times \kappa_2}.$$

For $x \in \mathbb{F}_2^{\kappa_2 \times 1}$, the vector

$$z = Z_0 x \in \mathcal{W}_2$$

is an intermediate codeword, and the corresponding transmitted codeword on the h selected rows is exactly

$$Gz = GZ_0 x = Bx.$$

If $x \neq 0$, then $z \neq 0$. Since $z \in \mathcal{W}$, there is a $u \in \mathcal{S}$ such that $z = C(u)$. Since $C(0) = 0$, we have $u \neq 0$. Consequently, for a set S of encoding rows,

$$B_S x = 0, \quad x \neq 0,$$

exhibits two distinct valid source blocks that agree on every encoding symbol indexed by S . Therefore

$$S \text{ determines the source block} \implies \text{rank}(B_S) = \kappa_2.$$

We need only this implication. Full rank of B_S rules out ambiguities coming from $\mathcal{W}_2$, but we do not claim that it is sufficient for RaptorQ decoding in general.

At this point the RaptorQ-specific algebra is complete. The remaining problem is exactly the earlier binary problem of requiring every $(f+1)$ -subset to determine the source block, with dimension κ_2 .

Theorem 13 (RaptorQ dimension bound). *Let $g_{X_1}, \dots, g_{X_{2f+1}}$ be any collection of RaptorQ encoding rows. If every $(f+1)$ -subset of the corresponding encoding symbols determines the K source symbols, then*

$$\kappa_2 \leq \lfloor \log_2(2f+2) \rfloor$$

and therefore

$$K \leq 7H_{\text{HDPC}} + \lfloor \log_2(2f+2) \rfloor.$$

Equivalently, if the latter bound is violated, every collection of $2f+1$ RaptorQ encoding rows contains an $(f+1)$ -subset that does not determine the source symbols.

Proof. Under the stated assumption, every $(f+1)$ rows of B have rank κ_2 . Theorem 6, with κ_2 in place of K , gives the first inequality. Lemma 12 gives the second. $\square$

Corollary 14 (RaptorQ overhead). *Suppose*

$$K = f + 1 - \Delta.$$

If every $(f + 1)$ -subset of the $2f + 1$ honest RaptorQ symbols determines the source block, then

$$\Delta \geq f + 1 - 7H_{\text{HDPC}} - \lfloor \log_2(2f + 2) \rfloor.$$

For every $K \leq 1032$, RFC 6330 has $H_{\text{HDPC}} = 10$, so Theorem 13 gives an attack whenever

$$K > 70 + \lfloor \log_2(2f + 2) \rfloor.$$

With zero overhead, where $K = f + 1$, this existence theorem first applies at $f = 77$, $K = 78$. The efficient randomized attack established immediately below requires the slightly stronger condition

$$K - 70 \geq \lceil \log_2(4f + 4) \rceil,$$

and at zero overhead first applies at $f = 78$, $K = 79$. For comparison, the R10 existence theorem applies already at $f = 3$, $K = 4$, the smallest source dimension allowed by the standardized code. At $f = 64$, writing $K = 65 - \Delta$, the R10 theorem applies for $\Delta \leq 57$, while neither binary RaptorQ condition applies for any $\Delta \geq 0$. At $f = 100$, writing $K = 101 - \Delta$, the R10 theorem applies for $\Delta \leq 93$, the RaptorQ existence theorem applies for $\Delta \leq 23$, and the efficient randomized RaptorQ attack applies for $\Delta \leq 22$. The binary reduction used for RaptorQ gives a narrower attack range at moderate values of f , and the randomized-search condition is occasionally one dimension stricter still. Over the full RFC 6330 parameter range, $H_{\text{HDPC}} \leq 16$, so the difference between the R10 and RaptorQ dimension bounds is at most 112 dimensions.

Theorem 15 (Efficient RaptorQ subset attack). *Suppose*

$$K - 7H_{\text{HDPC}} \geq \lceil \log_2(4f + 4) \rceil.$$

For every collection of $2f + 1$ RaptorQ encoding rows, a randomized adversary can find a bad $(f + 1)$ -subset in expected at most $4f + 4$ trials. After T independent trials, the probability of not yet finding a bad subset is at most

$$\exp\left(-\frac{T}{4f + 4}\right).$$

Proof. By Lemma 12, the adversary computes a basis of $\mathcal{W}_2$ by binary Gaussian elimination. It then forms Z_0 and $B = GZ_0$ for the realized honest encoding rows. The same lemma gives

$$\kappa_2 \geq K - 7H_{\text{HDPC}} \geq \lceil \log_2(4f + 4) \rceil.$$

Theorem 11 applied to B gives a nonzero x with $\text{wt}(Bx) \leq f$. The corresponding $z = Z_0x$ is a nonzero intermediate codeword whose transmitted codeword vanishes on at least $f + 1$ rows assigned to honest parties. Choose any $f + 1$ of those zero rows; they form the required bad subset. $\square$

The same Byzantine erasure attack applies to RaptorQ. After H is revealed, the adversary computes the bad subset and outputs it. Two distinct valid source blocks agree on every encoding symbol in that subset, so the subset does not determine the source block. Since the attack works for every realized H , it breaks Byzantine erasure security against every legal challenger under the theorem's condition.

Appendix C reports the end-to-end decoder experiments and the exhaustive standardized computation of κ_2 . Appendix A gives stronger results from the larger subfield subcodes of the intermediate code.

4 Beyond the optimal-resilience $n = 3f + 1$ regime

This section extends the dimension bounds to $n > 3f + 1$. Let

$$n = 3f + 1 + t, \quad t \geq 0.$$

The reduction forces t of the honest encoding symbols to zero, costing at most t dimensions, and then applies the $n = 3f + 1$ bound to the remaining symbols. Suppose data availability is established once $n - f$ parties attest that they hold their assigned fragments. As many as f of those attesters may be Byzantine, so only

$$s = n - 2f = f + 1 + t$$

honest attesters are guaranteed. In the worst case there are

$$h = n - f = 2f + 1 + t$$

honest parties in total.

The Byzantine erasure game against a challenger $\mathcal{C}$ is the game from Section 1 with these cardinalities. The challenger first chooses a set E of n distinct valid indices, and the adversary chooses a message m . The challenger then chooses $H \subseteq E$ with $|H| = h = n - f$, and after seeing H the adversary chooses $S \subseteq H$ with $|S| = s = n - 2f$. The adversary wins if the encoding symbols indexed by S do not uniquely determine m .

Fix such a set H and let A_H be the corresponding $h \times K$ binary matrix. For this fixed H , the stronger information-theoretic condition that every s rows of A_H have rank K is sufficient to prevent a Byzantine erasure attack. By Corollary 3, this condition implies that the code generated by A_H has minimum distance at least

$$h - s + 1 = f + 1.$$

For $t > 0$ this is no longer in the direct Plotkin regime. Restricting the message space so that any fixed t encoding symbols vanish imposes at most t independent linear constraints and leaves $2f + 1$ encoding symbols, reducing to the $n = 3f + 1$ setting.

Proposition 16 (Dimension bound above $3f + 1$). *If every s rows of A_H have rank K , then*

$$K \leq t + \lfloor \log_2(2f + 2) \rfloor.$$

Hence, if this bound is violated, every possible H contains a bad s -subset.

Proof. Choose any t rows of A_H , indexed by T , and define

$$\mathcal{U} := \{x \in \mathbb{F}_2^{K \times 1} : (A_H)_T x = 0\}.$$

Rank-nullity gives

$$\dim \mathcal{U} = K - \text{rank}((A_H)_T) \geq K - t.$$

The t corresponding encoding symbols vanish on $\mathcal{U}$. The hypothesis that every $s = f + 1 + t$ rows of A_H have rank K implies that every $f + 1$ of the remaining $2f + 1$ rows restrict to an injective map on $\mathcal{U}$: otherwise a nonzero $x \in \mathcal{U}$ annihilated by those $f + 1$ rows would also be annihilated by the rows in T , giving a bad s -subset. After choosing a basis of $\mathcal{U}$, Theorem 6 applied to these restricted rows gives

$$\dim \mathcal{U} \leq \lfloor \log_2(2f + 2) \rfloor,$$

and the result follows. $\square$

Corollary 17 (RaptorQ above $3f + 1$). *For a RaptorQ source block, if every s -subset of a set H of encoding symbols assigned to the honest parties determines the source block, then*

$$K \leq t + 7H_{\text{HDPC}} + \lfloor \log_2(2f + 2) \rfloor.$$

Proof. Use the binary subfield subcode $\mathcal{W}_2$ from Section 3 and the corresponding $h \times \kappa_2$ binary matrix B . Every s rows of B have rank κ_2 , so Proposition 16 gives

$$\kappa_2 \leq t + \lfloor \log_2(2f + 2) \rfloor.$$

Lemma 12 gives $\kappa_2 \geq K - 7H_{\text{HDPC}}$. □

Increasing n also permits an MDS code to use source dimension as large as

$$K = s = n - 2f,$$

so values $K > f + 1$ can improve the coding rate. For an attack, however, smaller K is the stronger regime to cover, because it corresponds to more coding redundancy. At $n = 4f$, where $t = f - 1$, Proposition 16 gives no attack for any $K \leq f + 1$ once $f \geq 3$. Appendix B gives much stronger results for standard R10 throughout

$$3f + 1 \leq n \leq 4f,$$

for $34 \leq K \leq 8192$ and $4f \leq 65520$. For canonical ESI prefixes, a bad subset exists for every legal H . Under uniformly random ESI selection, with probability at least $1/2$ over E , a fixed precomputed witness yields a bad subset for every legal subsequent H .

The generic Byzantine erasure attack is also efficient under a slightly stronger condition. For R10, fix t encoding symbols and restrict the message space so that those symbols vanish, as in the proof above. If

$$K - t \geq \lceil \log_2(4f + 4) \rceil,$$

Theorem 11 finds, in expected at most $4f + 4$ sampling trials, a nonzero restricted codeword that vanishes on at least $f + 1$ of the remaining $2f + 1$ encoding symbols. Together with the t encoding symbols already forced to zero, these give a bad $s = f + 1 + t$ subset. For RaptorQ the same argument applies to $\mathcal{W}_2$ whenever

$$K - 7H_{\text{HDPC}} - t \geq \lceil \log_2(4f + 4) \rceil.$$

The finite bounds have a simple communication interpretation. Suppose

$$n \sim cf$$

for a fixed constant $c > 3$. Since

$$t = n - 3f - 1 = \frac{c-3}{c}n + o(n),$$

Proposition 16 gives the necessary condition

$$K \leq \frac{c-3}{c}n + o(n)$$

for R10. The same asymptotic condition follows for RaptorQ whenever $H_{\text{HDPC}} = o(n)$.

In the one-symbol-per-party model used here, the coded-data communication blowup is essentially n/K . Leaving the parameter region in which the generic attack is guaranteed can itself require large communication blowup. For example, when $n \sim 4f$, the necessary condition is roughly $K \leq n/4$, corresponding to a communication blowup of roughly 4 or more. This inequality only marks the limit of the generic Plotkin guarantee. Satisfying it does not imply safety: particular R10 or RaptorQ instances may still admit Byzantine erasure attacks.

5 Related work and practical context

This section compares our Byzantine erasure attacks with prior targeted-erasure attacks, fixed-rate Raptor distance analyses, cryptographic defenses, every-authorized-set constructions, and the Walrus deployment history. The distinguishing feature in our BFT setting is that the adversary sees the realized honest encoding rows before choosing the subset from which reconstruction must succeed.

5.1 Targeted erasures against RaptorQ

Earlier work on rateless-code authentication noted a related “distribution attack” in peer-to-peer content delivery: a malicious source can send valid encoded symbols according to a maliciously chosen distribution rather than the distribution prescribed by the rateless code [8]. Lopes and Neves subsequently gave a concrete and implemented targeted-erasure attack against standardized RaptorQ [9]. Their starting observation is closely related to ours: once the deterministic relationship between encoding-symbol identifiers and encoding equations is known, a network adversary can choose which packets to suppress instead of accepting a random erasure pattern. They search for received packet sets whose RaptorQ decoding matrix is rank deficient. Thus the broad claim that excellent random-erasure performance need not survive adversarial packet selection was established before this work. Lopes and Neves also argue that the same rationale should extend to R10, whose encoding-symbol identifiers likewise determine a public pseudorandom LT structure.

The optimization problem and quantitative regime are different. Lopes and Neves seek erasure patterns that disable a fixed RFC 6330 transmission while dropping as few packets as possible. They therefore solve an additional optimization problem that our attack does not. Their search enumerates combinations of removed and replacement packets. They report that this offline search ran for more than six months, with experiments for input dimensions from 10 through 1002 and decoding overheads 0, 1, and 2. Because RFC 6330 fixes the equation associated with each encoding-symbol identifier, a resulting pattern can be reused for the same parameters. The small tested overheads describe the reach of that computational study, not a proof that targeted failures cease to exist at larger overhead.

The Byzantine erasure problem studied here does not ask for a minimum-erasure pattern. After the honest set H is revealed in the attack game, the adversary needs any bad $(f + 1)$ -subset of those $2f + 1$ encoding symbols. Theorem 13 proves that such a subset exists for every realized RaptorQ assignment under its stated bound. Theorem 15 finds one in expected $O(f)$ trials under the slightly stronger bound. Our objective is different: we do not minimize the number of erased symbols. The resulting guarantees apply to every realized honest assignment under the stated bounds and extend to much larger decoding overheads than the 0, 1, and 2 values studied by Lopes and Neves. Appendix A gives empirical attacks at still larger overheads beyond the binary reduction.

5.2 Fixed-rate R10 and Raptor distance

Paolini et al. [18] study fixed-rate truncations of the standardized R10 encoder and use a generic minimum-distance estimation method to search for low-weight codewords. For a $(512, 256)$ fixed-rate code, they report an estimated minimum distance of 25, along with low-weight examples at other sizes. Lázaro et al. [19, 20] subsequently derive weight-distribution and distance-spectrum results for fixed-rate Raptor ensembles with linear-random precodes. These works study coding-theoretic distance properties; they do not consider adversarial subset selection or Byzantine erasure

security. Our R10-specific results address that different problem: Appendix B gives explicit precomputed attack witnesses throughout the stated standardized parameter ranges and proves that the resulting bad reconstruction subset exists for every honest assignment in those ranges.

5.3 Cryptographic defenses against targeted erasures

Falcon Codes were proposed as a cryptographic defense against targeted erasures and corruptions at the LT-coding layer [10]. Their core construction uses a strong pseudorandom generator to make the LT encoding graph unpredictable, encrypts encoded symbols, and authenticates them. The authors also observe that hiding or randomizing the graph alone is insufficient: encoded symbol values can themselves leak information about the hidden graph and enable data-dependent targeted erasures. These observations strengthen the case for hiding both the graph and the encoded data, but do not strengthen the targeted-erasure bounds for standard RaptorQ, whose encoding graph is public. Falcon also does not directly address the reconstruct-later AVID setting considered here. Its security hides coding structure from an adversarial channel. In the reconstruct-later AVID attack considered here, the Byzantine disperser is itself the encoder and therefore knows the encoding graph and encoded symbols. Thus hiding this information from the adversarial channel does not prevent the attacks considered here.

5.4 Every-authorized-set reconstruction

Cao, Yu, Yang, Lou, and Hou explicitly observe that the usual probabilistic LT guarantee is insufficient when every authorized subset of storage servers must permit recovery [11]. Their construction tests every authorized server subset and regenerates the LT encoding if any such subset fails to decode. Each server stores many LT symbols, so an authorized server subset selects groups of encoding rows rather than arbitrary individual rows. This grouping places their construction outside our one-symbol-per-party Plotkin attack model.

A recent systems-level observation appears in eAID [12]. eAID considers N nodes with crash-fault bound $F < N/2$, fixes its recovery threshold at $F + 1$, and uses Reed–Solomon coding. The authors explicitly distinguish this choice from LT and fountain codes: their safety argument requires any $F + 1$ fragments to suffice, and autonomous pruning is simple only when decodability depends on the number of surviving fragments rather than on which particular fragments survive. This independently recognizes, at the design level, the distinction between a probabilistic fountain-code guarantee and a requirement that every surviving threshold set decode. eAID may assign multiple fragments to each node, so its benign crash-failure setting differs from our one-symbol-per-party Byzantine attack model. It does not analyze adversarial subset selection.

