Bridging Shannon and Hamming:
Codes for computationally simple channels

Venkatesan Guruswami

Carnegie Mellon University

Based on joint work with Adam D. Smith (Penn State)

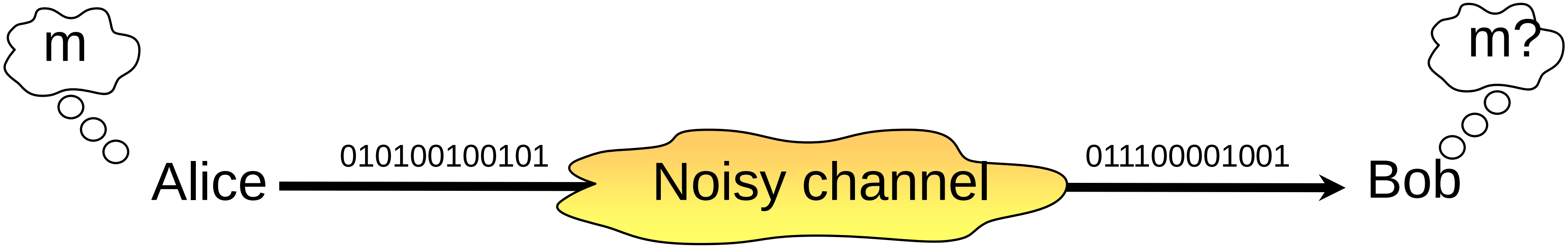
-- 3rd EaGL Theory Day ---

October 9, 2010

Outline

- Background & context
  - Error models, Shannon & Hamming
  - List decoding
- Computationally bounded channels
  - Previous results (with "setup")
- Our results
  - Explicit optimal rate codes (for two simple channels)
- Proof tools & ideas

Two classic channel models

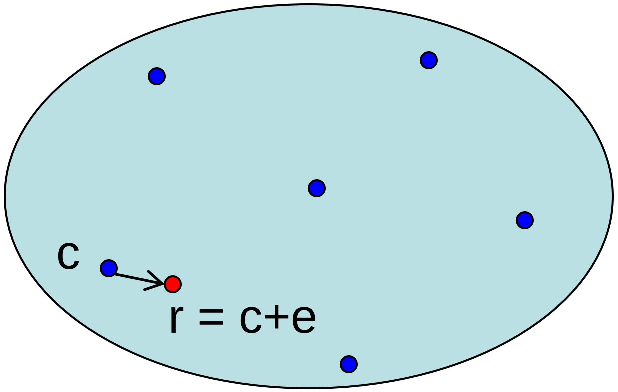


- Alice sends n bits
- ***Shannon***: Binary symmetric channel BSC_p
  - Flips each bit independently with probability p (error binomially distributed)
- ***Hamming***: Worst-case (adversarial) errors ADV_p
  - Channel outputs arbitrary word within distance pn of input

Best possible “rate” of reliable information transmission?

How many bits can we communicate by sending n bits on channel?

Error-correcting codes



Codewords well-separated

<u>(Binary) code:</u>

encoding $C : \{0,1\}^k \rightarrow \{0,1\}^n$

- $c = C(m)$
  - m = *message*
  - c = *codeword*

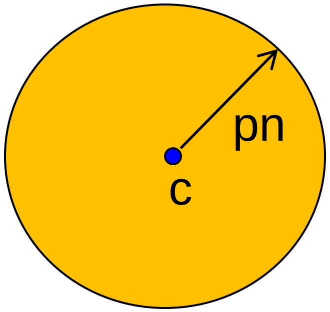
Rate $R = k/n$

- information per bit of codeword
- Want $R > 0$ as $k, n \rightarrow \infty$

<u>Idea/hope</u>: *codeword* $c \in C$ can be determined (efficiently) from noisy version $r = c + e$

- e unknown *error* vector obeying some "noise model"

Shannon capacity limit



Suppose pn bits can get flipped,

$p \in [0,1/2)$ ***error fraction***

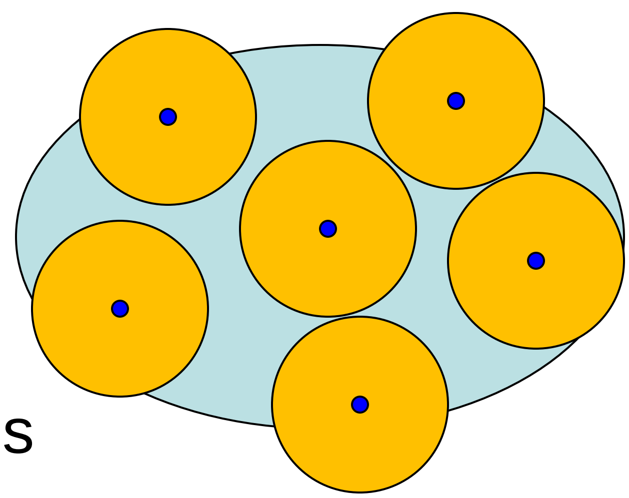
- $c \rightarrow r = c + e, \quad wt(e) \leq pn$

Decoding region for $c \in C$ has volume $\approx 2^{h(p)n}$

- $h(p) = -p \log_2 p - (1-p) \log_2 (1-p)$, binary entropy function

$\approx$ Disjoint decoding regions

- # codewords $\leq 2^n / 2^{h(p)n}$
- Rate $\leq 1 - h(p)$



Good codes $\Leftrightarrow$ Good sphere packings

Shannon's theorem

<u>Theorem</u>: There *exists* a code $C : \{0,1\}^{Rn} \rightarrow \{0,1\}^n$ of rate $R = 1-h(p)-\varepsilon$ such that $\forall m$, for $e \in_R$ *Binom(n,p)*

$$\Pr[\, C(m)+e \in \cup_{m' \neq m} B(C(m'),pn) \,] \leq \exp(-a_\varepsilon\, n).$$