5.5 Walrus

As noted in the introduction, a preliminary Walrus design [22] proposed using RaptorQ in the reconstruct-later design pattern. Its September 2024 whitepaper described decoding as probabilistic and chose only a small gap between the input dimension and the desired reconstruction threshold. A February 2025 snapshot of the Walrus repository preserves the RaptorQ implementation alongside the new Reed–Solomon implementation [23]. At the shard level, this is precisely the coding pattern studied here: availability is certified before reconstruction, while later recovery relies on threshold subsets of RaptorQ encoding symbols. Walrus may assign multiple shards to one storage node, however, so our one-symbol-per-party theorem does not by itself establish an attack on every stake-weighted RaptorQ testnet configuration.

Before mainnet, Walrus replaced RaptorQ with Reed–Solomon coding. The mainnet announcement [24] says that the performance was similar for its parameters and describes Reed–Solomon as providing “perfect robustness,” meaning reconstruction always succeeds once the threshold number of slivers is present. The current Walrus paper [25] thanks Joachim Neu for identifying a serious vulnerability in the previous testnet implementation, states that it was related to RaptorQ, and says that it led to the switch to Reed–Solomon. The public sources do not describe that vulnerability technically, so we do not know how it relates to the attack studied here.

6 Conclusion

AVID protocols use erasure coding in two different ways. In reconstruct-later AVID, parties attest to their coded fragments before reconstruction. If Byzantine attestors later withhold their fragments, availability may therefore have to follow from a subset of honest fragments selected after the encoding assignment is known. We formalize resistance to this attack as Byzantine erasure security against a challenger $\mathcal{C}$ that specifies the protocol environment. Under the stated parameter conditions, the generic efficient attacks in this paper show that neither R10 nor RaptorQ has Byzantine erasure security against any legal challenger.

In reconstruct-first AVID, honest parties reconstruct the message before issuing the availability attestation. Any $2f + 1$ attestors therefore include at least $f + 1$ honest parties that already hold the message. Availability no longer depends on later reconstruction from a subset of coded fragments selected after the assignment is known, so the attacks studied here do not apply to this design pattern. Raptor codes may still be used as a dissemination mechanism in such protocols.

Thus the issue is not the use of Raptor codes in AVID per se. It is relying on probabilistic decoding guarantees in a setting where an efficient adversary may choose the reconstruction subset after seeing the realized assignment.

The MDS alternative costs little. Appendix E benchmarks single-threaded Reed–Solomon implementations against R10 and RaptorQ at $K = f + 1$ and $n = 3f + 1$ for $31 \leq f \leq 256$ and messages of 0.5 to 4 MiB. In this regime, Reed–Solomon is often faster than, and never more than a small constant factor slower than, either R10 or RaptorQ, for both encoding and decoding, and its source throughput never falls below 31.9 Gbit/s for encoding and 7.0 Gbit/s for decoding on one core.

Reproducibility artifacts. Separate reproducibility artifacts accompany the main RaptorQ attack experiments, the RaptorQ subfield experiments, the larger- n R10 results, and the performance benchmarks [38, 39, 40, 41].

Disclosure of AI use

Large language models (various versions of Claude and ChatGPT) were used to help brainstorm ideas, to assist in writing text and code, and to proofread. The authors take full responsibility for the content of this paper.

References

- [1] M. Plotkin. Binary Codes with Specified Minimum Distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

- [2] P. Delsarte. An Algebraic Approach to the Association Schemes of Coding Theory. *Philips Research Reports Supplements*, no. 10, 1973.
- [3] I. S. Reed and G. Solomon. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. doi:10.1137/0108018.
- [4] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. Raptor Forward Error Correction Scheme for Object Delivery. RFC 5053, October 2007. <https://www.rfc-editor.org/rfc/rfc5053.html>.
- [5] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder. RaptorQ Forward Error Correction Scheme for Object Delivery. RFC 6330, August 2011. <https://www.rfc-editor.org/rfc/rfc6330.html>.
- [6] A. Shokrollahi. Raptor Codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006. doi:10.1109/TIT.2006.874390.
- [7] A. Shokrollahi and M. Luby. Raptor Codes. *Foundations and Trends in Communications and Information Theory*, 6(3–4):213–322, 2009. doi:10.1561/01000000060.
- [8] M. N. Krohn, M. J. Freedman, and D. Mazières. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In *2004 IEEE Symposium on Security and Privacy*, pages 226–240, 2004. doi:10.1109/SECPRI.2004.1301326.
- [9] J. Lopes and N. Neves. Stopping a Rapid Tornado with a Puff. In *2014 IEEE Symposium on Security and Privacy*, pages 509–523, 2014. doi:10.1109/SP.2014.39.
- [10] A. Juels, J. Kelley, R. Tamassia, and N. Triandopoulos. Falcon Codes: Fast, Authenticated LT Codes (Or: Making Rapid Tornadoes Unstoppable). In *22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1032–1047, 2015. doi:10.1145/2810103.2813728. Cryptology ePrint Archive, Paper 2014/903.
- [11] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. T. Hou. LT Codes-Based Secure and Reliable Cloud Storage Service. In *2012 Proceedings IEEE INFOCOM*, pages 693–701, 2012. doi:10.1109/INFOCOM.2012.6195814.
- [12] R. Kerur, D. Agrawal, D. Malkhi, M. K. Reiter, and A. Wieder. eAID: Elastic Asynchronous Information Dispersal with Post-Dissemination Pruning. arXiv:2603.24761v2, June 2026.
- [13] C. Cachin and S. Tessaro. Asynchronous Verifiable Information Dispersal. In *24th IEEE Symposium on Reliable Distributed Systems*, pages 191–202, 2005. doi:10.1109/RELDIS.2005.9.
- [14] Y. Lu, Z. Lu, Q. Tang, and G. Wang. Dumbo-MVBA: Optimal Multi-valued Validated Asynchronous Byzantine Agreement, Revisited. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2020. doi:10.1145/3382734.3405707.
- [15] L. Yang, S. J. Park, M. Alizadeh, S. Kannan, and D. Tse. DispersedLedger: High-Throughput Byzantine Consensus on Variable Bandwidth Networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 493–512, 2022.
- [16] V. Shoup. Sing a Song of Simplex. In *38th International Symposium on Distributed Computing (DISC)*, LIPIcs 319, pages 37:1–37:22, 2024. doi:10.4230/LIPIcs.DISC.2024.37.

- [17] V. Shoup and N. P. Smart. Lightweight Asynchronous Verifiable Secret Sharing with Optimal Resilience. *Journal of Cryptology*, 37:27, 2024. doi:10.1007/s00145-024-09505-6.
- [18] E. Paolini, M. Varrella, M. Chiani, B. Matuz, and G. Liva. Low-Complexity LDPC Codes with Near-Optimum Performance over the BEC. In *2008 4th Advanced Satellite Mobile Systems Conference (ASMS)*, 2008. doi:10.1109/ASMS.2008.54. arXiv:0804.2991.
- [19] F. Lázaro, E. Paolini, G. Liva, and G. Bauch. On the Weight Distribution of Fixed-Rate Raptor Codes. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2880–2884, 2015. doi:10.1109/ISIT.2015.7282983. arXiv:1502.01567.
- [20] F. Lázaro, E. Paolini, G. Liva, and G. Bauch. Distance Spectrum of Fixed-Rate Raptor Codes with Linear Random Precoders. *IEEE Journal on Selected Areas in Communications*, 34(2):422–436, 2016. doi:10.1109/JSAC.2015.2504281. arXiv:1511.00562.
- [21] F. Lázaro, G. Liva, E. Paolini, and G. Bauch. Bounds on the Error Probability of Raptor Codes. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–5, 2016. doi:10.1109/GLOCOM.2016.7841639. arXiv:1604.07560.
- [22] Mysten Labs Team. Walrus: An Efficient Decentralized Storage Network. Whitepaper, version 1.0, September 16, 2024. <https://sonnino.com/papers/walrus-whitepaper.pdf>.
- [23] Mysten Labs. Walrus source repository. Historical revision 5362ab787064, February 24, 2025. <https://github.com/MystenLabs/walrus/tree/5362ab787064641e7e0a5bcdd98109243e099e57>.
- [24] Walrus Team. Announcing Mainnet. March 27, 2025. https://docs.wal.app/blog/06_mainnet.
- [25] G. Danezis, G. Giuliani, L. Kokoris Kogias, M. Legner, J.-P. Smith, A. Sonnino, and K. Wüst. Walrus: An Efficient Decentralized Storage Network. In *2026 ACM SIGSAC Conference on Computer and Communications Security*, 2026. arXiv:2505.05370v4.
- [26] B. Gilkalaye. RaptorCast: Designing a Messaging Layer. Category Labs, May 1, 2025. <https://www.category.xyz/blogs/raptorcast-designing-a-messaging-layer>.
- [27] S.-J. Lin, T. Y. Al-Naffouri, Y. S. Han, and W.-H. Chung. Novel Polynomial Basis With Fast Fourier Transform and Its Application to Reed–Solomon Erasure Codes. *IEEE Transactions on Information Theory*, 62(11):6284–6299, 2016.
- [28] S.-J. Lin, A. Alloum, and T. Y. Al-Naffouri. Principal Pivot Transforms on Radix-2 DFT-Type Matrices. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2358–2362, 2017.
- [29] N. Tang and Y. Lin. Fast Encoding and Decoding Algorithms for Arbitrary (n, k) Reed–Solomon Codes over $\mathbb{F}_{2^m}$. *IEEE Communications Letters*, 24(4):716–719, 2020.
- [30] F. Didier. Efficient Erasure Decoding of Reed–Solomon Codes. CoRR abs/0901.1886, 2009. arXiv:0901.1886.
- [31] J. N. Cooper and R. B. Ellis. Linearly Bounded Liars, Adaptive Covering Codes, and Deterministic Random Walks. *Journal of Combinatorics*, 1(3–4):307–333, 2010. arXiv:0909.0029.

- [32] A. Siegel. Median Bounds and Their Application. *Journal of Algorithms*, 38(1):184–236, 2001. doi:10.1006/jagm.2000.1138.
- [33] C. Berner. `raptorq`: Rust implementation of RaptorQ (RFC 6330). Version 2.0.1. <https://github.com/cberner/raptorq>.
- [34] Category Labs. `monad-bft`, `monad-raptor` component. Software repository, pinned at commit `fcafa0d7cb96`. <https://github.com/category-labs/monad-bft>.
- [35] Firedancer Contributors. `firedancer`, `src/ballet/reedsol`. Software repository, pinned at commit `813cebdb1a02`. <https://github.com/firedancer-io/firedancer>.
- [36] Intel Corporation. ISA-L: Intel Intelligent Storage Acceleration Library. Version 2.31.1. <https://github.com/intel/isa-l>.
- [37] A. Trier Olesen. `reed-solomon-simd`: Reed–Solomon erasure coding over $\mathbb{F}_{2^{16}}$. Version 3.1.0. <https://github.com/AndersTrier/reed-solomon-simd>.
- [38] Reproducibility artifact for the main RaptorQ attack experiments. 2026. <https://github.com/victorshoup/plotkin-attacks-artifacts/blob/v1/raptorq-attack-artifact.zip>.
- [39] Reproducibility artifact for the RaptorQ subfield experiments. 2026. <https://github.com/victorshoup/plotkin-attacks-artifacts/blob/v1/raptorq-subfield-artifact.zip>.
- [40] Reproducibility artifact for the sparse-intermediate larger- n R10 results. 2026. <https://github.com/victorshoup/plotkin-attacks-artifacts/blob/v1/r10-larger-n-artifact.zip>.
- [41] Reproducibility artifact for the erasure-code performance benchmarks. 2026. <https://github.com/victorshoup/plotkin-attacks-artifacts/blob/v1/erasure-code-benchmark-artifact.zip>.

A Stronger RaptorQ attacks over subfields of $\mathbb{F}_{256}$

The RaptorQ attack in Section 3 pays a substantial quantitative price for reducing the intermediate code to binary. Each HDPC relation over $\mathbb{F}_{256}$ becomes eight binary equations, so the argument retains only $\kappa_2 \geq K - 7H_{\text{HDPC}}$ dimensions of the K -dimensional intermediate code. For $K \leq 1032$, RFC 6330 has $H_{\text{HDPC}} = 10$, so the reduction can lose 70 dimensions. At zero overhead, where $K = f + 1$, the resulting existence theorem does not apply until $f = 77$, and the efficient randomized attack does not apply until $f = 78$. The goal of this appendix is to determine how much attack range is recovered by working over larger subfields, where more of the intermediate code remains available. The empirical gain is large. Over $\mathbb{F}_{256}$, the finite search below finds a witness in all 100 sampled choices of H at every tested zero-overhead point from $f = 9$ through $f = 45$. At positive overhead, the three fields behave differently. The $\mathbb{F}_4$ attack region is broad and regular, $\mathbb{F}_{16}$ is substantially narrower, and $\mathbb{F}_{256}$ exhibits additional irregular regions that persist far beyond its main transition. These empirical search results do not assert a theorem for every possible H , but they show that the binary reduction leaves substantial quantitative attack strength unused.

To investigate this systematically, we study the subfield subcodes obtained by restricting the intermediate code to $\mathbb{F}_q$, for $q \in \{2, 4, 16, 256\}$. Moving to a larger subfield leaves more intermediate codewords available to the attack. At the same time, low-weight codewords become rarer over a larger alphabet. For a uniformly random coordinate over $\mathbb{F}_q$, the probability of being nonzero is $1 - 1/q$, whereas an attack on the $2f + 1$ coordinates indexed by H needs the transmitted image of a nonzero intermediate codeword to have weight at most f . The experiments below show how these effects trade off in standardized RaptorQ. Below, we use *transition* descriptively for a region in which the observed search success changes sharply, and *secondary regions* for separated successes beyond the main high-success region.

A.1 Subfield subcodes of the intermediate code

This subsection identifies the parts of the RaptorQ intermediate code that remain over each subfield and records the exact dimensions available to an attack. We use the RaptorQ notation from Section 3. Thus

$$\mathcal{W} = \text{im } C \subseteq \mathbb{F}_{256}^{L \times 1}$$

is the intermediate code, where

$$L = K' + S_{\text{LDPC}} + H_{\text{HDPC}}.$$

For

$$q \in \{2, 4, 16, 256\},$$

define the subfield subcode

$$\mathcal{W}_q := \mathcal{W} \cap \mathbb{F}_q^{L \times 1}, \quad \kappa_q := \dim_{\mathbb{F}_q} \mathcal{W}_q.$$

Let

$$e_q := [\mathbb{F}_{256} : \mathbb{F}_q],$$

so

$$e_2 = 8, \quad e_4 = 4, \quad e_{16} = 2, \quad e_{256} = 1.$$

Lemma 18 (Subfield-subcode dimension). *For every $q \in \{2, 4, 16, 256\}$,*

$$\kappa_q \geq K - (e_q - 1)H_{\text{HDPC}}.$$

Moreover, a basis of $\mathcal{W}_q$ can be computed by Gaussian elimination over $\mathbb{F}_q$.

Proof. The proof is the same rank-nullity argument as Lemma 12. Fix an $\mathbb{F}_q$ -linear isomorphism

$$\iota_q : \mathbb{F}_{256} \longrightarrow \mathbb{F}_q^{e_q \times 1},$$

and extend it coordinatewise to the H_{HDPC} HDPC equations. For $z \in \mathbb{F}_q^{L \times 1}$, impose the binary LDPC equations, the expanded HDPC equations, and the padding equations. The resulting $\mathbb{F}_q$ -linear map has kernel exactly $\mathcal{W}_q$ and codomain dimension

$$S_{\text{LDPC}} + e_q H_{\text{HDPC}} + (K' - K).$$

Rank-nullity therefore gives

$$\begin{aligned} \kappa_q &\geq L - S_{\text{LDPC}} - e_q H_{\text{HDPC}} - (K' - K) \\ &= K - (e_q - 1)H_{\text{HDPC}}. \end{aligned}$$

Gaussian elimination computes the kernel. □

Thus

$$\kappa_2 \geq K - 7H_{\text{HDPC}}, \quad \kappa_4 \geq K - 3H_{\text{HDPC}}, \quad \kappa_{16} \geq K - H_{\text{HDPC}}, \quad \kappa_{256} = K.$$

The last equality also follows because C is injective and $\mathcal{W}$ has $\mathbb{F}_{256}$ -dimension K .