Various efficient (polytime encodable/decodable) constructions

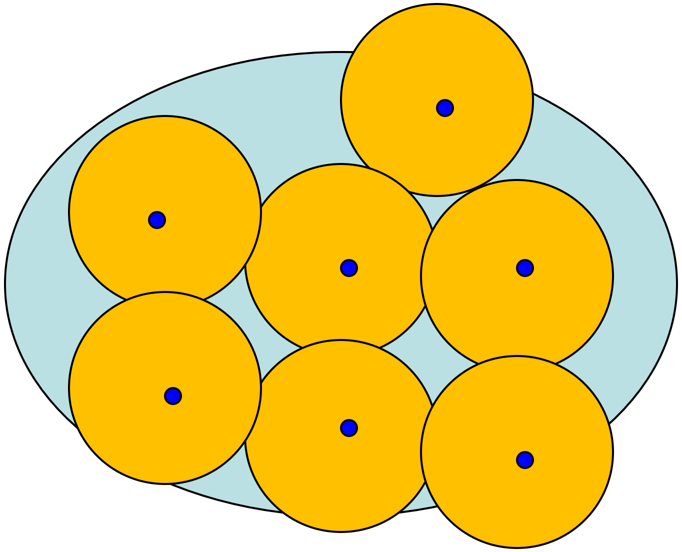
- Concatenated codes
- LDPC codes*
- Polar codes

i.i.d errors is a strong assumption

- eg., errors often bursty…

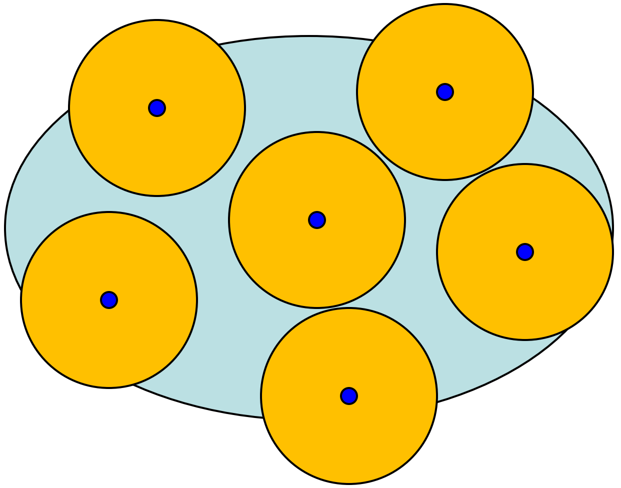
What about worst-case errors?

- all we know is $wt(e) \leq pn$



Worst-case errors

Largest rate of binary code s.t. Hamming balls of radius pn around them are *fully* disjoint?



<u>Answer</u>: Unknown!