For $\mathbb{F}_4$, an exhaustive exact computation over the RFC 6330 parameter table gives

$$\kappa_4 = \begin{cases} 1, & K = 30, \\ \max(0, K - 3H_{\text{HDPC}}), & \text{every other RFC-valid } K. \end{cases}$$

For $\mathbb{F}_{16}$, a second exhaustive exact computation over the full RFC 6330 parameter table gives

$$\kappa_{16} = \max(0, K - H_{\text{HDPC}})$$

for every RFC-valid K , with no exceptions. Within each fixed- K' parameter band, decreasing K only adds padding constraints, so the corresponding constraint matrices are nested. Where the claimed dimension formula is positive, equality means that all constraint rows are independent; it suffices to verify this at the smallest K in that part of the band, because deleting padding rows preserves row independence. Where the claimed dimension formula is zero, equality means full column rank; it suffices to verify this at the largest K in that part of the band, because adding padding rows preserves full column rank. Any band that meets the relevant dimension boundary is checked directly at the crossing values. The $\mathbb{F}_{256}$ equality $\kappa_{256} = K$ is exact algebraically.

A.2 Projective shortening over $\mathbb{F}_q$

This subsection describes the search algorithm used in the experiments below. Given $2f + 1$ LT rows indexed by H , the algorithm searches $\mathcal{W}_q$ for a nonzero intermediate codeword whose transmitted image on those rows has weight at most f . We call this procedure **projective shortening**. The search follows one greedy path at a time and then explores additional paths that differ only when the greedy rule has a tie.

Fix q and choose a basis

$$w_1, \dots, w_{\kappa_q}$$

of $\mathcal{W}_q$. For each of the $2f + 1$ LT rows g_i indexed by H , define

$$b_i := (g_i w_1, \dots, g_i w_{\kappa_q}) \in \mathbb{F}_q^{1 \times \kappa_q},$$

and let

$$B_q := \begin{pmatrix} b_1 \\ \vdots \\ b_{2f+1} \end{pmatrix}.$$

For $x \in \mathbb{F}_q^{\kappa_q \times 1}$, the corresponding intermediate codeword is

$$w = \sum_{j=1}^{\kappa_q} x_j w_j,$$

and coordinate i evaluates to $b_i x$. Thus the search wants a nonzero x such that

$$\text{wt}(B_q x) \leq f,$$

or equivalently at least $f + 1$ of the values $b_i x$ are zero.

We first describe one greedy path. Suppose a set T of coordinates has already been forced to zero. Define the residual coefficient space

$$\mathcal{U}_T := \{x \in \mathbb{F}_q^{\kappa_q} : b_i x = 0 \text{ for every } i \in T\}.$$

Then

$$\dim \mathcal{U}_T = \kappa_q - \text{rank}_{\mathbb{F}_q}(B_{q,T}).$$

A path trades one residual dimension at a time for as many additional forced zeros as the current rows permit.

If $\kappa_q = 0$, there is no nonzero coefficient vector and the search returns no candidate. Otherwise, starting from $T = \emptyset$ and $\mathcal{U}_T = \mathbb{F}_q^{\kappa_q}$, one path repeats the following steps.

1. Restrict every row to the current residual space $\mathcal{U}_T$ and compute its dimension. A row that becomes zero already vanishes on every remaining candidate x .
2. If the residual dimension is at most the exact-enumeration cutoff, enumerate every projective direction in the residual space and terminate this path. The path succeeds if the minimum transmitted weight found is at most f .
3. Otherwise, group the nonzero restricted rows that are nonzero scalar multiples of one another. Such rows impose exactly the same zero constraint. Each such group is a **projective class**. If no nonzero restricted row remains, terminate this path without a candidate.

4. Choose a projective class of maximum cardinality and impose its common zero constraint. Because the restricted row is nonzero, this lowers the residual dimension by one while forcing every coordinate in that class to zero. Other rows may also become zero after the restriction. Then return to the first step, since previously different rows can become scalar multiples after a restriction.

The exact-enumeration test therefore occurs before any new projective-class constraint is imposed, including on the first iteration. The no-class stopping rule is conservative and can only decrease the reported search success fraction.

The exact-enumeration step uses one representative from each nonzero scalar class of coefficient vectors. Nonzero scalar multiples of x produce transmitted codewords with the same zero set and Hamming weight, so a d -dimensional residual space has only

$$N_q(d) = \frac{q^d - 1}{q - 1}$$

projective directions to check. In the experiments below, exact enumeration begins at

$$(q, d) = (4, 8), (16, 5), (256, 3),$$

for which

$$N_4(8) = 21,845, \quad N_{16}(5) = 69,905, \quad N_{256}(3) = 65,793.$$

These dimensions keep the exact enumeration below about 70,000 projective directions. They are computational limits, not coding-theoretic thresholds.

A single greedy path is deterministic except when several projective classes tie for maximum cardinality. The first greedy path resolves every such tie by a fixed canonical order. Each subsequent greedy path follows exactly the same greedy rule but chooses reproducibly at random among the classes tied for the current maximum. It never deliberately chooses a smaller class. Thus the explored paths differ only in their tie choices; they do not exhaust the full tie-breaking tree. The search succeeds if any explored path finds such a witness.

A.3 Full-domain experimental design

This subsection specifies the sampling, search, and matched-random controls for the full-domain empirical results below. Write

$$\Delta := f + 1 - K.$$

For each f and sample number, set $n = 3f + 1$. Consider the challenger that chooses E uniformly among the n -element subsets of the full 24-bit ESI domain of RFC 6330 [5] and, after m is fixed, chooses H uniformly among the $(2f + 1)$ -subsets of E . The induced marginal distribution of H is uniform over all $(2f + 1)$ -subsets of the full ESI domain. Since the search depends only on H , the experiment samples H directly from this marginal distribution. Each sampled H is generated once and reused for every applicable value of Δ and every field. Thus changing the field or overhead changes the induced code and search problem while holding H fixed. The search explores 40 greedy paths, with no early stopping and with the exact-enumeration limits above. The canonical-prefix experiment later uses a different ESI sampling design, described in its own subsection.

The zero-overhead experiment uses

$$f = 9, 10, \dots, 45, \quad \Delta = 0,$$

with 100 sampled choices of H at each point. The main positive-overhead grid uses

$$f \in \{12, 16, 20, \dots, 96\}, \quad 1 \leq \Delta \leq 40, \quad K = f + 1 - \Delta \geq 10.$$

This gives 690 valid (f, Δ) points, with 100 sampled choices of H at every point. A separate $\mathbb{F}_{256}$ deep-tail experiment uses

$$f \in \{64, 80, 96\}, \quad 41 \leq \Delta \leq 68, \quad K \geq 10,$$

giving 71 additional valid points. The same choices of H are reused whenever these phases share the same f .

For every full-domain empirical point we also run the identical search on a matched random linear code. The matched code has length $2f + 1$ and dimension κ_q , and is sampled uniformly from full-column-rank matrices over $\mathbb{F}_q$. The RaptorQ and matched-random searches use the same field arithmetic, projective normalization, tie handling, exact-enumeration dimension, and success test. Every generated random matrix is checked for full column rank.

The empirical quantity reported below is the *observed search success fraction*. A success means that the specified finite search found a nonzero intermediate codeword whose transmitted image has weight at most f . A failed search does not imply that no such witness exists unless the entire relevant code space was enumerated.

A.4 Empirical attack range

This subsection first compares the fields at zero overhead and then measures the positive-overhead attack region on a regular two-dimensional grid.

Empirical $\Delta = 0$ transition. Figure 1 shows the observed search success fraction at $K = f + 1$ over the common range $9 \leq f \leq 45$. The exact subfield dimensions give a useful baseline for reading the figure. Throughout this range $H_{\text{HDPC}} = 10$. Table 1 compares the first tested f in this range at which each subfield code is nonzero with three empirical search markers. The first column is exact. The remaining columns describe the stated 100-sample finite search.

field	first tested f with $\kappa_q > 0$	first observed success	first $\geq 5/100$	first $\geq 10/100$
$\mathbb{F}_4$	29	36	38	38
$\mathbb{F}_{16}$	10	20	27	30
$\mathbb{F}_{256}$	9	9	9	9

Table 1: Exact dimension availability and empirical $\Delta = 0$ search markers. The last three columns are observed outcomes of the specified finite search, not attack-existence probabilities or mathematical thresholds.

The gap between nonzero dimension and observed low-weight codewords is substantial over the two proper larger subfields. The quaternary code first becomes nonzero at $f = 29$, but the search finds no witness until $f = 36$ and does not reach 5/100 until $f = 38$. The $\mathbb{F}_{16}$ code is already nonzero at $f = 10$, but the first observed witness occurs at $f = 20$, with the 5/100 and 10/100 markers at $f = 27$ and $f = 30$. Over the full field, the search succeeds on all 100 sampled choices of H at every tested value $f = 9, \dots, 45$. The matched random codes show the same broad zero-overhead field ordering, so this early full-field transition by itself is not evidence of special

RaptorQ structure. For comparison, at $\Delta = 0$ the binary existence theorem in Section 3 first applies at $f = 77$, and the efficient randomized binary attack first applies at $f = 78$.

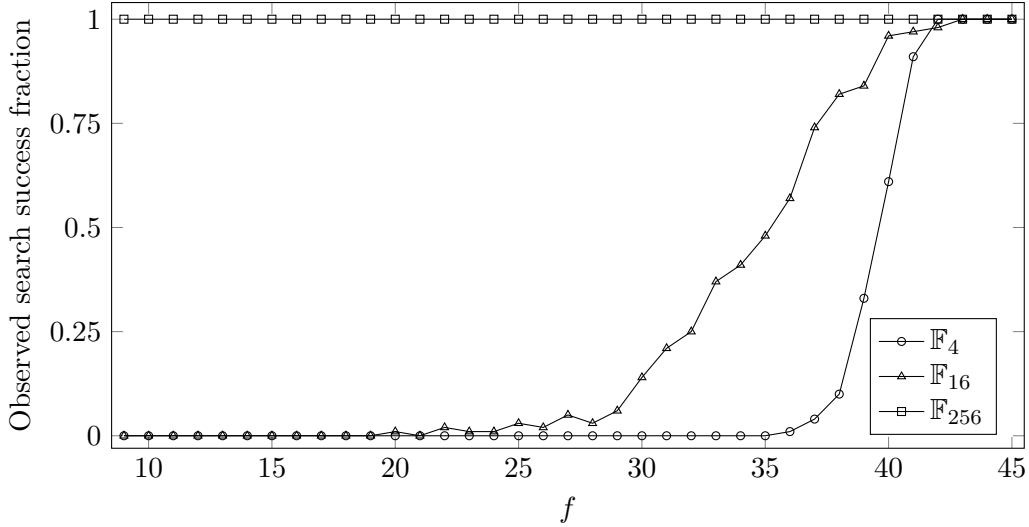


Figure 1: Empirical $\Delta = 0$ transition over the common range $9 \leq f \leq 45$. Each point is the observed success fraction of the search exploring 40 greedy paths on 100 independently sampled choices of H from the full 24-bit domain. The curves summarize search outcomes, not attack-existence probabilities or mathematical thresholds.

Positive overhead. The practically relevant question is how far these attacks persist after decoding overhead is added. Figure 2 shows all 690 included points of the regular positive-overhead grid. Each cell is an actual measurement. No interpolation is used. White cells are excluded because this experiment uses only $K \geq 10$. Among included cells, zero observed success is shown in light gray and increasing observed success is shown by progressively darker gray.

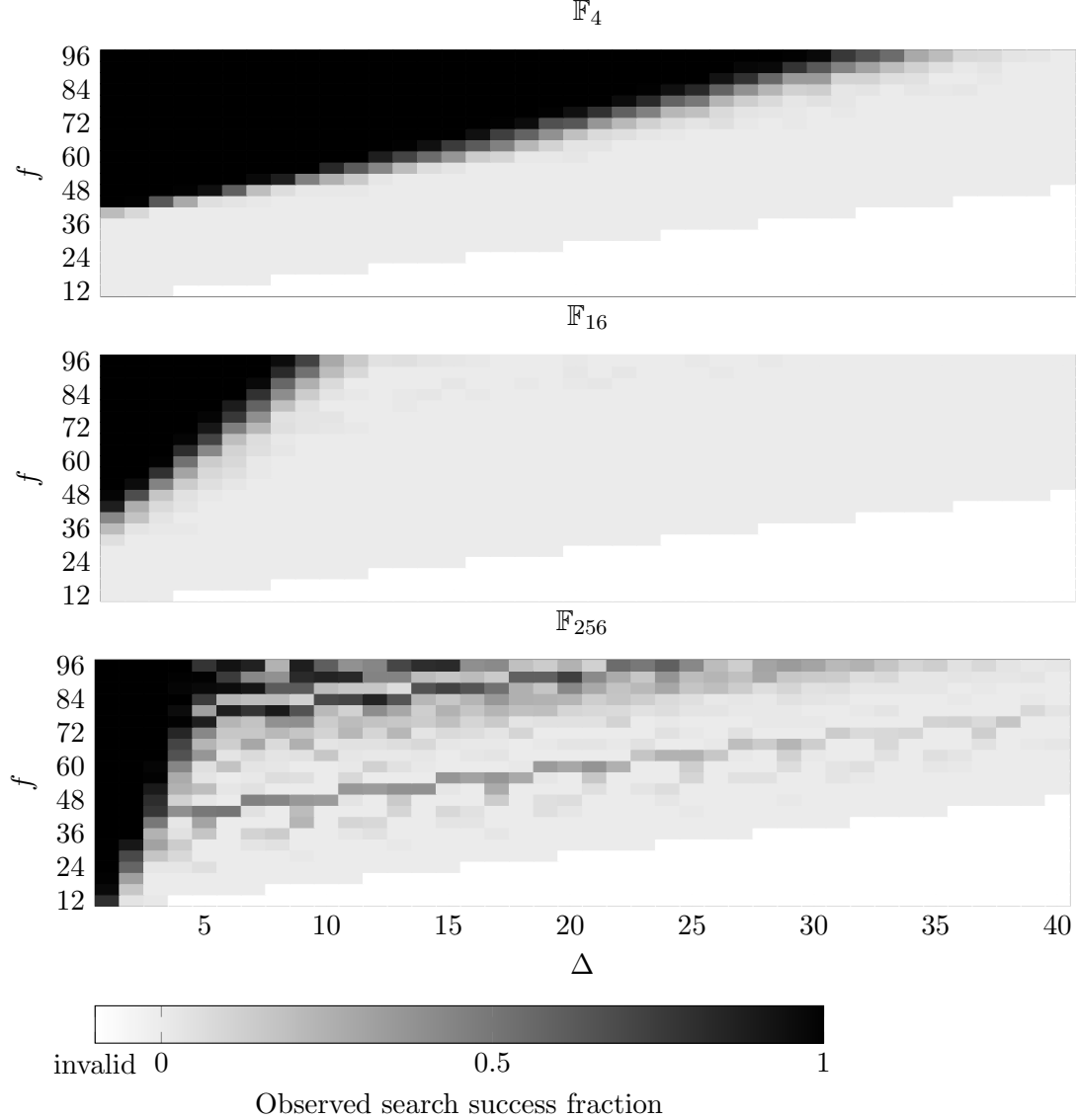


Figure 2: Observed search success fractions on the regular positive-overhead grid $f \in \{12, 16, 20, \dots, 96\}$, $1 \leq \Delta \leq 40$, and $K = f + 1 - \Delta \geq 10$. Every included cell uses 100 sampled choices of H and the search exploring 40 greedy paths. White cells are excluded by the $K \geq 10$ experimental range; included zero-success cells are light gray. No interpolation is used.