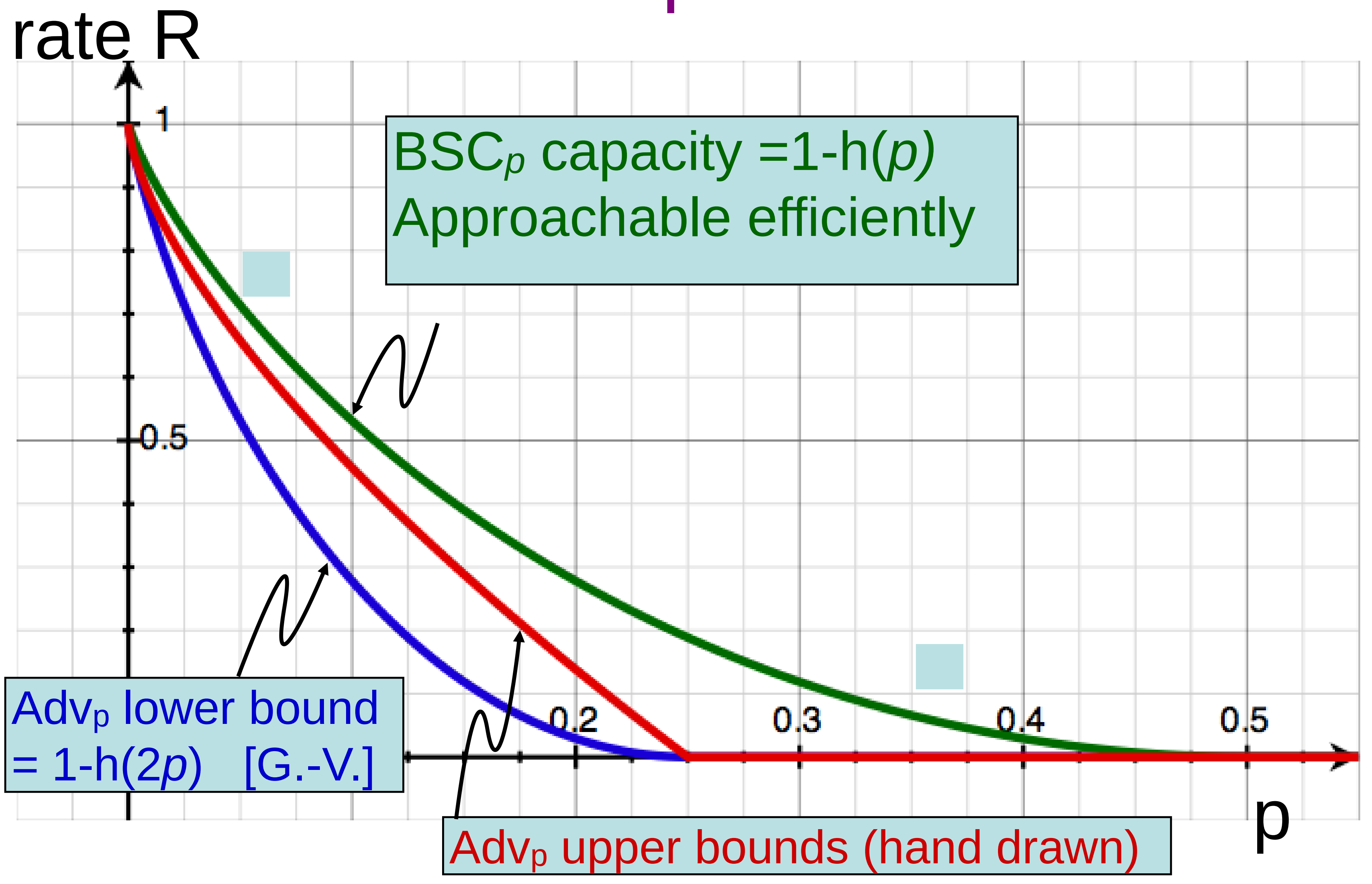
But it is *strictly* $< 1-h(p)$

- Rate $\rightarrow 0$ for $p \geq 1/4$.
- Best known rate (existential)
  - $1-h(2p)$

Big price:

- for similar rate, can correct only $\approx 1/2$ # errors for worst-case model

A plot


rate R
1
0.5
BSC$_p$ capacity =1-h(p)
Approachable efficiently
Adv$_p$ lower bound
= 1-h(2p)   [G.-V.]
0.2
0.3
0.4
0.5
p
Adv$_p$ upper bounds (hand drawn)

Why care about worst-case errors?

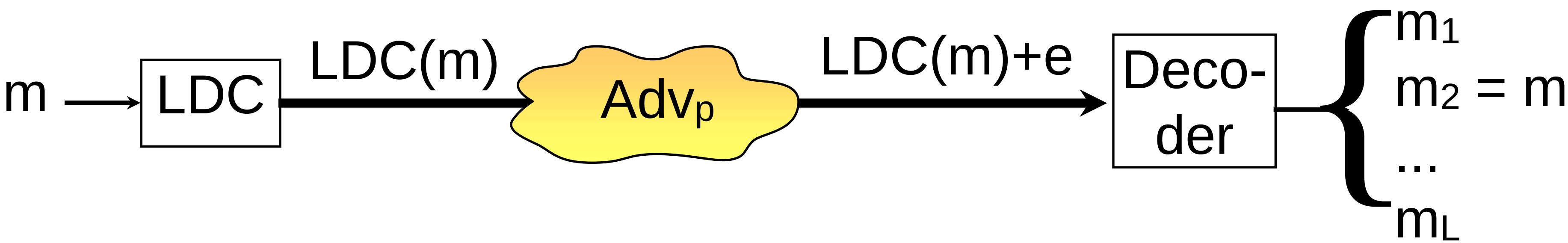
- As computer scientists, we like to!
- “Extraneous” applications of codes
  - Cryptography, complexity theory (pseudorandomness, hardness amplification, etc.)

Communication: Modeling *unknown* or *varying* channels

- Codes for probabilistic model may fail if stochastic assumptions are wrong
  - Eg. Concatenated codes for bursty errors
- Codes for worst-case errors robust against variety of channels

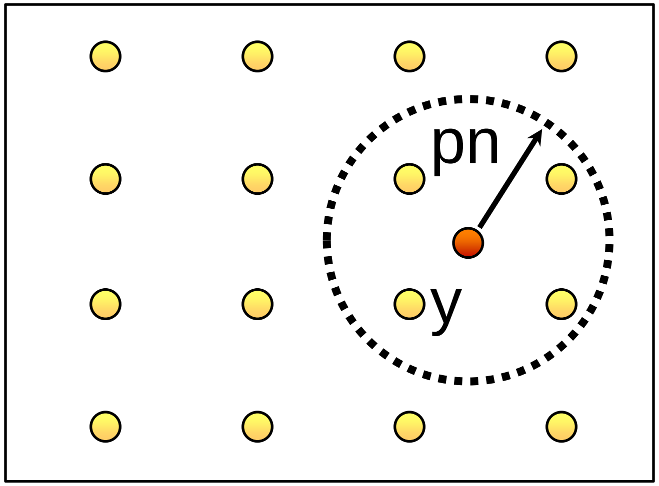
Bridging Shannon & Hamming I

<u>List decoding</u>: Relax decoding goal; recover small list of messages (that includes correct message m)



LDC: $\{0,1\}^k \rightarrow \{0,1\}^n$ is *(p,L)-list-decodable* if

- every $y \in \{0,1\}^n$ is within distance pn of $\leq L$ codewords



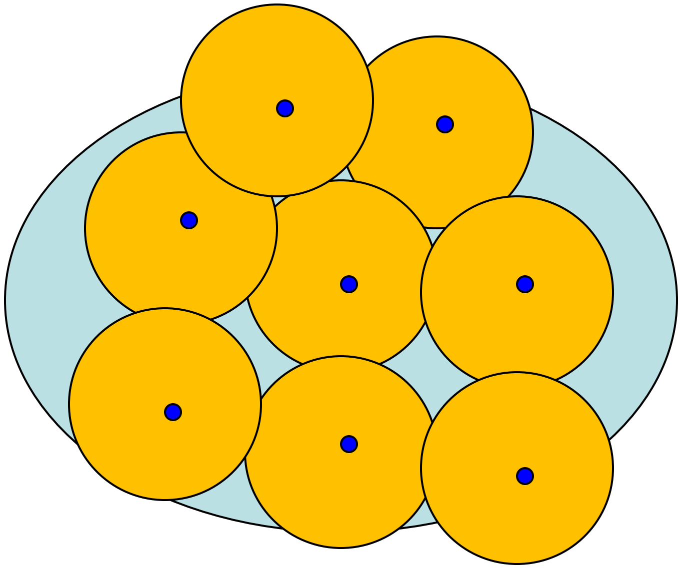
List decoding & Shannon capacity

<u>Thm</u> [Zyablov-Pinkser'81,Elias'91]: W.h.p., a random code of rate $1-h(p)-\varepsilon$ is (p,L)-list-decodable for list size $L = 1/\varepsilon$

$\Leftrightarrow$ Packing of radius pn Hamming balls covering each point $\leq 1/\varepsilon$ times

[G.-Håstad-Kopparty'10]:

- Also true for random *linear* code

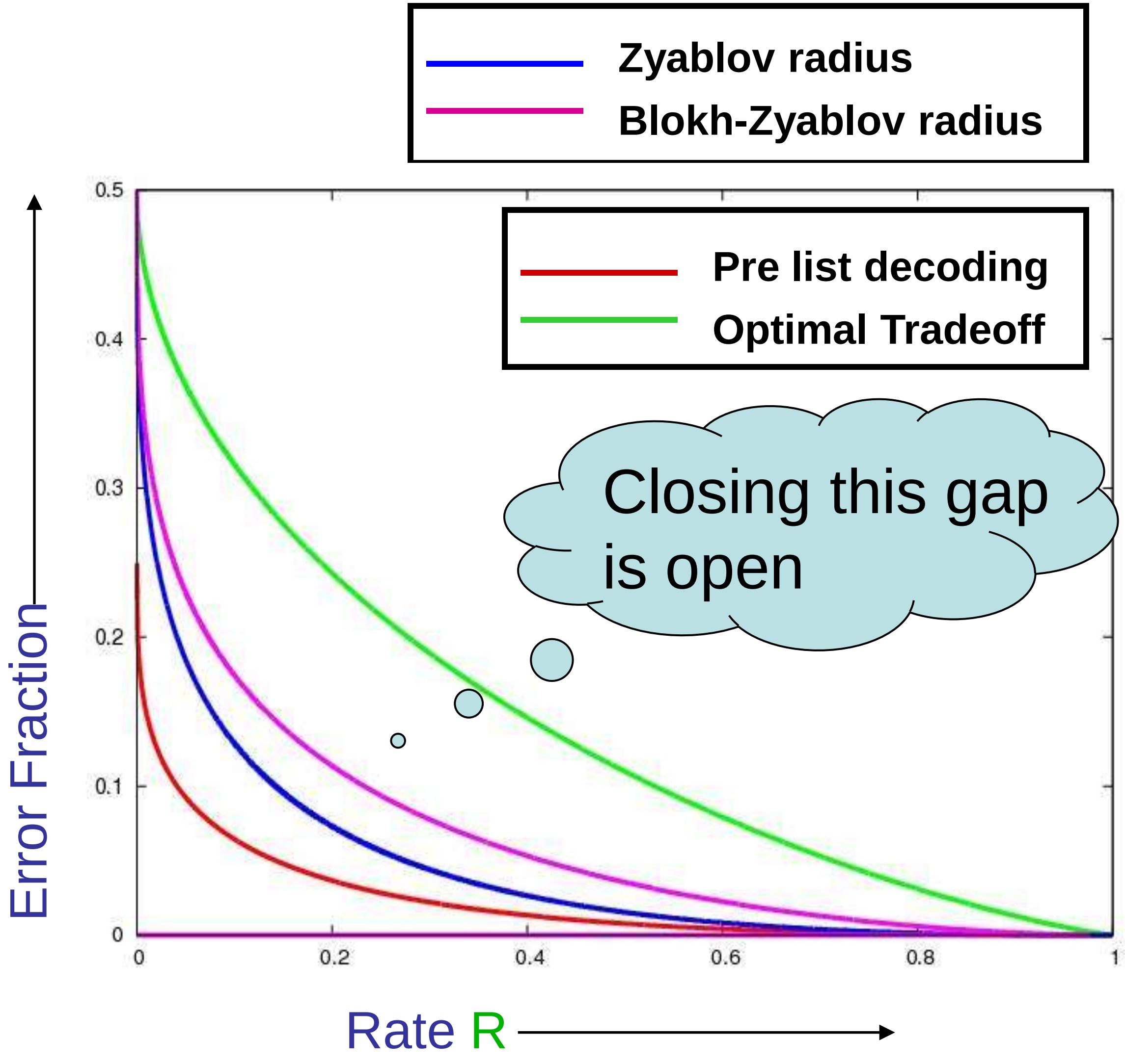


Is having a list useful?

Yes, for various reasons

- better than giving up,
- w.h.p. list size 1,
- fits the bill perfectly in complexity applications
- Versatile primitive (will see in this talk!)