The three heat maps show that field size does not impose a single ordering at positive overhead. The $\mathbb{F}_4$ region is broad and comparatively regular as f grows. The $\mathbb{F}_{16}$ region is substantially narrower despite its much earlier $\Delta = 0$ transition. The $\mathbb{F}_{256}$ region contains a narrower high-success core together with many irregular secondary regions. Across the 690 valid grid points, the search finds at least one witness at 347, 145, and 427 points over $\mathbb{F}_4$, $\mathbb{F}_{16}$, and $\mathbb{F}_{256}$, respectively. The corresponding counts for at least 5/100 observed successes are 312, 106, and 310, and for at least 10/100 are 297, 94, and 262. Thus the full field gives the broadest set of points with any observed attack, while the quaternary search retains a broader regular region once a modest empirical-frequency requirement is imposed.

The separate $\mathbb{F}_{256}$ deep-tail sweep confirms that the secondary regions continue beyond $\Delta = 40$. At $f = 64$, none of the valid tail points $\Delta = 41, \dots, 55$ succeeds. At $f = 80$, seven tail points succeed, with the deepest observed success at $\Delta = 52$. At $f = 96$, thirteen tail points succeed, with the deepest observed success at $\Delta = 65$. The success counts at these deepest points are 1/100 and 2/100, respectively. These are sparse empirical successes, not monotone boundaries, but they show that the full-field secondary regions are not artifacts of a few hand-picked values of Δ .

A.5 Canonical ESI prefix

The full-domain experiments above use a challenger that samples E uniformly and then samples H uniformly from E . A natural systematic RaptorQ configuration may instead fix

$$E = \{0, 1, \dots, 3f\}.$$

The first K positions are **systematic**: for ESIs $0, \dots, K - 1$, the encoding symbols equal the K source symbols. We therefore repeat the same 40-path search with this fixed E and H chosen uniformly among its $(2f + 1)$ -subsets, using 100 samples at every tested point. At $\Delta = 0$, we test $f = 32, \dots, 45$ over $\mathbb{F}_4$, $f = 20, \dots, 35$ over $\mathbb{F}_{16}$, and $f = 9, \dots, 20$ over $\mathbb{F}_{256}$. For positive overhead, we test the 60 valid points with

$$f \in \{24, 32, 40, 48, 64, 80, 96\}, \quad \Delta \in \{1, 2, 4, 6, 8, 12, 16, 24, 32, 40\}, \quad K \geq 10,$$

together with the 11 valid $\mathbb{F}_{256}$ points having $f \in \{64, 80, 96\}$ and $\Delta \in \{48, 56, 61, 64, 68\}$, and the two additional points (64, 14) over $\mathbb{F}_4$ and (96, 31) over $\mathbb{F}_{256}$ used below.

Table 2 summarizes the canonical-prefix comparison. For the $\Delta = 0$ columns, each triple gives the first sampled f with any observed success, at least 5/100 successes, and at least 10/100 successes. The positive-overhead columns sum the observed successes over the common 60-point grid and its 100 samples per point.

field	$\Delta = 0$ markers		positive-grid successes / 6000	
	full-domain	canonical	full-domain	canonical
$\mathbb{F}_4$	36/38/38	36/38/39	2676	2665
$\mathbb{F}_{16}$	20/27/30	22/27/30	1432	1442
$\mathbb{F}_{256}$	9/9/9	9/9/9	2253	2092

Table 2: Full-domain and canonical-prefix ESI profiles under the same finite search budget. The $\Delta = 0$ triples are descriptive search markers; the positive-grid columns sum successes over 60 common points with 100 samples per point.

The canonical prefix leaves the $\Delta = 0$ picture essentially unchanged. Over $\mathbb{F}_4$ and $\mathbb{F}_{16}$, the positive-overhead totals are also nearly unchanged. Over $\mathbb{F}_{256}$, the aggregate success count is lower, but the effect is strongly point-dependent rather than a uniform weakening: among the 60 common grid points the canonical count is higher at 7, equal at 28, and lower at 25. For example, $(f, \Delta) = (48, 8)$ changes from 43/100 full-domain to 9/100 canonical, whereas (64, 16) changes from 2/100 to 33/100 and (96, 31) from 21/100 to 44/100. The deep secondary region also remains visible: at (96, 61) the search succeeds on 6/100 full-domain samples and 9/100 canonical samples.

Within the canonical experiment itself, among mixed-success positive-grid points, a larger number of systematic symbols in H is not consistently associated with lower search success. Thus the canonical prefix changes the numerical $\mathbb{F}_{256}$ landscape at some points, but it neither removes the attacks nor gives evidence of a monotone mitigation from the number of systematic symbols.

A.6 Comparison with matched random codes

This subsection asks whether the observed field dependence is generic behavior for a q -ary linear code of the same length and dimension, or whether standardized RaptorQ contributes additional structure. At every empirical point, the matched random code is a uniformly random full-column-rank

$$(2f + 1) \times \kappa_q$$

matrix over $\mathbb{F}_q$. The RaptorQ and random-code searches use the same field arithmetic, projective normalization, tie handling, exact-enumeration limit, and success test, and each main-grid instance explores 40 greedy paths. Every random matrix is checked for full column rank.

We also use a simple calculation to ask whether a low-weight codeword should exist at all in a matched random code. For a uniformly random κ_q -dimensional linear code of length $2f + 1$ over $\mathbb{F}_q$, let E_q denote the expected number of nonzero codewords of weight at most f :

$$E_q = (q^{\kappa_q} - 1) \frac{\sum_{w=1}^f \binom{2f+1}{w} (q-1)^w}{q^{2f+1} - 1}. \quad (1)$$

The first factor counts candidate codewords. The second is the probability that a uniformly random nonzero vector in $\mathbb{F}_q^{2f+1}$ has weight at most f . Thus E_q is the average number of qualifying codewords when many matched random codes are drawn. It is an existence benchmark, not a measure of how easily the search finds such a codeword. If $E_q < 1$, Markov's inequality gives

$$\Pr[\text{a nonzero codeword of weight at most } f \text{ exists}] \leq E_q.$$

If E_q is large, the conclusion is weaker: many such codewords exist on average, but they need not be easy for this search to find. The matched random-code search provides that separate search comparison.

There is a simple large- f guide to the dependence on overhead. Suppose the loss $K - \kappa_q$ stays bounded independently of f , as guaranteed by Lemma 18 while H_{HDPC} is fixed. With $K = f + 1 - \Delta$, the exponentially varying part of E_q , up to factors polynomial in f , is

$$\left(\frac{4(q-1)}{q} \right)^f q^{-\Delta}.$$

The key dependence on overhead is therefore $q^{-\Delta}$. Wherever increasing Δ by one also lowers κ_q by one, each additional unit of overhead reduces this random-code existence benchmark by about a factor of q . If Δ/f is held fixed, that reduction is exponential in f .

Table 3 summarizes the matched comparison over all 690 valid positive-overhead grid points. There are 69,000 RaptorQ samples and 69,000 matched random-code samples per field. The last column is the mean, over the 690 points, of the absolute difference between the two success counts.

The $\mathbb{F}_4$ and $\mathbb{F}_{16}$ searches therefore track their matched random-code controls very closely in aggregate. This agreement is empirical and does not assert that the code ensembles are identical. The $\mathbb{F}_{256}$ results are qualitatively different. On the main grid, the RaptorQ search has at least one success at 427 points, compared with only 80 points for the matched random codes.

At $(f, \Delta) = (96, 24)$,

$$E_{256} = 2^{-3.664} \approx 0.079.$$

field	RaptorQ successes	random-code successes	mean absolute difference
$\mathbb{F}_4$	26,099/69,000	26,045/69,000	0.66/100
$\mathbb{F}_{16}$	7,476/69,000	7,416/69,000	0.40/100
$\mathbb{F}_{256}$	13,360/69,000	5,025/69,000	12.08/100

Table 3: Aggregate observed search successes over the regular positive-overhead grid for standardized RaptorQ and matched random full-rank linear codes. The aggregate counts summarize the pointwise data; they do not replace the heat maps.

By Markov’s inequality, a uniformly random code with the same length and dimension contains a nonzero codeword of weight at most f with probability at most about 7.9%. The matched-random search succeeds on 0/100 samples, while the standardized RaptorQ search succeeds on 59/100. At (96, 31),

$$E_{256} = 2^{-59.664} \approx 1.1 \times 10^{-18},$$

while the RaptorQ search succeeds on 21/100 samples and the matched-random search on none. In the deep tail, the RaptorQ search records 76 successes over the 7,100 tested instances, while the matched-random search records none. At (96, 61),

$$\log_2 E_{256} = -299.664,$$

yet the RaptorQ search still succeeds on 6/100 samples. An explicit witness from this point was independently validated. These deep $\mathbb{F}_{256}$ successes cannot plausibly be explained by ordinary matched-random-code behavior.

A.7 What projective shortening exploits

This subsection measures how much the greedy search actually gains from grouping proportional rows. At a fixed residual dimension d , a smaller field has fewer projective directions

$$N_q(d) = \frac{q^d - 1}{q - 1},$$

so accidental projective coincidences should be more common for smaller q . The data confirm this generic effect, while also showing field-dependent excesses over the matched-random baseline.

For every main-search path, we record the largest projective class at each residual dimension before exact enumeration. We also record the cumulative projective saving, defined as the number of rows obtained from the chosen projective classes beyond one row per residual dimension consumed. Table 4 compares RaptorQ with the matched random codes at the same residual dimension. A nontrivial step is one at which the largest current projective class contains at least two rows.

The $\mathbb{F}_4$ row matches the simple field-size intuition: nontrivial projective classes are relatively common, but they occur at essentially the same rate in the matched random codes. Over $\mathbb{F}_{16}$, the absolute rate is much smaller, but the RaptorQ rows already show a measurable excess over their random baseline at residual dimension 9. That excess is not large enough to produce a correspondingly large difference in aggregate search success. Over $\mathbb{F}_{256}$, the excess is much stronger and grows as the residual dimension decreases. At residual dimension 4, the mean largest class is still only 1.136, so the gain comes from repeated modest coincidences rather than from a few enormous projective classes.

field	d	nontrivial steps RQ/random	mean cumulative saving RQ/random
$\mathbb{F}_4$	9	7.41%/7.30%	0.111/0.098
$\mathbb{F}_{16}$	9	0.103%/0.00004%	0.019/0.0000004
$\mathbb{F}_{256}$	9	1.011%/0	0.067/0
$\mathbb{F}_{256}$	6	4.20%/0	0.155/0
$\mathbb{F}_{256}$	4	13.18%/0.026%	0.367/0.00026

Table 4: Projective-class statistics pooled over the regular positive-overhead grid at matched residual dimensions. The comparison is between standardized RaptorQ and matched random codes under the same search procedure.

Search sensitivity. The search still explores only finitely many greedy paths, so its success fraction depends on how computation is divided between path breadth and exact enumeration. A separate paired diagnostic compares the main 40-path search with one canonical greedy path followed by deeper exact enumeration. At $\mathbb{F}_4$, (64, 14), the main search succeeds on 93/100 samples, while the single canonical path with exact enumeration at residual dimension 12 succeeds on 73/100. At $\mathbb{F}_{16}$, (80, 8), the corresponding counts are 18/100 and 5/100. Over $\mathbb{F}_{256}$, the shallow point (64, 4) gives 65/100 versus 30/100, whereas the deeper points (96, 24) and (96, 31) give 59/100 versus 66/100 and 21/100 versus 25/100. These searches spend computation differently and are not equal-work comparisons. The reported success fractions should therefore be read as outcomes of the specified finite search, not as attack-existence probabilities.

A.8 Where the $\mathbb{F}_{256}$ structure enters

This subsection asks where the extra rank deficiency seen over the full field enters the RaptorQ construction. The matched random-code and projective-class comparisons show that the deep $\mathbb{F}_{256}$ behavior contains additional structure. The following rank decomposition separates dependencies already present among the raw LT rows from those introduced by restricting to valid RaptorQ intermediate codewords.

For a coordinate set S , let

$$G_S$$

be the raw binary RFC 6330 LT-row matrix, and let

$$B_{q,S}$$

be the induced matrix after restricting the intermediate code to $\mathcal{W}_q$. Then

$$|S| - \text{rank}(B_{q,S}) = (|S| - \text{rank}(G_S)) + (\text{rank}(G_S) - \text{rank}(B_{q,S})). \quad (2)$$

The first term is rank loss already present in the raw LT rows. The second is the additional loss from restricting to the intermediate code. Because G_S is binary, its rank is unchanged by scalar extension to $\mathbb{F}_4$, $\mathbb{F}_{16}$, or $\mathbb{F}_{256}$.

For $q = 256$, the second term has an exact row-space interpretation. Set

$$R_{\text{pre}} := \begin{pmatrix} P \\ G_{\text{PAD}} \end{pmatrix}.$$

The argument of Lemma 12, run over $\mathbb{F}_{256}$ without the subfield restriction, gives $\mathcal{W} = \ker R_{\text{pre}}$. For any two matrices G and R with the same number of columns, rank-nullity gives

$$\text{rank}(G|_{\ker R}) = \text{rank} \begin{pmatrix} G \\ R \end{pmatrix} - \text{rank}(R),$$

because both sides equal $\dim \ker R - \dim(\ker G \cap \ker R)$. Since $B_{256,S}$ represents the restriction of G_S to $\mathcal{W}$,

$$\text{rank}(G_S) - \text{rank}(B_{256,S}) = \dim(\text{rowspan}(G_S) \cap \text{rowspan}(R_{\text{pre}})).$$

Equivalently,

$$\text{rank}(B_{256,S}) = \text{rank} \begin{pmatrix} G_S \\ P \\ G_{\text{PAD}} \end{pmatrix} - \text{rank} \begin{pmatrix} P \\ G_{\text{PAD}} \end{pmatrix}.$$

For every successful RaptorQ search, let Z be the set of all coordinates indexed by H on which the final transmitted codeword is zero. Applying (2) with $S = Z$ makes the calculation depend only on the final witness, not on the greedy path that found it. Table 5 gives means over all successful $\mathbb{F}_{256}$ searches at several representative positive-overhead points.

f	Δ	successes	total rank loss	raw LT	restriction
48	2	100	3.79	0.03	3.76
80	8	89	28.63	10.89	17.74
96	24	59	45.49	29.29	16.20
96	31	21	35.52	31.67	3.86
96	61	6	65.67	46.17	19.50

Table 5: Mean complete-zero-set rank decomposition over the successful standardized-RaptorQ searches at selected $\mathbb{F}_{256}$ positive-overhead points. The last three columns satisfy total rank loss = raw-LT rank loss + restriction loss.

The mechanism changes with depth. At $(48, 2)$, almost all of the rank loss comes from restricting to the intermediate code. At $(80, 8)$ and $(96, 24)$, both terms contribute substantially. At $(96, 31)$, the raw LT rows account for most of the rank loss, and they remain the larger contribution at the much deeper point $(96, 61)$. The same decomposition is available for every successful search. Pooled over all successful positive-overhead witnesses, the mean raw-LT and restriction terms are 0.98 and 49.23 over $\mathbb{F}_4$, 0.17 and 17.01 over $\mathbb{F}_{16}$, and 9.06 and 8.89 over $\mathbb{F}_{256}$. Thus the smaller-subfield behavior is overwhelmingly restriction driven, whereas the full-field search exposes substantial dependencies already present among the standardized LT rows.

Taken together, the matched random-code comparison, the projective-class data, and the rank decomposition suggest a useful empirical distinction. Over $\mathbb{F}_4$, the subfield restriction produces attack behavior extremely close to that of matched random linear codes. Over $\mathbb{F}_{16}$, aggregate search success remains close to matched random behavior even though small projective-geometry deviations are visible. Over the full field, additional RaptorQ structure becomes prominent, producing irregular secondary attack regions and projective coincidences far beyond the matched random baseline. This is an empirical interpretation of the measured data, not a structural theorem about all RaptorQ row sets.