Unfortunately, no constructive result achieving rate $\rightarrow$ 1-h(p) is known for binary list decoding



Optimal trade-off
$R \approx 1 - h(p)$

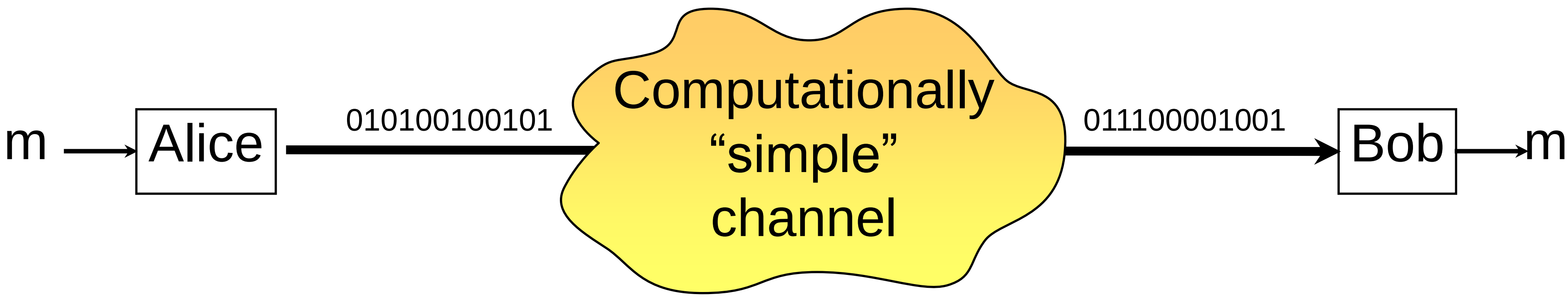
Constructive:
Zyablov, Blokh-Zyablov:
[G.-Rudra'08,'09]
Polynomial-based codes + concatenation

Outline

- Background & context
  - Error models, Shannon & Hamming
  - List decoding
- Computationally bounded channels
  - Previous results (with "setup")
- Our results
  - Explicit optimal rate codes (for two simple channels)
- Proof tools & ideas

Computationally limited channels

- Channel models that lie between adversarial channels and specific stochastic assumptions



- [Lipton'94] : "simple" = simulatable by small circuit
  - *Natural processes may be mercurial, but perhaps not arbitrarily malicious*
  - Eg. $O(n^2)$ boolean gates for block length n
    - Covers models in literature such as AVCs.
  - studied in [Ding-Gopalan-Lipton'06, Micali-Peikert-Sudan-Wilson'06]

Computationally limited channels

Formally: channel class specified by

- Complexity of channel
- Error parameter **p**: channel introduces **$\leq pn$** errors w.h.p.

Examples:

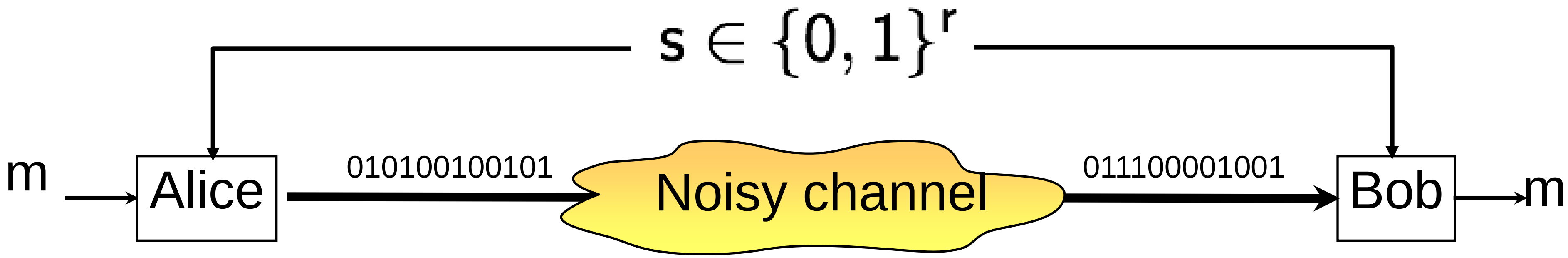
- Polynomial-size: circuits of size **n^b** for known **b**
- Log-space: one-pass circuit using **$O(\log n)$** bits of memory
- Additive channel: XOR with arbitrary oblivious error vector

Single code must work for all channels in class

Previous work

Need *setup* assumptions:

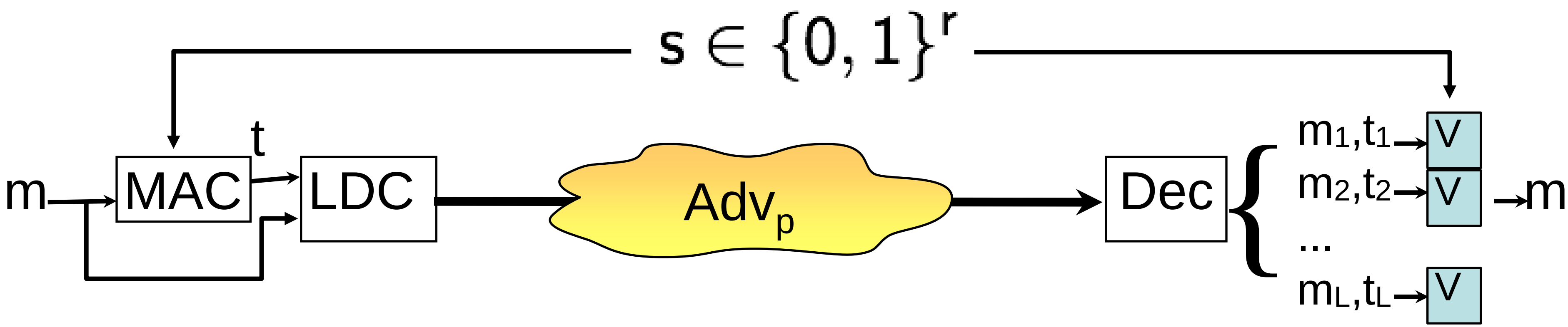
- [Lipton 1994]: shared secret randomness
  - Encoder/decoder share random bits s hidden from channel



- [Micali-Peikert-Sudan-Wilson 2006]: public key
  - Bob, channel have Alice's public key; only Alice has private key
  - Alice uses private key to encode

Private codes

With shared randomness, *don't even need any computational assumption* if we had optimal rate ***list-decodable*** codes* [Langberg'04, Smith'07]



<u>Idea:</u> Alice <u>*authenticates*</u> m using s as key

- If MAC has forgery probability δ, then Bob fails to uniquely decode m with probability ≤ L δ
- MAC tag can have tag & key length O(log n)
  - O(log n) shared randomness
  - negligible loss in rate

**(which we don't)*

Our Results

(Optimal rate) codes with no shared setup

1. **Additive errors**: efficient, uniquely decodable codes that approach Shannon capacity ($1-h(p)$)
   - Previously: only inefficient constructions known via random coding [Cziszar-Narayan'88,'89; Langberg'08]
   - We also provide a simpler existence proof

Formally, explicit randomized code

$C : \{0,1\}^k \times \{0,1\}^r \rightarrow \{0,1\}^n$ of rate $k/n=1-h(p)-\varepsilon$ &

efficient decoder Dec such th

$\forall m\ \forall e$, $wt(e) \leq pn$,

$\mathrm{Prob}_{\omega}\,[\,\mathrm{Dec}(C(m,\omega) + e) = m\,] > 1 - o(1)$

Decoder doesn't know encoder's random bits

Our Results

(Optimal rate) codes with no shared setup

2. Logspace errors: efficient list-decodable code with optimal rate (approaching 1-h(p))

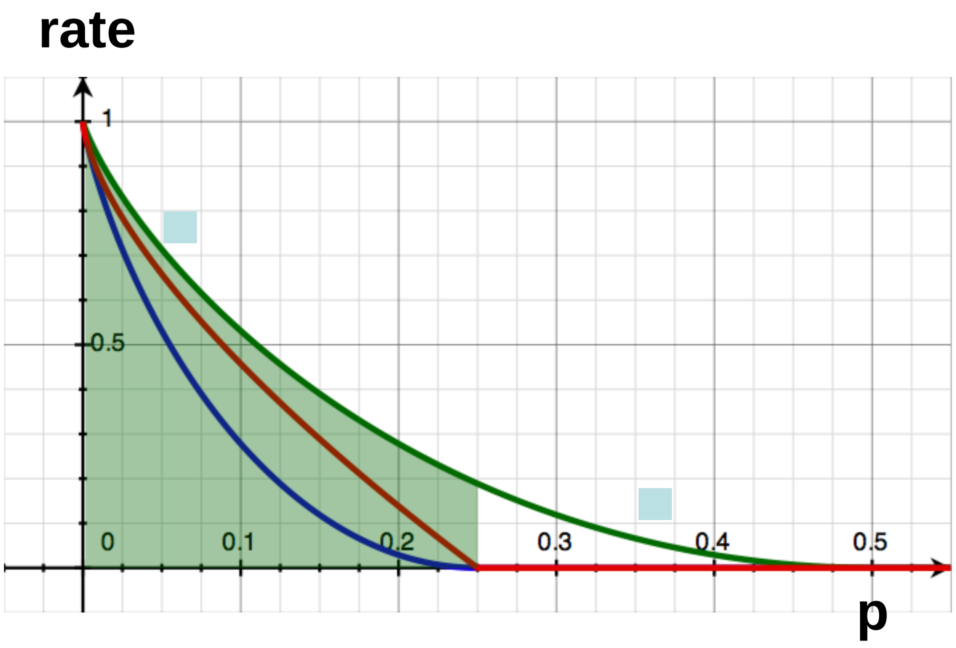
- Previously: no better than uniquely-decodable codes
- List decoding = decoder outputs L messages one of which is m w.h.p. (*not* all close-by codewords)

3. Polynomial-time errors: efficient list-decodable code with rate ≈ 1-h(p), assuming *p.r.g.*

Why list decoding?

Lemma: Unique decoding has rate zero when $p > 1/4$ even for simple bit-fixing channel (which is O(1) space)

Open: Unique decoding past worst-case errors for $p < 1/4$ for low-space online channels ?



The ¼ barrier

Lemma's proof idea:

- Channel moves codeword $c=C(m,\omega)$ towards random codeword $c'=C(m',\omega')$, flipping c_i with probability ½ when $c_i \neq c'_i$
  - constant space
  - expected fraction of flips $\leq$ ¼
  - Output distribution symmetric w.r.t. inversion of c and c'

Technical Part

Additive/oblivious errors

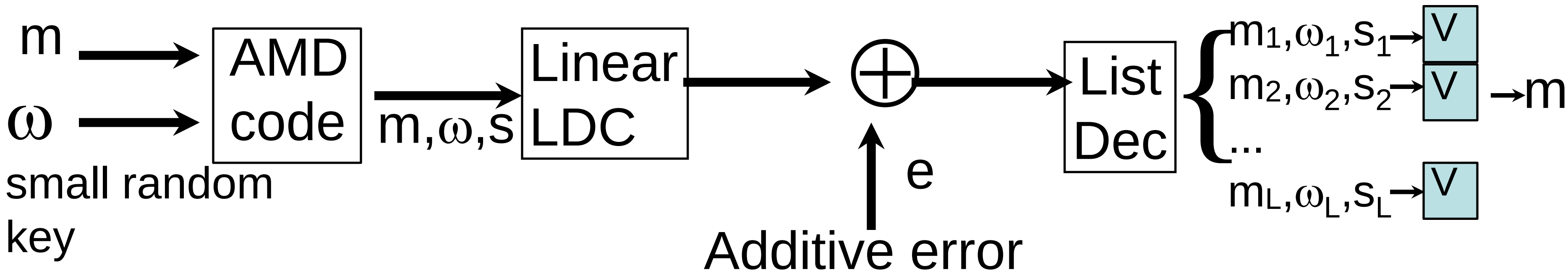
Randomized code $C : \{0,1\}^k \times \{0,1\}^r \rightarrow \{0,1\}^n$ of rate $k/n = 1 - h(p) - \varepsilon$ & decoding function Dec s.t.