A.9 An exact quaternary consequence

This subsection separates one exact consequence that holds for every possible H from the empirical search results above. The empirical searches concern random choices of H and a finite search budget. The quaternary subfield also gives a separate result that holds for every possible H . Fix any collection of $2f + 1$ RaptorQ encoding rows indexed by H and evaluate them on a basis of $\mathcal{W}_4$. This gives

$$B_4 \in \mathbb{F}_4^{(2f+1) \times \kappa_4}.$$

If B_4 has a nonzero kernel, the corresponding nonzero intermediate codeword vanishes on every row indexed by H and the attack is immediate. Otherwise the image of B_4 is a quaternary linear code of length $2f + 1$ and dimension κ_4 . The low-weight characterization of Lemma 2 and the minimum-distance formulation of Corollary 3 are field-independent and apply verbatim over $\mathbb{F}_4$. If every $(f + 1)$ -subset determined the source block, every nonzero codeword in this image would therefore have weight at least $f + 1$. Thus this hypothesis implies a quaternary

$$[2f + 1, \kappa_4, d \geq f + 1]$$

linear code.

We apply the Delsarte linear-programming bound [2]. The bounds in Table 6 were verified with exact rational arithmetic. Here U_4 is an upper bound on the dimension of a quaternary linear code of length $2f + 1$ and minimum distance at least $f + 1$.

f	κ_4	U_4
72	43	42
73	44	43
74	45	43
75	46	44
76	47	44

Table 6: Exact quaternary Delsarte bounds at $K = f + 1$. In every row $\kappa_4 > U_4$, so the required minimum distance is impossible.

It follows that for every $72 \leq f \leq 76$ with $K = f + 1$, every collection of $2f + 1$ standardized RaptorQ rows contains a bad $(f + 1)$ -subset. The binary theorem of Section 3 applies at $K = f + 1$ from $f = 77$ onward over the RFC-valid parameter range. Combining the two results shows that every collection of $2f + 1$ standardized RaptorQ rows contains a bad $(f + 1)$ -subset for

$$\boxed{f \geq 72, \quad K = f + 1.}$$

This conclusion is existential. The Delsarte argument proves that a bad subset exists for every realized H , but it does not identify the subset or give an efficient method for finding it.

A.10 Scope and verification

This subsection separates the algebraic, exact finite, and empirical statements and records the corresponding verification boundaries. Lemma 18 is algebraic. The displayed formulas for κ_4 and κ_{16} are exact finite computations over every RFC-valid source dimension. The Delsarte bounds are exact rational certificates. The $\Delta = 0$, positive-overhead, canonical-prefix, matched random-code, projective-class, and search-sensitivity results are observed outcomes for the stated ESI distributions, sample counts, and finite search procedures. A failed search is not a negative code result.

The full-domain empirical run contains 451,400 saved instance records. The canonical-prefix experiment contains 23,500 additional saved instances. The complete-zero-set rank decomposition was checked automatically for all 52,409 successful RaptorQ witnesses in the full-domain run. Every applicable $\mathbb{F}_{256}$ precode-intersection identity passed. Five predetermined explicit witnesses from the full-domain run, including a deep $\mathbb{F}_{256}$ witness at $(f, \Delta) = (96, 61)$, were checked against an independent RaptorQ implementation. The canonical-prefix experiment independently checks five

additional selected witnesses. The independent checks reconstruct the nonzero source vector and intermediate codeword, evaluate every ESI in H , and verify the complete zero set and transmitted weight. They do not exercise decoder behavior, which is not required for the explicit witness rank argument.

The Appendix A reproducibility artifact [39] records the paper-facing empirical data and summaries, selected witness files, and the code, seeds, configurations, provenance commitments, and regeneration scripts needed to reproduce the omitted raw campaign data. It also contains the exhaustive $\mathbb{F}_4$ and $\mathbb{F}_{16}$ dimension certificates and the exact quaternary Delsarte certificates, with the older GF(4) material preserved under its original frozen provenance.

B Stronger larger- n attacks on R10

Section 4 extends the generic Plotkin attack to

$$n = 3f + 1 + t,$$

giving an attack whenever

$$K > t + \lfloor \log_2(2f + 2) \rfloor.$$

That argument uses no structure specific to R10. This appendix exploits the standard R10 encoding structure to obtain much stronger constructive attacks.

For $n \geq 1$, write

$$P_n := \{0, 1, \dots, n - 1\}$$

for the canonical ESI prefix. For every standardized $34 \leq K \leq 8192$, every $f \geq 1$ satisfying $4f \leq 65520$, and every

$$3f + 1 \leq n \leq 4f,$$

we show that, for the canonical assignment $E = P_n$, a deterministic efficient adversary wins the Byzantine erasure game for every subsequent legal choice of H . The same canonical-prefix result also holds at the isolated value $K = 30$. At $n = 4f$, the generic Plotkin attack reaches only

$$K > f - 1 + \lfloor \log_2(2f + 2) \rfloor \approx f.$$

For $f \geq 3$, this condition lies entirely above the traditional range $K \leq f + 1$, whereas the R10-specific attack applies throughout the standardized range $K \geq 34$.

Recall that Section 2.1 calls a nonzero input vector x a *witness* when its encoded word has sufficiently small Hamming weight. In the R10 setting, a witness is a nonzero source vector. By linearity, it may equivalently be viewed as the difference between two source vectors. When this definition is applied to the Byzantine erasure game, the witness depends on the realized E and H . We call a nonzero source vector s a **global witness** for E if its transmitted codeword on E has weight at most f . A global witness is chosen independently of H , and for every legal $H \subseteq E$ its restriction is a witness in the sense of Section 2.1. For every covered pair (K, f) , the global witness needed for the canonical-prefix result below has already been precomputed.

If $K > 2f$, no witness is needed: the adversary must choose only $n - 2f \leq 2f < K$ rows, so any such set is rank deficient. Suppose $K \leq 2f$. The finite certificates below give a global witness for P_{4f} at every K and f in the canonical-prefix range above. Restriction to P_n cannot increase the transmitted weight, so the same source vector is also a global witness for P_n . Equivalently, it

has at least $n - f$ zero positions in P_n . Every legal $H \subseteq P_n$ has size $n - f$, and therefore contains at least

$$(n - f) + (n - f) - n = n - 2f$$

of these zero positions. After H is revealed, the adversary chooses any $n - 2f$ of them. Thus the same global witness works for every subsequent legal choice of H .

Randomizing the transmitted ESI values does not eliminate the attack. We call a nonzero source vector a **random-ESI witness** if it is fixed independently of f , n , and the sampled ESI set E , and, for every admissible f and n , it is a global witness for E with probability at least $1/2$ when E consists of n distinct ESIs sampled uniformly without replacement from

$$\{0, 1, \dots, 65520\}.$$

For every standardized $34 \leq K \leq 8192$ and also for $K \in \{25, 29, 30, 31\}$, a random-ESI witness depending only on K has already been precomputed. Whenever it is a global witness for the sampled E , it works for every subsequent legal choice of H .

The next subsection introduces the certificate representation used for both results and explains how the certificates are found.

B.1 R10 source and intermediate coordinates

We describe the successive linear representations used to obtain and verify the certificates. Let $s \in \mathbb{F}_2^K$ be a standard R10 source vector. Standard systematic R10 admits an invertible rate-one change of source coordinates [4, 18]. Let

$$T \in \mathbb{F}_2^{K \times K}$$

be the corresponding invertible matrix, and set

$$x := T^{-1}s.$$

The change of coordinates is chosen so that the transmitted symbols at ESIs $0, \dots, K - 1$ are exactly the coordinates of s . We call x the *non-systematic source coordinates*. Thus $s = Tx$, and the standard and non-systematic source coordinates describe the same transmitted R10 codeword.

Next, let p denote the R10 precode. Let G_L and G_H denote the standard R10 LDPC and Half-code maps. Writing the intermediate coordinates in source, LDPC, and Half-code order, set

$$z := p(x) = (x, G_L x, G_H(x, G_L x)).$$

The map p is a bijection from $\mathbb{F}_2^K$, viewed in non-systematic source coordinates, onto the K -dimensional binary R10 intermediate code $\mathcal{I}_K := p(\mathbb{F}_2^K)$. Thus $z \in \mathcal{I}_K$ is the intermediate codeword immediately before LT encoding.

Finally, for an ESI set E , let G_E denote the matrix of the standard R10 LT rows indexed by E . Set

$$c_E := G_E z,$$

so c_E is the corresponding transmitted codeword on E . The complete sequence is therefore

$$s \xrightarrow{T^{-1}} x \xrightarrow{p} z \xrightarrow{G_E} c_E.$$

In particular,

$$s \longleftrightarrow x \longleftrightarrow z$$

are in bijective correspondence. Every nonzero intermediate codeword $z \in \mathcal{I}_K$ therefore corresponds to a unique nonzero standard R10 source vector.

The finite certificates used below are sparse nonzero intermediate codewords $z \in \mathcal{I}_K$. Every selected intermediate codeword has Hamming weight 4 or 6. For every standardized $4 \leq K \leq 47$, the artifact exhaustively enumerates all valid intermediate codewords through weight 6 and selects the certificates directly from this search. For larger K , direct enumeration is substantially more expensive. A much cheaper restricted search uses the non-systematic source coordinates and tests source vectors of the form

$$x = e_a + e_b,$$

where e_j is the j th standard basis vector. Empirically, this search produces the weight-4 or weight-6 intermediate codewords used for most larger K . The shortcut is only a certificate-generation method: the proofs and verification use the selected intermediate codewords and their LT images directly, and do not rely on the search heuristic.

These observations suggest a useful decomposition of why the searches succeed. Sparse intermediate codewords are naturally favorable for the LT stage: the R10 LT rows are sparse, and any LT row whose support is disjoint from that of the intermediate codeword contributes a zero. The more R10-specific phenomenon is that the precode repeatedly produces valid intermediate codewords of weight 4 or 6, often from non-systematic source vectors of the form $e_a + e_b$; such behavior would be highly atypical for a generic binary linear code. Together, these observations give some intuition for why the finite searches below are successful.

The direct low-weight search is especially simple for R10 because the intermediate code is binary: once a support is chosen, every nonzero coordinate on that support is forced to be 1, so validity can be tested using only XORs. For RaptorQ, a low-weight intermediate codeword also has unknown nonzero field coefficients, so an analogous exhaustive search must determine both the support and those coefficients. The subfield experiments of Appendix A provide a different way to restrict that search space.

B.2 Canonical prefixes

Proposition 19 (Canonical-prefix sparse intermediate certificates). *Let K be a standardized R10 source size satisfying*

$$K = 30 \quad \text{or} \quad 34 \leq K \leq 8192.$$

For every $f \geq 1$ satisfying

$$4f \leq 65520, \quad K \leq 2f,$$

there is a nonzero valid intermediate codeword $z \in \mathcal{I}_K$ of Hamming weight 4 or 6 such that

$$\text{wt}(G_{P_{4f}} z) \leq f.$$

For $K = 30$ and for every $35 \leq K \leq 8192$, one fixed z_K , depending only on K , works for every eligible f . For $K = 34$, two fixed weight-4 intermediate codewords suffice, with their validity ranges jointly covering every eligible f .

Proof. This is a finite computer-assisted statement. The artifact of Section B.4 stores the selected intermediate supports and independently verifies that each support defines a nonzero codeword in $\mathcal{I}_K$ and satisfies every claimed prefix-weight bound. $\square$

Corollary 20 (Canonical-prefix attack for standard R10). *Let K be a standardized R10 source size satisfying*

$$K = 30 \quad \text{or} \quad 34 \leq K \leq 8192.$$

Let $f \geq 1$ satisfy $4f \leq 65520$, and let

$$3f + 1 \leq n \leq 4f.$$

Consider any legal challenger $\mathcal{C}$ that chooses

$$E = P_n$$

in the Byzantine erasure game. There is a deterministic efficient adversary that wins against $\mathcal{C}$. More precisely, after receiving any legal

$$H \subseteq P_n, \quad |H| = n - f,$$

the adversary finds a set

$$S \subseteq H, \quad |S| = n - 2f,$$

whose standard R10 rows have rank less than K .

Proof. If $K > 2f$, then

$$|S| = n - 2f \leq 2f < K,$$

so the adversary may choose any $n - 2f$ elements of H .

Suppose $K \leq 2f$. Choose an intermediate codeword z supplied by Proposition 19 for K and f . By restriction from P_{4f} to P_n ,

$$\text{wt}(G_{P_n} z) \leq f.$$

Thus the standard source vector corresponding to z is a global witness for P_n . Let $Z \subseteq P_n$ be the zero positions of $G_{P_n} z$. Then

$$|Z| \geq n - f.$$

Since $|H| = n - f$,

$$|Z \cap H| \geq (n - f) + (n - f) - n = n - 2f.$$

The adversary chooses any $n - 2f$ elements of $Z \cap H$ as S . The rows indexed by S annihilate the nonzero standard R10 source vector corresponding to z and hence have rank less than K . Thus S is bad and the adversary wins. The certificate is selected and its zero set on P_n determined before H is revealed. Once H is revealed, the only online work is intersection and truncation to $n - 2f$ positions. $\square$

For comparison, Paolini et al. [18] report low-weight codewords of weights 14, 25, and 26 for the canonical-prefix fixed-rate R10 truncations (128, 64), (512, 256), and (2048, 1024), respectively. These parameter choices have $n = 4f$ and $K = 2f$. Each reported weight is at most f , so the argument above gives the same bad-subset conclusion at each of these particular parameter choices. Their work studies minimum distance rather than adversarial subset selection; the examples are individual codewords rather than a parameter-wide attack result.

At $f = 128$ and $n = 512$, Proposition 16 gives the generic necessary condition $K \leq 135$ for escaping its attack guarantee. In particular, the generic attack reaches none of the traditional range $K \leq 129 = f + 1$. Corollary 20 covers every standardized $K \geq 34$. Thus escaping this particular

contiguous R10-specific attack guarantee requires $K \leq 33$, corresponding to communication blowup at least

$$\frac{512}{33} \approx 15.52.$$

The isolated certificate at $K = 30$ gives an additional attack below this contiguous threshold. This does not assert that $K \leq 33$ is safe or that 34 is an optimal threshold.

B.3 Uniform random ESIs

We next ask whether randomizing the transmitted ESI values avoids the attack and show that it does not. Let

$$\Omega := \{0, 1, \dots, 65520\}, \quad Q := |\Omega| = 65521.$$

For $z \in \mathcal{I}_K$, we call

$$\text{wt}(G_\Omega z)$$

the **full-cycle weight** of z .

Proposition 21 (Full-cycle R10 intermediate certificates). *For every standardized K satisfying*

$$K \in \{25, 29, 30, 31\} \quad \text{or} \quad 34 \leq K \leq 8192,$$

there is a fixed nonzero valid intermediate codeword $z_K \in \mathcal{I}_K$ of Hamming weight 4 or 6 whose full-cycle weight M_K satisfies

$$4M_K < Q.$$

The vector z_K depends only on K and is fixed independently of f , n , and any later ESI sample.

Proof. This is a finite computer-assisted statement. The artifact of Section B.4 stores each selected intermediate support, verifies that it defines a nonzero codeword in $\mathcal{I}_K$, evaluates its LT image on every ESI in Ω , and checks the stated inequality. $\square$

Corollary 22 (Uniform-random-ESI attack probability). *Let K be any standardized R10 source size covered by Proposition 21. The standard source vector corresponding to z_K is a random-ESI witness. More precisely, fix an integer $f \geq 1$ with $4f \leq 65520$ and let*

$$3f + 1 \leq n \leq 4f.$$

Let E be a set of n distinct ESIs sampled uniformly without replacement from Ω . With probability at least $1/2$ over E , the source vector corresponding to z_K is a global witness for E ; on this event, for every legal

$$H \subseteq E, \quad |H| = n - f,$$

a deterministic efficient adversary finds a set

$$S \subseteq H, \quad |S| = n - 2f,$$

whose standard R10 rows have rank less than K .

Proof. Let W be the weight of $G_E z_K$. Then

$$W \sim \text{Hypergeometric}(Q, M_K, n), \quad \mathbb{E}W = \frac{nM_K}{Q} < \frac{n}{4} \leq f.$$

A median of a hypergeometric random variable is the floor or ceiling of its mean [31, Corollary 13]; see also [32]. Hence a median of W is at most f , so

$$\Pr[W \leq f] \geq \frac{1}{2}.$$

On this event, the standard source vector corresponding to z_K is a global witness for E , and its transmitted codeword has at least $n - f$ zero positions in E . Every legal $H \subseteq E$ has size $n - f$, so it intersects these zero positions in at least

$$(n - f) + (n - f) - n = n - 2f$$

positions. The adversary chooses any $n - 2f$ of them as S . The rows indexed by S annihilate the nonzero standard R10 source vector corresponding to z_K and hence have rank less than K . Thus S is bad and the adversary wins. $\square$

Remark 23. The probability $1/2$ in Corollary 22 comes from requiring the fixed random-ESI witness to work for every subsequent legal choice of H . If, instead, after E is sampled, H is chosen uniformly among the $(n - f)$ -subsets of E , then marginally H is uniform among the $(n - f)$ -subsets of Ω . For the fixed random-ESI witness corresponding to z_K , let $Y := \text{wt}(G_H z_K)$; then

$$Y \sim \text{Hypergeometric}(Q, M_K, n - f), \quad \mathbb{E}Y = \frac{(n - f)M_K}{Q} < \frac{n - f}{4} \leq \frac{3f}{4}.$$

The witness fails to certify the attack only when $Y > f$, so in this weaker random- H model failure is an upper-tail event above a mean below $3f/4$.

B.4 Reproducibility artifact

The accompanying artifact [40] stores the intermediate supports used in Propositions 19 and 21. Its verifier checks that every stored support defines a nonzero valid R10 intermediate codeword, recovers the corresponding source vector, checks agreement with a pinned R10 encoder, and verifies every claimed canonical-prefix or full-cycle weight. The verifier consumes only the compact intermediate-support tables and the pinned implementation; it does not depend on discovery-search state.

For every standardized $4 \leq K \leq 47$, the artifact records exhaustive direct intermediate-code searches through weight 6. For larger K , most stored supports are obtained from the much cheaper source-pair search described above, with selected direct-search replacements. The artifact also preserves exploratory, post hoc analysis of structural features of the R10 precode that may help explain the observed sparse intermediate certificates. This material was developed after the witnesses were found empirically and is not used in the proofs or verification procedures.

C Experimental verification of the RaptorQ attack

We implemented the RaptorQ attack of Section 3 against RFC 6330 using cberner’s `raptorq` crate, version 2.0.1 [33]. The accompanying reproducibility artifact [38] records the attack implementation

and verification data. The attack code uses the crate’s RFC 6330 matrices to construct $\mathcal{W}_2$ and B , but leaves the encoder and decoder unchanged.

For each parameter choice, we run 25 independent attack trials. In each trial, we fix $E = \{0, \dots, 3f\}$, choose H uniformly among its $(2f + 1)$ -subsets, and use fresh randomness in the witness search. We find a nonzero x with $\text{wt}(Bx) \leq f$, choose $f + 1$ zero coordinates of Bx , and encode two distinct source blocks whose symbols agree at those ESIs. The ordinary RaptorQ decoder fails on every adversarial subset tested.

For comparison, for each parameter choice we choose a fresh H in the same way and test 200 uniformly random $(f + 1)$ -subsets with the same decoder. All 200 subsets decode successfully for each parameter choice.

f	K	κ_2	$f + 1$	adversarial subsets	random subsets
80	80	10	81	25/25 fail	0/200 fail
100	90	20	101	25/25 fail	0/200 fail
127	100	30	128	25/25 fail	0/200 fail
150	100	30	151	25/25 fail	0/200 fail

Table 7: End-to-end RaptorQ decoding results for adversarial and random subsets.

We also computed κ_2 for every standardized RaptorQ source dimension. In every case,

$$\kappa_2 = \max(0, K - 7H_{\text{HDPC}}).$$

Thus the bound in Lemma 12 is exact for the RFC 6330 parameter set.

Appendix A gives stronger results from the larger subfield subcodes of the intermediate code. Those results include attacks in parameter regimes where $\kappa_2 = 0$, together with full-domain random-ESI experiments, canonical-prefix experiments, and matched random-code comparisons over $\mathbb{F}_4$, $\mathbb{F}_{16}$, and $\mathbb{F}_{256}$.

D Probability spaces and decoding guarantees

Byzantine erasure security is defined relative to a challenger $\mathcal{C}$ and concerns adversarial subset selection after the realized H is revealed, whereas Raptor-code decoding guarantees are often stated probabilistically. A decoding-failure probability is therefore incomplete until its random variables and quantifier order are named. Several distinct probability spaces arise in the Raptor-code literature and in the protocol considered here:

- (1) *code or ensemble randomness*, in which encoding equations, a precode, coefficients, or an entire code are sampled;
- (2) *random-ESI selection*, in which encoding-symbol identifiers are sampled from a fixed standardized code;
- (3) *random erasures or received subsets*, in which a fixed stream or finite collection of encoding symbols is subjected to a random loss or reception process;
- (4) *challenger randomness*, in which $\mathcal{C}$ randomizes its legal choices of E or H ; and
- (5) *attack randomness*, in which the adversary uses internal randomness to search for a bad subset of a fixed realized instance.

Random-ESI selection can itself be part of challenger randomness; what matters is which experiment the probability belongs to and where it appears in the order of choices. These experiments are not interchangeable. For a fully specified deterministic code and a fixed received set, decoding success is deterministic: the relevant equations determine the K input symbols or they do not.

For standardized R10 and RaptorQ, the encoding relationships are fixed once the encoding parameters and encoding-symbol identifiers are fixed [4, 5]. Thus any decoding-failure probability for a concrete standardized instance comes from an additional random experiment.

RFC 6330 is explicit about one such experiment for RaptorQ. Its mandatory recovery properties concern K' , $K' + 1$, or $K' + 2$ encoding symbols whose ESIs are chosen independently and uniformly at random from the range of possible ESIs (RFC 6330, Section 5.8). These are random-ESI guarantees. They do not assert that every set of the stated size decodes, and they do not cover a set selected after the encoding equations are known. The Lopes–Neves attack exploits exactly this distinction for RaptorQ [9].

Random erasures give a different experiment. Shokrollahi and Luby report empirical overhead–failure curves for the concrete R10 and RQ constructions by scanning the possible ESIs and independently dropping each ESI with a specified loss probability until enough symbols survive [7, Appendix B]. This tests ordinary independent losses for a fixed deterministic construction; it is distinct from sampling ESIs uniformly from the full identifier space.

Some analyses use code-ensemble randomness instead. At a high level, a Raptor encoder first applies a precode to the source symbols to obtain intermediate symbols, and then uses a Luby transform (LT) to form encoding symbols from the intermediate symbols. In our row convention, each LT encoding row specifies a sparse linear combination of the intermediate symbols that produces one encoding symbol. In probabilistic LT models, these rows are sampled according to a degree distribution. In standardized R10, by contrast, the encoding parameters and ESI deterministically specify the corresponding row [4]. For our R10 attack, this internal LT structure matters only in establishing the fixed binary input-to-encoding map; the attack thereafter treats the resulting binary matrix as arbitrary. Classical LT and Raptor analyses sample such encoding rows according to a degree distribution, so the decoding probability is taken over the resulting random construction [6, 7]. For example, Lázaro, Liva, Paolini, and Bauch analyze a random Raptor-code model in which each LT encoding row is generated randomly according to a prescribed degree distribution: the degree is sampled, the participating intermediate-symbol indices are selected uniformly, and, over nonbinary fields, the nonzero coefficients are sampled as well [21]. Their decoding-failure probability therefore averages over random LT construction, even when the outer code is fixed. They also consider ensembles in which the outer code is random. Their numerical experiments use the LT degree distribution employed by standard R10, but the random LT ensemble is not the deterministic identifier-to-row mapping of RFC 5053 [21, 4]. Consequently, those bounds do not state a failure probability for one fixed RFC 5053 instance and one specified received set.

Lázaro et al. observe that their results do not depend on the channel erasure statistics [21]. That robustness concerns erasures that remain independent of the realized random LT encoding rows used in the proof. It does not justify conditioning on the realized rows and then choosing which ones survive. After that conditioning and adaptive selection, the survivor distribution is no longer the distribution analyzed by the ensemble argument. The game-level distinction can be written directly. For a fixed realized set H , let A_H be the corresponding row matrix, and for $S \subseteq H$ let A_S denote the rows corresponding to S . A random-subset statement has the form

$$\Pr_S[\text{rank}(A_S) = K] \geq 1 - \varepsilon,$$

where S is sampled according to the specified reception or erasure model. In the Byzantine erasure game, by contrast, H is revealed before the adversary chooses S . A sufficient information-theoretic condition is

$$\forall S \subseteq H, |S| = f + 1 : \text{rank}(A_S) = K,$$

but Byzantine erasure security against $\mathcal{C}$ only requires that it be computationally infeasible for an adversary to find a rank-deficient S in the game against $\mathcal{C}$. A small average failure probability for a random S establishes neither statement.

Under the dimension conditions of Theorems 6 and 13, every realized set H contains a bad subset. Under the stronger conditions of Theorems 11 and 15, a randomized adversary finds one efficiently in every fixed H . Thus R10 and RaptorQ fail Byzantine erasure security against every legal challenger in those regimes. The randomized attacks introduce a different probability, namely the adversary’s internal randomness in searching a fixed realized matrix for a bad subset. By contrast, the full-domain RaptorQ experiments in Appendix A use a particular randomized challenger that samples E uniformly and, after m is fixed, samples H uniformly from E ; the experiment samples H directly because its marginal distribution is uniform over subsets of the full ESI domain.

E Implementation and performance evaluation

The security results make an MDS code such as Reed–Solomon a natural alternative to R10 and RaptorQ. A common historical objection is computational cost: older systems literature often describes Reed–Solomon encoding or decoding as practically quadratic and substantially slower than LT or Raptor coding [11, 10]. More recently, eAID notes that SIMD-optimized Reed–Solomon libraries can offer competitive practical throughput [12]. This appendix measures that cost at the one-symbol-per-party parameters used throughout the paper. The implementations described here are research implementations, not production code. The accompanying benchmark artifact [41] records the harness, verification checks, source data, and implementation provenance for these measurements.

E.1 Benchmark model and measurement protocol

This subsection fixes the parameters, machine, and timing protocol used for every measurement. For each tested f , we set

$$K = f + 1, \quad n = 3f + 1.$$

Let M be the nominal source-message size in bytes, chosen as 0.5, 1, 2, or 4 MiB. For each (f, M) , the symbol size is $\lceil M/K \rceil$ rounded up to a multiple of 64 bytes, so the actual source-message size is K times the rounded symbol size and is slightly larger than the nominal target in some rows.

The full grid contains 48 values of f and 192 parameter points. For the $n \leq 256$ region, where $\mathbb{F}_{2^8}$ supplies enough field elements for RS-8, we use

$$f \in \{31, 32, 33, 40, 48, 56, 63, 64, 65, 72, 80, 85\}.$$

For $n > 256$, we sample every sixth value from 86 through 254. We also include

$$f \in \{96, 127, 128, 129, 160, 192, 224, 255, 256\}.$$

The transform decoder’s work buffer rounds up at powers of two, so its cost jumps at $f = 2^j + 1$. The grid deliberately samples the triples $2^j - 1, 2^j, 2^j + 1$ around these jumps. RaptorQ is

unavailable at eight points where the derived symbol size exceeds its 16-bit transfer-symbol-size field.

All measurements use one core of an AMD Ryzen 9 7950X. The measured process is pinned to core 4, the core is fixed at 4.5 GHz with boost disabled, and all codec paths are single-threaded. Rust code is built with `-C target-cpu=native`, and the C driver with `-O3 -march=native`. Each process runs under `setarch -R` so that address-space layout is fixed across compared configurations. The environment blocks of compared benchmark legs are padded to the same byte length, so changing a configuration does not shift the initial process stack.

Each parameter point is measured in five independent fresh-process invocations. Each invocation performs three discarded warmup trials followed by 20 measured trials under its own deterministic seed. The headline time is the median of the five per-invocation medians. The harness also records the minimum and maximum of those five medians, as well as the pooled 90th percentile and maximum over all 100 measured trials. Every reconstruction is checked against the original source.

Encoding and decoding are measured in separate benchmark legs. In the encoding leg, the decoder runs once as a correctness gate before the measured trials; in the decoding leg, the encoder runs once during setup and every measured trial decodes that codeword under a fresh erasure pattern. This separation avoids cross-operation cache effects.

Per-call working and output allocations required by the codec are inside the timed region. The allocator is configured so that freed working buffers remain resident and can be recycled, rather than being unpredictably returned to the operating system by glibc’s dynamic `mmap` and `trim` thresholds. Work depending only on the code shape is built once and reused. Work depending on the erasure pattern is rebuilt for every message.

For encoding, R10 times encoder setup plus generation of all n non-systematic transmitted symbols. RaptorQ times per-message setup plus generation of the $n - K$ repair symbols, with its reusable K -specific encoding plan outside the steady-state timer. The systematic Reed–Solomon implementations time generation of the $n - K$ parity symbols. For decoding, all decoder construction and reconstruction work through materialization of the contiguous source message is inside the timer. R10 and RaptorQ receive symbols incrementally in the same seeded pseudorandom arrival order until decoding succeeds. The R10 harness follows the same incremental call sequence used by the implementation’s production integration: each arriving symbol is submitted to the persistent decoder, decoding is attempted after every arrival, and source reconstruction is performed immediately on success. The MDS implementations receive exactly the first K symbols in that order and reconstruct in batch.

E.2 Implementations

We benchmark four primary implementations: R10, RaptorQ, RS–8, and RS–16. Two external Reed–Solomon implementations serve as descriptive references.

R10 and RaptorQ. R10 is the non-systematic RFC 5053-derived `monad-raptor` implementation from Category Labs’ `monad-bft` repository [34]. RaptorQ is `raptorq` 2.0.1 [33] implementing RFC 6330, with its reusable source-block encoding plan excluded from the per-message encoding timer. The RaptorQ build enables its optimized SIMD paths, including AVX-512 on the benchmark platform. Neither code is MDS, so the decoder may consume more than K symbols. The R10 decoder is used without modification. Depending on the arrival order, its normal execution may combine belief-propagation peeling with inactivation machinery. Instrumentation confirms that this machinery is entered in most trials on the main grid, but its dense Gaussian-elimination

subroutine runs only in a minority of trials and only on small systems. Thus the reported R10 decode times measure the implementation’s normal end-to-end decoding strategy rather than a peeling-only path; the measurements do not establish which component of the inactivation machinery dominates its cost.

RS–8. For $n \leq 256$, we benchmark a systematic Reed–Solomon code over $\mathbb{F}_{2^8}$ using the construction described in Appendix F. Its additive-FFT encoder independently implements and generalizes the construction used by Firedancer’s `reedsol` [35]. The decoder uses the closed-form Cauchy matrix decoder described in Appendix F. The implementation uses substantially modified AVX-512 vector kernels from Intel’s ISA-L library [36]. We call this implementation RS–8.

RS–16. To span the full grid, we benchmark a systematic Reed–Solomon code over the composite field $\mathbb{F}_{2^{16}}$ described in Appendix F.5, using the Cantor-node construction described in Appendix F. The additive-FFT encoder and transform decoder began as a port of `reed-solomon-simd` 3.1.0 and were subsequently modified substantially [37].

The same encoded shards can be decoded in two ways. The matrix decoder uses the closed-form Cauchy plan. The transform decoder uses an additive-FFT erasure decoder. Under the pseudorandom-arrival benchmark, the matrix decoder is generally faster below $f \approx 220$ and the transform decoder above it. The crossover lies between $f = 200$ and 230 depending on message size, and the two methods may therefore be dispatched using the code shape and the realized erasure pattern. The transform has power-of-two work-buffer jumps at $f = 2^j + 1$; within the paper’s grid these lie below the crossover, while beyond the grid they can temporarily make the matrix decoder preferable again. We call the resulting implementation RS–16.

Before timing, both decoder paths were verified at every grid point and checked against the same encoded shards.

Firedancer reference. As an external $\mathbb{F}_{2^8}$ reference, we also benchmark Firedancer’s `reedsol` implementation at the pinned revision [35]. The encoder is regenerated with only Firedancer’s policy size ceilings raised so that the same construction can cover the paper’s $n \leq 256$ shapes. For decoding, a benchmark adapter applies the same reconstruction algorithm under our K -of- n reconstruction model and omits parity reconstruction and corruption checking, which are not charged to the other codecs.

reed-solomon-simd reference. For reference, we also benchmark `reed-solomon-simd` 3.1.0 [37]. It is used only as a descriptive external reference and is omitted from the paper tables.