$\forall m \ \forall e$, $wt(e) \leq pn$,

$\text{Prob}_\omega [\ \text{Dec}(C(m,\omega) + e) = m\ ] > 1 - o(1)$

New existence proof

Linear list-decodable code + "additive" MAC (called **A**lgebraic **M**anipulation **D**etection code, [Cramer-Dodis-Fehr-Padro-Wichs'08])

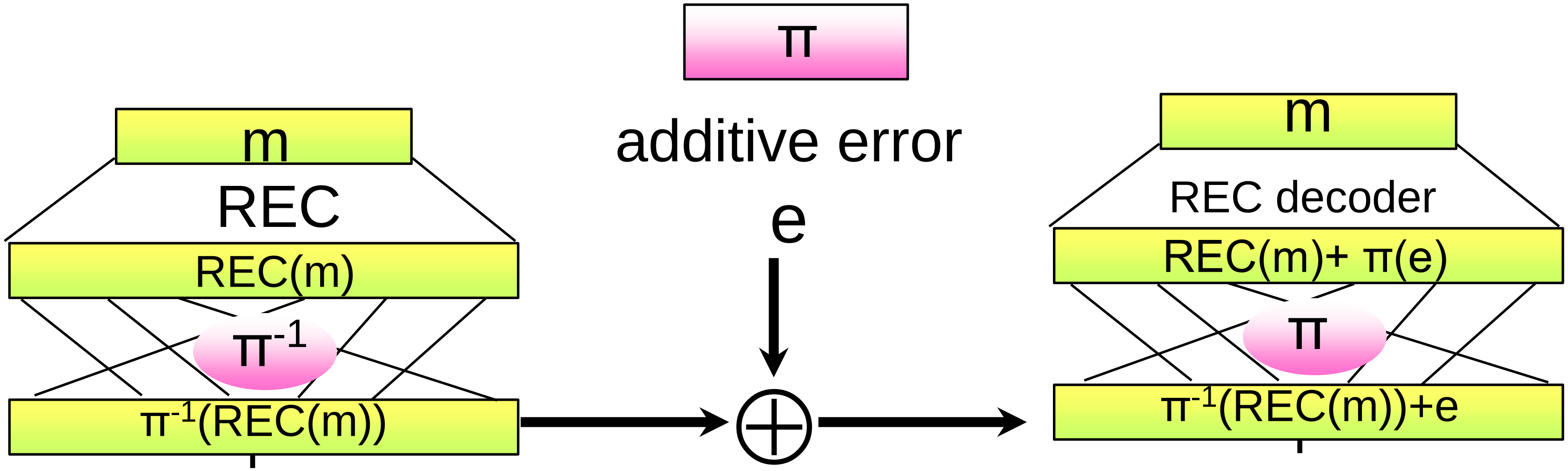


Decoder can disambiguate ***without*** knowing ω

Key point: For fixed e, the *additive offsets* of the spurious (m_i,ω_i,s_i) from (m,ω,s) are fixed.

Unlikely these L offsets cause forgery.

Code scrambling:
a simple solution with shared randomness

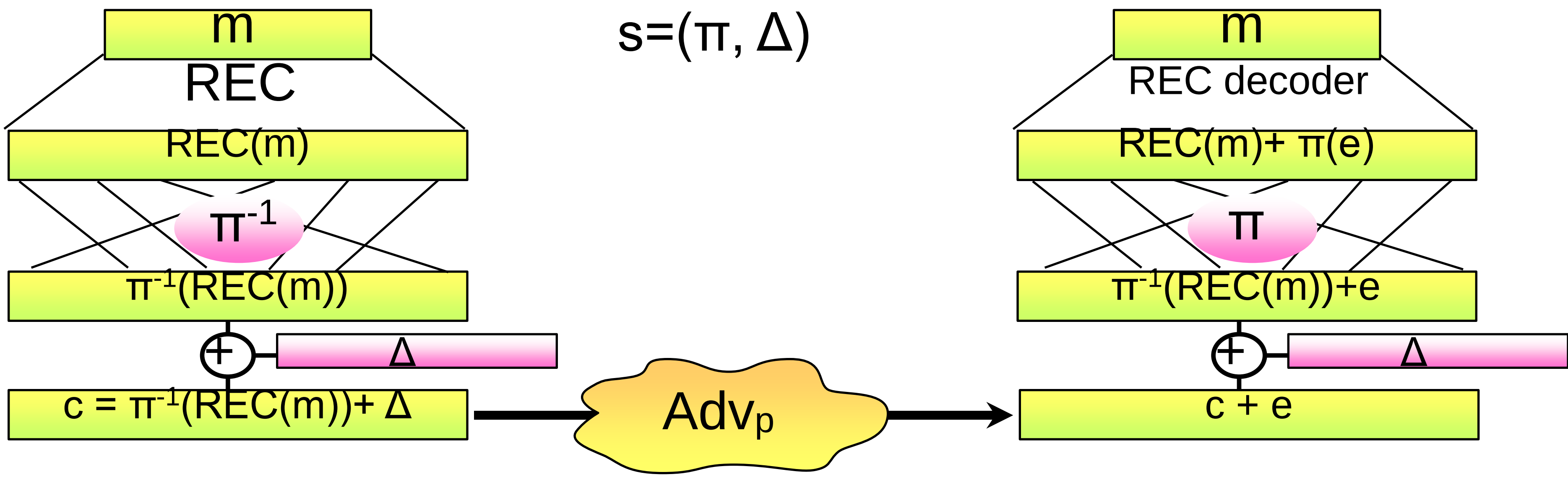


Shared random permutation π of {1,...,n}

- Code REC of rate ≈ 1-h(p) to correct fraction p ***random*** errors [eg. Forney's concatenated codes]
- Encoding: $c = \pi^{-1}(\mathrm{REC}(m))$
- Effectively permutes e into *random* error vector

Comment

- Similar solution works for *adversarial* errors Adv_p
- Shared randomness = (π, Δ)
  - Δ acts as one-time pad, making e independent of π



Explicit codes for additive errors (with no shared setup)

Explicit randomized code $C : \{0,1\}^k \times \{0,1\}^r \rightarrow \{0,1\}^n$ of rate $k/n = 1 - h(p) - \varepsilon$ & *efficient* decoder Dec s.t.