E.3 Selected absolute timings

Table 8 gives representative absolute timings. RS–8 is available only for $n \leq 256$; dashes in that column above the boundary indicate that the field is too small for the code length. RS–16 uses the additive-FFT encoder in every row. For the displayed rows through $f = 129$ it uses the matrix decoder, and at $f = 256$ it uses the transform decoder. The other decoder can consume the same codewords. The selected rows include representative small party counts, the $n = 256$ field boundary, a power-of-two transform transition, and the largest value of f in the grid. All displayed decode times use the seeded pseudorandom arrival order of Appendix E.1; the supplementary all-parity stress test is reported below.

f	n	MiB	R10 enc./dec.	RaptorQ enc./dec.	RS-8 enc./dec.	RS-16 enc./dec.
32	97	0.5	0.145 / 0.232	0.466 / 0.361	0.049 / 0.055	0.067 / 0.110
32	97	1	0.308 / 0.410	1.034 / 0.817	0.098 / 0.107	0.134 / 0.215
32	97	2	0.626 / 0.779	2.068 / 1.835	0.204 / 0.214	0.266 / 0.441
32	97	4	1.305 / 1.549	– / –	0.422 / 0.443	0.549 / 0.889
64	193	0.5	0.149 / 0.305	0.424 / 0.360	0.072 / 0.110	0.085 / 0.186
64	193	1	0.301 / 0.481	0.869 / 0.788	0.142 / 0.220	0.177 / 0.359
64	193	2	0.632 / 0.890	1.928 / 1.531	0.281 / 0.434	0.342 / 0.747
64	193	4	1.305 / 1.709	4.182 / 3.329	0.573 / 0.896	0.688 / 1.566
85	256	0.5	0.145 / 0.333	0.404 / 0.371	0.089 / 0.150	0.097 / 0.231
85	256	1	0.289 / 0.479	0.816 / 0.795	0.182 / 0.291	0.202 / 0.452
85	256	2	0.610 / 0.943	1.781 / 1.648	0.369 / 0.572	0.401 / 0.896
85	256	4	1.264 / 1.736	3.990 / 3.238	0.763 / 1.157	0.822 / 1.991
86	259	0.5	0.140 / 0.328	0.404 / 0.374	– / –	0.096 / 0.236
86	259	1	0.273 / 0.454	0.668 / 0.783	– / –	0.200 / 0.460
86	259	2	0.573 / 0.876	1.696 / 1.482	– / –	0.393 / 0.955
86	259	4	1.196 / 1.633	3.764 / 3.354	– / –	0.822 / 1.897
128	385	0.5	0.151 / 0.375	0.432 / 0.420	– / –	0.087 / 0.343
128	385	1	0.286 / 0.491	0.838 / 0.686	– / –	0.177 / 0.651
128	385	2	0.592 / 0.868	1.621 / 1.530	– / –	0.347 / 1.375
128	385	4	1.252 / 1.664	4.046 / 3.148	– / –	0.693 / 2.802
129	388	0.5	0.153 / 0.446	0.433 / 0.427	– / –	0.107 / 0.345
129	388	1	0.294 / 0.575	0.697 / 0.693	– / –	0.216 / 0.665
129	388	2	0.604 / 0.985	1.737 / 1.352	– / –	0.423 / 1.376
129	388	4	1.277 / 1.831	3.930 / 3.434	– / –	0.937 / 2.807
256	769	0.5	0.188 / 0.471	0.425 / 0.624	– / –	0.114 / 0.601
256	769	1	0.316 / 0.647	0.711 / 0.945	– / –	0.222 / 1.013
256	769	2	0.621 / 0.926	1.379 / 1.640	– / –	0.434 / 1.909
256	769	4	1.348 / 1.692	3.235 / 2.710	– / –	0.861 / 4.351

Table 8: Selected points from the single-threaded benchmark. Times are milliseconds, and each codec cell reports encoding time followed by decoding time. For each f , $K = f + 1$ and $n = 3f + 1$. The heavier horizontal rule marks the $n = 256$ boundary. All entries use the protocol and measurement protocol of Appendix E.1.

E.4 Full-grid comparisons

Table 9 summarizes the full grid. Each entry is the median timing ratio of the first-named implementation to the second, followed by the number of points at which the first is faster. A ratio below one therefore favors the first implementation. Encode and decode are kept separate throughout; we do not treat $M/(t_{\text{enc}} + t_{\text{dec}})$ as a throughput metric.

Region	comparison	encode	decode
$n \leq 256$	RS-8 / R10	0.464 (48/48)	0.411 (48/48)
$n \leq 256$	RS-8 / RaptorQ	0.151 (40/40)	0.264 (40/40)
$n \leq 256$	RS-8 / RS-16	0.794 (48/48)	0.572 (48/48)
$n > 256$	RS-16 / R10	0.619 (144/144)	1.439 (18/144)
$n > 256$	RS-16 / RaptorQ	0.236 (144/144)	0.959 (80/144)

Table 9: Full-grid median timing ratios and win counts. For RS-16, the emitted representation is fixed and the faster of its two decoders is used for each measured shape under pseudorandom arrivals above the boundary; below the boundary its matrix decoder is used. RaptorQ has only 40 points below the boundary because of its symbol-size limit.

For $n \leq 256$, RS-8 is faster than R10 and RaptorQ on both encoding and decoding at every common measured point. At the median it encodes about $2.2\times$ faster than R10 and $6.6\times$ faster than RaptorQ, and decodes about $2.4\times$ faster than R10 and $3.8\times$ faster than RaptorQ. Against RS-16 in this region, RS-8 decodes and encodes faster at all 48 points, by median factors of about 1.75 and 1.26, respectively. At the immediate field transition from $f = 85$, $n = 256$ to $f = 86$, $n = 259$, comparing equal message sizes, the ratio of the RS-16 encoding time to the RS-8 encoding time ranges from 1.06 to 1.10 across the four message sizes, while the corresponding decoding ratio ranges from 1.58 to 1.67.

As an external $\mathbb{F}_{2^8}$ reference, we compare RS-8 with Firedancer’s `reedsol` under the same measurement protocol. At the paper’s $n = 3f + 1$ shapes, RS-8 is faster on both terms at all 48 points, with median speedups of about $1.80\times$ on encoding and $5.95\times$ on decoding. At Firedancer’s native $K = 32$, $n = 64$ shape, its encoder is instead $1.25\times$ – $1.40\times$ faster than RS-8, so this comparison is workload-specific.

Above $n = 256$, RS-16 changes the comparison substantially. Its encoder is faster than R10 and RaptorQ at all 144 points, by median factors of about 1.6 and 4.2, respectively. Its decoder is about $1.44\times$ slower than R10 at the median, but about 4.1% faster than RaptorQ at the median. Against the `reed-solomon-simd` reference, the RS-16 transform encoder, using the interleaved byte representation defined in Appendix F.5, is faster at all 144 points, with median time ratio 0.464. Choosing the faster RS-16 decoder by shape under pseudorandom arrivals, RS-16 decodes faster at all 144 points, with median time ratio 0.425. The published crate does not provide AVX-512 or GFNI kernels, so we treat this comparison only as a descriptive external reference point.

Under pseudorandom arrivals, the matrix and transform decoders cross near $f \approx 220$ inside the measured grid. With this interleaved representation, the crossing lies between $f = 206$ and 212 at 0.5 and 1 MiB, between 200 and 206 at 2 MiB, and between 224 and 230 at 4 MiB. This is a receiver-local choice because both decoders consume the same emitted codewords. These power-of-two work-buffer jumps mean that the rule should be understood as a receiver-local dispatch rather than as an asymptotic monotonicity claim.

All-parity arrivals. As a supplementary stress test, we repeated the decoding benchmark on a subset of the main grid under an all-parity arrival pattern, in which the receiver obtains K parity symbols and no original source symbols. For the matrix decoders, this sets the number r of missing source symbols to its maximum K , rather than about $2K/3$ under pseudorandom arrivals. Relative to pseudorandom arrivals, the median decode-time ratio is 1.449 for RS-8 and 1.462 for the RS-16 matrix decoder, while the corresponding ratio for the RS-16 transform decoder is 1.007. Thus the preferred RS-16 decoder depends on both the code shape and the realized erasure pattern, and the receiver can choose between the two after observing its K received symbols. This stress test does not preserve the slight RS-16 decode advantage over RaptorQ seen under pseudorandom arrivals: at many of the measured points above $n = 256$, RaptorQ is faster, while Reed-Solomon retains its deterministic K -of- n reconstruction guarantee.

Beyond the main grid. To check that the in-grid decoder crossover persists at larger party counts, we ran a separate RS-16 sweep through $f = 2047$. It used the same machine, core pinning, fixed address-space layout, allocator configuration, and five fresh-process invocations per point as the main grid, with the headline time again taken as the median of the five invocation medians; each invocation used 15 trials. Table 10 reports representative 1 MiB results for RS-16.

f	n	matrix decode	transform decode	matrix / transform
511	1534	2.786	1.196	2.33×
1023	3070	7.129	1.145	6.23×
2047	6142	24.925	1.403	17.76×

Table 10: RS-16 decoding beyond the main grid for a 1 MiB source-message target. Times are milliseconds. Both decoders consume the same interleaved codewords; the final column is matrix-decoder time divided by transform-decoder time.

At these fixed-message-size points, the transform decoder’s advantage widens from $2.33\times$ at $f = 511$ to about $18\times$ at $f = 2047$. Thus the asymptotic advantage of the transform decoder is visible in the measurements rather than only in its $O(K \log K)$ complexity. The power-of-two work-buffer jumps remain visible beyond the grid: at $f = 257$ the transform decoder is about $1.34\times$ slower than the matrix decoder at 1 MiB and $2.11\times$ slower at 4 MiB. By $f = 513$ it is $1.33\times$ faster at 1 MiB, but remains about $1.08\times$ slower at 4 MiB. This reinforces the use of receiver-local dispatch rather than a single monotone threshold.

E.5 Received-symbol consumption

The CPU timings do not include the additional receive volume required when a fountain decoder needs more than K symbols. Table 11 reports this separately. Every MDS implementation consumes exactly K symbols.

Thus R10 needs about 27.5% more symbols than an MDS code at the median across the grid, and as much as 42.9% more in the worst measured trial. Accordingly, the R10-versus-MDS decode timings are implementation-level CPU comparisons under each codec’s normal input requirements, not comparisons at equal received traffic. RaptorQ is effectively MDS at these tested shapes at the median, but it is not MDS as a coding guarantee and reaches $1.0312K$ in the worst trial.

codec	median pointwise median	largest pointwise median	worst trial
MDS Reed–Solomon	$1.0000K$	$1.0000K$	$1.0000K$
RaptorQ	$1.0000K$	$1.0000K$	$1.0312K$
R10	$1.2752K$	$1.3438K$	$1.4286K$

Table 11: Received symbols required for successful reconstruction. The first column is the median, over grid points, of each point’s median consumption.

E.6 Absolute source throughput

We report throughput in *source-message* bits per second. If B is the rounded symbol size, then the reported rate is

$$R_{\text{src}} = \frac{8KB}{t}.$$

This normalization has the same meaning across systematic and non-systematic encoders and across all decoders. For a systematic encoder, parity-output throughput would instead use $8(n - K)B/t$, and full-codeword volume would use $8nB/t$. At the parameters here those are approximately $2R_{\text{src}}$ and $3R_{\text{src}}$, so calling either one simply “encoding throughput” would obscure what was counted. For MDS decoding, the decoder reads exactly K coded symbols, so its coded-input byte rate equals its source-message byte rate. The extra input volume for R10 and RaptorQ is instead reported explicitly in Table 11.

Taking RS–8 for $n \leq 256$ and RS–16 above the boundary, and allowing RS–16 to choose its matrix or transform decoder by shape under pseudorandom arrivals, the worst measured MDS source-throughput floors over all 192 main-grid points are as follows.

operation	worst measured source throughput	grid point
encode	31.9 Gbit/s	$f = 134$, 4 MiB
decode	7.00 Gbit/s	$f = 256$, 0.5 MiB

Table 12: Worst measured single-threaded MDS source-throughput floors on the main pseudorandom-arrival grid. The underlying rates are approximately 3804 MiB/s for encoding and 835 MiB/s for decoding, converted to decimal Gbit/s for comparison with line rates.

These are single-core measurements at a fixed 4.5 GHz, below the processor’s peak clock. The decoding floor therefore remains at least 7.0 Gbit/s of source data even at the worst measured shape under pseudorandom arrivals. For encoding, the roughly one-third code rate means that a 31.9 Gbit/s source rate corresponds to roughly three times that rate in complete systematic codeword volume. We do not report $M/(t_{\text{enc}} + t_{\text{dec}})$ as throughput: the sum is a combined serial coding time, not in general a steady-state rate because the two operations occur on different parties and may overlap or pipeline.

E.7 Protocol-level interpretation

The relative importance of encoding and decoding depends on the surrounding protocol. In an asymmetric dispersal, one designated sender encodes a block while each eventual recipient decodes it. Encode and decode latency are both relevant to an individual sender-to-recipient path, but they are paid by different parties and should not be collapsed into a fictitious common throughput.

In a symmetric round or batch in which every party originates one block, each party performs roughly one encode and $n - 1$ decodes, so decoding receives much greater aggregate weight.

The measurements therefore support two conclusions rather than a universal CPU ranking. First, the deterministic K -of- n guarantee of an MDS code can be obtained at multi-gigabit single-core rates over the measured range. Second, the relative CPU premium depends on the operation and shape. At $n \leq 256$, RS-8 is faster than R10 and RaptorQ on both encoding and decoding throughout the common measured grid under pseudorandom arrivals. Above the 256-symbol boundary, R10 retains a decode advantage, while RS-16 has a much faster encoder and consumes exactly the information-theoretic minimum number of received symbols. RaptorQ is close to MDS in measured symbol consumption, and under pseudorandom arrivals RS-16 is approximately level with or modestly faster than it on decode above the boundary while substantially faster on encode. The all-parity stress test above shows that the decode-time comparison is erasure-pattern dependent.

F Reed–Solomon constructions and decoding algorithms

This appendix gives the algebraic details underlying the experimental Reed–Solomon implementations of Appendix E. The main points are that systematic evaluation Reed–Solomon has a generalized Cauchy generator, that a subspace choice of evaluation nodes makes that generator Cauchy up to one scalar per coset, and that shortening and principal pivoting preserve the generalized Cauchy form for arbitrary K . These constructions admit additive-FFT encoding together with closed-form Cauchy decoding.

Throughout this appendix, the field has characteristic two, so field addition is denoted by $\oplus$.

F.1 Systematic evaluation Reed–Solomon as a generalized Cauchy code

Fix $K + m$ distinct field elements, with data nodes $x_0, \dots, x_{K-1}$ and parity nodes $y_0, \dots, y_{m-1}$. A message $d_0, \dots, d_{K-1}$ determines the unique polynomial P of degree less than K satisfying $P(x_i) = d_i$. The systematic Reed–Solomon codeword consists of the evaluations of P at all data and parity nodes.

The Lagrange basis polynomials are

$$L_i(z) = \frac{\prod_{\ell \neq i} (z \oplus x_\ell)}{\prod_{\ell \neq i} (x_i \oplus x_\ell)}.$$

Hence the parity generator is

$$G_{j,i} = L_i(y_j) = \frac{\prod_{\ell \neq i} (y_j \oplus x_\ell)}{\prod_{\ell \neq i} (x_i \oplus x_\ell)}.$$

Define

$$a_j = \prod_{\ell=0}^{K-1} (y_j \oplus x_\ell), \quad w_i = \prod_{\ell \neq i} (x_i \oplus x_\ell).$$

Then

$$G_{j,i} = \frac{a_j}{(y_j \oplus x_i)w_i},$$

so

$$G = \text{diag}(a_j) C \text{diag}(w_i^{-1}), \quad C_{j,i} = \frac{1}{y_j \oplus x_i}.$$

All diagonal factors are nonzero because the evaluation nodes are distinct. Thus every systematic evaluation Reed–Solomon code has a generalized Cauchy parity generator. The bare Cauchy matrix C generally defines a different systematic code on the same nodes: it omits the nonzero row and column scalings above and need not emit the same parity symbols. The two parity generators are related by invertible row and column scalings, which preserve nonsingularity of every square submatrix and hence the MDS property.

F.2 Subspace nodes, shortening, and pivoting

For the transform descriptions below, fix a binary basis $\beta_0, \dots, \beta_{q-1}$ of $\mathbb{F}_{2^q}$ and define

$$\phi(i) = \bigoplus_{b: i_b=1} \beta_b,$$

where i_b is bit b of the integer i . Then integer XOR on indices corresponds to field addition under ϕ , so the indices $0, \dots, 2^r - 1$ enumerate the subspace

$$V_r = \text{span}_{\mathbb{F}_2}(\beta_0, \dots, \beta_{r-1}),$$

and every aligned block of 2^r indices enumerates a coset of V_r .

Suppose first that the data nodes are exactly a binary subspace V of size $K_0 = 2^r$. For any $x_i \in V$, translation by x_i permutes V , so

$$w_i = \prod_{v \in V \setminus \{x_i\}} (x_i \oplus v) = \prod_{u \in V \setminus \{0\}} u =: w$$

is independent of i .

Define the subspace polynomial

$$s_V(z) = \prod_{v \in V} (z \oplus v).$$

If $v_0 \in V$, translation by v_0 permutes the factors, and therefore

$$s_V(z \oplus v_0) = s_V(z).$$

Thus s_V is constant on each coset of V . If parity node y_j lies in coset t with representative c_t , then

$$a_j = s_V(y_j) = s_V(c_t).$$

Consequently the parity generator has the form

$$G_{j,i} = \frac{\gamma_t}{y_j \oplus x_i},$$

where $\gamma_t = s_V(c_t)/w$ depends only on the coset containing y_j . The generator is therefore a pure Cauchy matrix up to one scalar per parity coset.

The power-of-two assumption can be removed by shortening. For arbitrary K , let

$$K_0 = 2^{\lceil \log_2 K \rceil},$$

and choose a subspace V of size K_0 . Start from the systematic $[K_0 + m, K_0]$ evaluation code whose data nodes are V . Fix the last $K_0 - K$ source coordinates to zero and delete them. Shortening an MDS code is MDS, so the result is a $[K + m, K]$ MDS code. Its parity generator is simply the first K columns of the same Cauchy matrix:

$$G_{j,i} = \frac{\gamma_t}{y_j \oplus x_i}, \quad 0 \leq i < K.$$

An additive FFT [27] computes this zero-padded shortening by evaluating over all K_0 data-node positions. It therefore needs

$$K_0 + m = 2^{\lceil \log_2 K \rceil} + m$$

distinct field points. We call the cosets of the data subspace V that contain parity nodes **recovery cosets**; at the paper's rate at most two are needed. RS-16 uses this endpoint's nodes in the larger field, with the implementation's Cantor basis, satisfying $\beta_0 = 1$ and $\beta_i^2 \oplus \beta_i = \beta_{i-1}$ for $i \geq 1$, to enumerate the data subspace and its recovery cosets. It divides the parity in recovery coset t by γ_t before emission, so its emitted parity generator is the bare Cauchy matrix

$$C_{j,i} = \frac{1}{y_j \oplus x_i}.$$

The collapse above uses only the subspace property; the basis choice is not part of the proof. The Cantor convention provides the recursion used by the additive FFT.

RS-8 instead chooses an intermediate pivot base K_p rather than fixing $K_p = K_0$. Principal-pivot transforms provide the basic mechanism [28], and related prefix-pivot constructions for arbitrary Reed-Solomon parameters appear in [29]. The corresponding shortening is encoded efficiently without spending all $K_0 - K$ intervening field points on zeros. Identify the elements of $\mathbb{F}_{2^8}$ with their byte indices, so field addition on these indices is XOR. Choose a pivot base K_p satisfying

$$K \leq K_p \leq K_0, \quad K_p + m \leq 256.$$

Interpolate the degree- $< K_p$ polynomial Q through the K data values at nodes $0, \dots, K-1$ and zeros at nodes $K, \dots, K_p-1$, then place the m parity symbols at nodes $K_p, \dots, K_p+m-1$. Define

$$Z(z) = \prod_{\ell=K}^{K_p-1} (z \oplus \ell).$$

Because Q vanishes at the shortened coordinates, write $Q = Zg$ with $\deg g < K$. If L_i is the Lagrange basis polynomial for the K data nodes alone, then the systematic parity generator is

$$G_{j,i} = \frac{Z(K_p + j)}{Z(i)} L_i(K_p + j).$$

Equivalently, with

$$A_j = \prod_{\ell=0}^{K_p-1} (K_p + j \oplus \ell), \quad W_i = \prod_{\substack{0 \leq \ell < K_p \\ \ell \neq i}} (i \oplus \ell),$$

we have

$$G_{j,i} = \frac{A_j}{(K_p + j \oplus i)W_i},$$

and hence

$$G = \text{diag}(A_j) C \text{diag}(W_i^{-1}), \quad C_{j,i} = \frac{1}{K_p + j \oplus i}.$$

Thus every admissible K_p gives a $[K + m, K]$ MDS generalized Reed–Solomon code with a generalized Cauchy parity generator. Different admissible values of K_p generally define different systematic codes, while preserving the same generalized Cauchy decoding structure. The endpoint $K_p = K_0$ is zero-padding, while $K_p = K$ is the full principal-pivot construction. Intermediate values trade pivot work against evaluation points. Since $K_p = K$ is feasible whenever $K + m \leq 256$, RS-8 reaches the entire $n \leq 256$ region. The implementation uses the deterministic largest-feasible pivot base

$$K_p = \min\{K_0, 256 - m\}.$$

Thus it uses the zero-padded endpoint whenever the field budget permits, and otherwise uses the largest admissible intermediate base. The choice depends only on the code shape and not on machine-specific timing.

F.3 Identity-row reduction

In a systematic code, the source rows of the generator are identity rows, so received source symbols require no inversion; only the missing source symbols must be solved for. Consider a systematic decoder that has received exactly K symbols. Suppose r source symbols are missing, so the receiver holds $K - r$ original source symbols and r parity symbols. Let d_O be the received originals, d_M the missing originals, and p the received parity symbols. After partitioning the parity rows according to these source coordinates, write

$$p = Ed_O \oplus Bd_M.$$

The residual matrix B is $r \times r$. Hence

$$d_M = B^{-1}p \oplus (B^{-1}E)d_O.$$

Only B must be inverted. This replaces inversion of the full $K \times K$ survivor matrix by an $r \times r$ residual problem and makes the Cauchy structure directly visible.

For a generalized Cauchy generator

$$G = \text{diag}(a_j) C \text{diag}(w_i^{-1}),$$

let D_a denote the diagonal row scalings on the received parity rows, and let D_M and D_O denote the diagonal w_i scalings on the missing and received source coordinates. Then the residual blocks have the form

$$B = D_a C_B D_M^{-1}, \quad E = D_a C_E D_O^{-1},$$

so

$$B^{-1} = D_M C_B^{-1} D_a^{-1}, \quad B^{-1}E = D_M C_B^{-1} C_E D_O^{-1}.$$

The parity-row factor D_a therefore cancels from $B^{-1}E$, the product we call the **fold**. The remaining diagonal factors only rescale the received parity, received-source, and recovered-source coordinates, and are folded into the coefficient vectors already materialized by the decoder. The residual inversion and fold below therefore apply unchanged to the underlying Cauchy blocks.

For a Cauchy parity generator, let $x_0, \dots, x_{r-1}$ be the received parity nodes and $y_0, \dots, y_{r-1}$ the missing source nodes. Then

$$B_{a,b} = \frac{1}{x_a \oplus y_b}.$$

Every square submatrix of a Cauchy matrix is again Cauchy and nonsingular, so this residual B is invertible for every survivor set. Define

$$n_x(a) = \frac{\prod_{\ell} (x_a \oplus y_{\ell})}{\prod_{\ell \neq a} (x_a \oplus x_{\ell})}, \quad n_y(b) = \frac{\prod_{\ell} (x_{\ell} \oplus y_b)}{\prod_{\ell \neq b} (y_b \oplus y_{\ell})}.$$

The inverse is

$$(B^{-1})_{b,a} = \frac{n_x(a)n_y(b)}{x_a \oplus y_b}.$$

The product families can be precomputed in $O(r^2)$ field operations, after which every inverse entry costs $O(1)$.

F.4 Closed-form fold

The inverse alone is not the whole decode plan. Let u_c be the node of received original c . The matrix multiplying the received originals is

$$E_{a,c} = \frac{1}{x_a \oplus u_c},$$

so a direct computation of $B^{-1}E$ costs $O(r^2(K-r))$ field operations. The fold also has a closed form.

In characteristic two,

$$\frac{1}{(x \oplus y)(x \oplus u)} = \frac{\frac{1}{x \oplus y} \oplus \frac{1}{x \oplus u}}{y \oplus u}.$$

Define

$$P(w) = \frac{\prod_{\ell} (w \oplus y_{\ell})}{\prod_{\ell} (w \oplus x_{\ell})}, \quad S(w) = \sum_a \frac{n_x(a)}{x_a \oplus w}.$$

The partial-fraction expansion gives

$$S(w) = 1 \oplus P(w).$$

Because $P(y_b) = 0$, we have $S(y_b) = 1$. Substituting the preceding identity into the inner product for $B^{-1}E$ therefore yields

$$(B^{-1}E)_{b,c} = \frac{n_y(b)P(u_c)}{y_b \oplus u_c}.$$

For each received original u_c , $P(u_c)$ costs $O(r)$ operations and is shared across all rows b . The fold therefore costs $O(r(K-r))$ rather than $O(r^2(K-r))$, and the complete plan costs

$$O(r^2 + r(K-r)) = O(K^2).$$

The resulting $r \times K$ coefficient matrix is then applied to the K received shards in one pass.

F.5 Composite $\mathbb{F}_{2^{16}}$ arithmetic

The 256-element field cannot supply enough distinct evaluation nodes above $n = 256$. The experimental ISA-L-based implementation represents $\mathbb{F}_{2^8}$ in the polynomial basis used by ISA-L, modulo $X^8 + X^4 + X^3 + X^2 + 1$ (0x11d), and uses

$$\mathbb{F}_{2^{16}} = \mathbb{F}_{2^8}[z]/(z^2 + z + \beta), \quad \beta = 0\mathbf{x}20.$$

In this representation, β has absolute trace one, so $z^2 + z + \beta$ is irreducible.

Write a field element as $a_0 + a_1z$. Multiplication by a fixed element $a_0 + a_1z$ acts on $v_0 + v_1z$ as

$$(a_0 + a_1z)(v_0 + v_1z) = (a_0v_0 \oplus \beta a_1v_1) + (a_1v_0 \oplus (a_0 \oplus a_1)v_1)z.$$

Equivalently, it is the 2×2 matrix over $\mathbb{F}_{2^8}$

$$\begin{pmatrix} a_0 & \beta a_1 \\ a_1 & a_0 \oplus a_1 \end{pmatrix}.$$

Consequently an $r \times K$ coefficient matrix over $\mathbb{F}_{2^{16}}$ can be expanded into a $2r \times 2K$ coefficient matrix over $\mathbb{F}_{2^8}$ and passed to ISA-L's existing coding-kernel interface.

Write the coefficient matrix and input vector as

$$A = A_0 + zA_1, \quad v = v_0 + zv_1,$$

with A_0, A_1 and v_0, v_1 over $\mathbb{F}_{2^8}$. The implementation reduces four base-field products to three by Karatsuba. With

$$P_0 = A_0v_0, \quad P_1 = (\beta A_1)v_1, \quad P_2 = (A_0 \oplus A_1)(v_0 \oplus v_1),$$

the low output plane is $P_0 \oplus P_1$ and the high plane is $P_2 \oplus P_0$. The final benchmark uses an interleaved field-element byte representation for emitted shards. Within each 64-byte block, 32 low bytes are followed by the corresponding 32 high bytes, representing 32 consecutive $\mathbb{F}_{2^{16}}$ elements. The matrix decoder converts the K received shards to the contiguous planes required by ISA-L inside the decode timer; the transform decoder works directly with the transform representation.

F.6 Additive-FFT encoding and decoding

For RS-8, the additive FFT computes the pivoted shortening of Appendix F.2. For this K_p , it interpolates through the K source values and the $K_p - K$ prescribed zeros, then evaluates at the parity nodes $K_p, \dots, K_p + m - 1$. The direct generalized Cauchy matrix

$$G_{j,i} = \frac{A_j}{(K_p + j \oplus i)W_i}$$

computes the same parity symbols and provides an independent algebraic form for validation. RS-8 uses the closed-form Cauchy decoder rather than a transform decoder. The row and column scalings are folded into the coefficient vectors already materialized by the decoder, so the pivot choice does not change the asymptotic decode work.

For RS-16, we return to the source-node x_i and parity-node y_j notation of Appendix F.2. The Cantor-node convention and zero-padded shortening are fixed at encoding time. The additive-FFT encoder emits the shortened code whose parity generator is

$$C_{j,i} = \gamma_t^{-1} G_{j,i} = \frac{1}{y_j \oplus x_i}.$$

The transform path zero-pads the shortened source coordinates, evaluates the corresponding degree-bounded interpolation polynomial P over the data subspace and recovery cosets, and emits $\gamma_t^{-1}P(y_j)$ for a parity node in recovery coset t . For each later recovery coset evaluated by the transform, the factor is applied when the interpolation coefficients are copied to that coset. When the implementation handles a small final recovery coset by direct matrix multiplication, the same C rows are applied directly. A separate, non-benchmarked matrix encoder also applies C directly and serves as an independent validation oracle; after both outputs are expressed with all low bytes followed by all high bytes, they are required to be byte-identical.

The RS-16 additive-FFT decoder implements the same interpolation map by an erasure-locator transform [30]. Let V_N be its power-of-two shard-transform evaluation domain, with

$$|V_N| = N \geq K_0 + m.$$

In the benchmarked K -of- n reconstruction, the decoder knows the K received transmitted values and the $K_0 - K$ prescribed shortening zeros. Let E contain the remaining positions in V_N , with nodes ω_i , and define the erasure locator

$$\Lambda(z) = \prod_{i \in E} (z \oplus \omega_i).$$

Then $|E| \leq N - K_0$. If P is the interpolation polynomial of the source values and shortening zeros and $H = P\Lambda$, then $\deg P < K_0$ and

$$\deg H \leq (K_0 - 1) + |E| \leq N - 1.$$

Moreover, H is known at every node of V_N : at a prescribed shortening position it is zero, at a received source position ω_j it is the received symbol times $\Lambda(\omega_j)$, at a received parity position in recovery coset t it is the received symbol times $\gamma_t \Lambda(\omega_j)$, which undoes the encoder's normalization in the same multiplication, and at a position in E it is zero. These N values therefore determine H uniquely. Since the nodes in E are distinct,

$$\Lambda'(\omega_i) = \prod_{j \in E \setminus \{i\}} (\omega_i \oplus \omega_j) \neq 0.$$

After interpolating H , formal differentiation gives, for $i \in E$,

$$P(\omega_i) = \frac{H'(\omega_i)}{\Lambda'(\omega_i)}.$$

The implementation computes the locator with Didier's additive/Walsh-transform method over the full 2^{16} -element field. On V_N , the additional product over field positions outside V_N is a nonzero constant because translation by any element of V_N permutes its complement, and this factor cancels from the derivative quotient at erased positions. The implementation then applies the inverse-transform/derivative/forward-transform sequence realizing the identity above. The shard-transform work has transform-like $O(K \log K)$ scaling at the paper's fixed rate, but the locator computation still spans all 2^{16} field positions and contributes a fixed per-decode setup cost. The closed-form Cauchy decoder instead has a quadratic plan and a dense rK shard-application step. It has smaller constants at the smaller values of f measured here, which produces the crossover reported in Appendix E.4.

The two RS-16 decoding paths share no decode-plan code and are checked against the same emitted codewords and survivor sets. Thus RS-16 uses one shortened Cantor-node MDS code with one fixed emitted representation, one additive-FFT encoder, and two interchangeable decoders chosen after the shards have been received.