$\forall m\ \forall e$, $wt(e) \leq pn$,

$$\mathrm{Prob}_{\omega}\left[\, Dec(C(m,\omega) + e) = m \,\right] > 1 - o(1)$$

Eliminating shared setup

Idea: Hide shared key ("control information") in codeword itself

- Use a control code to encode control info (to protect it from errors)
- Ensure decoder can recover control info correctly
  - Must hide its encoding in "random" locations of overall codeword (and control info includes this data also!)
  - ***But isn't this the original problem?***
    - *And doesn't control code hurt the rate?*
- With control info correctly recovered, can appeal to shared randomness solution (unscramble & run REC decoder)

Control code

To afford encoding control information ω without losing overall rate, have to keep it small, say $\varepsilon^2 n$ bits long

- ω can't be uniformly random permutation

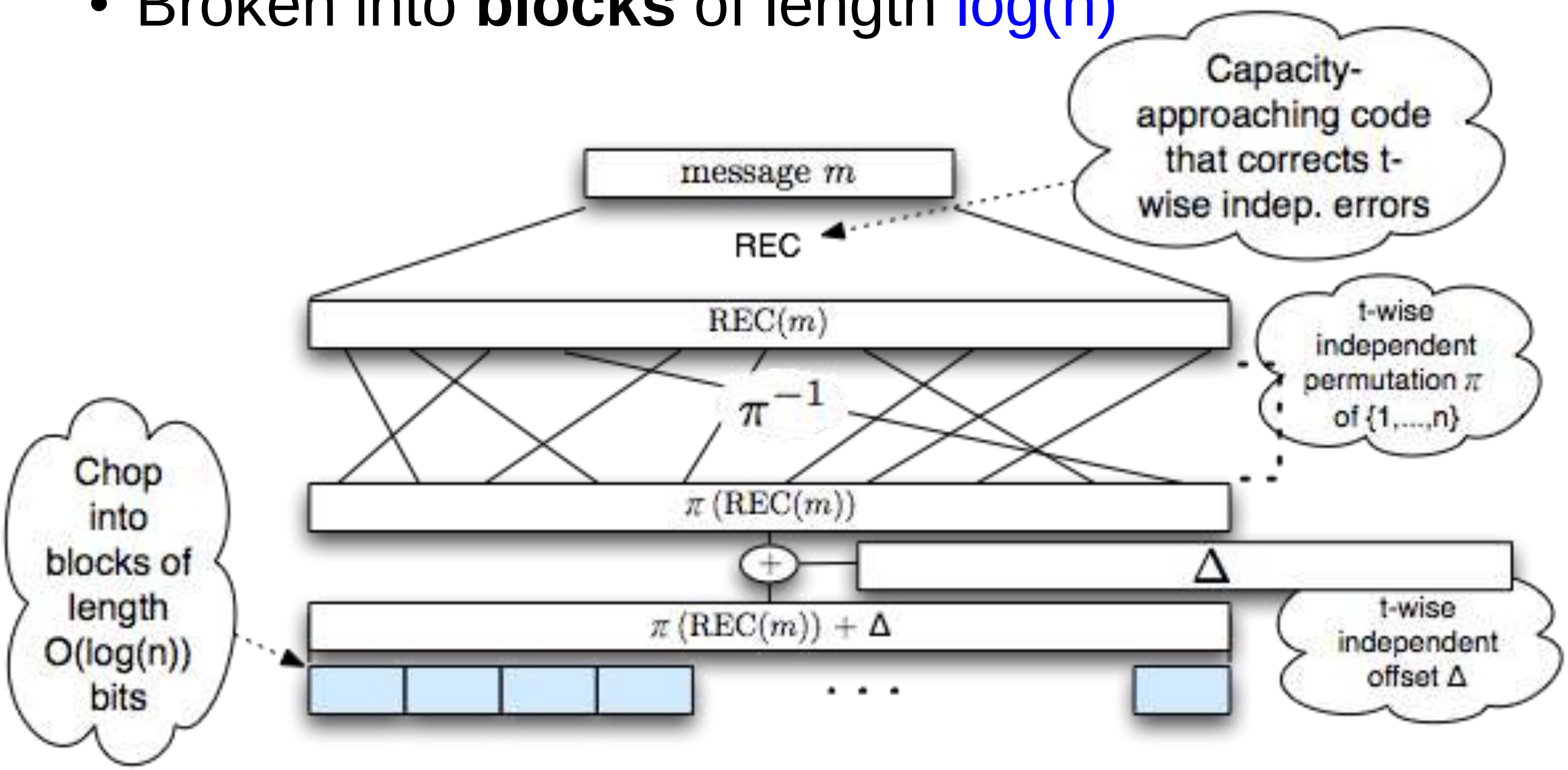
But, if we make ω small, we can use *very low-rate* code to safeguard it

- eg., encode it into εn bits
  (still negligible effect on overall rate)

- Weaker goal (rate << capacity), thus easier

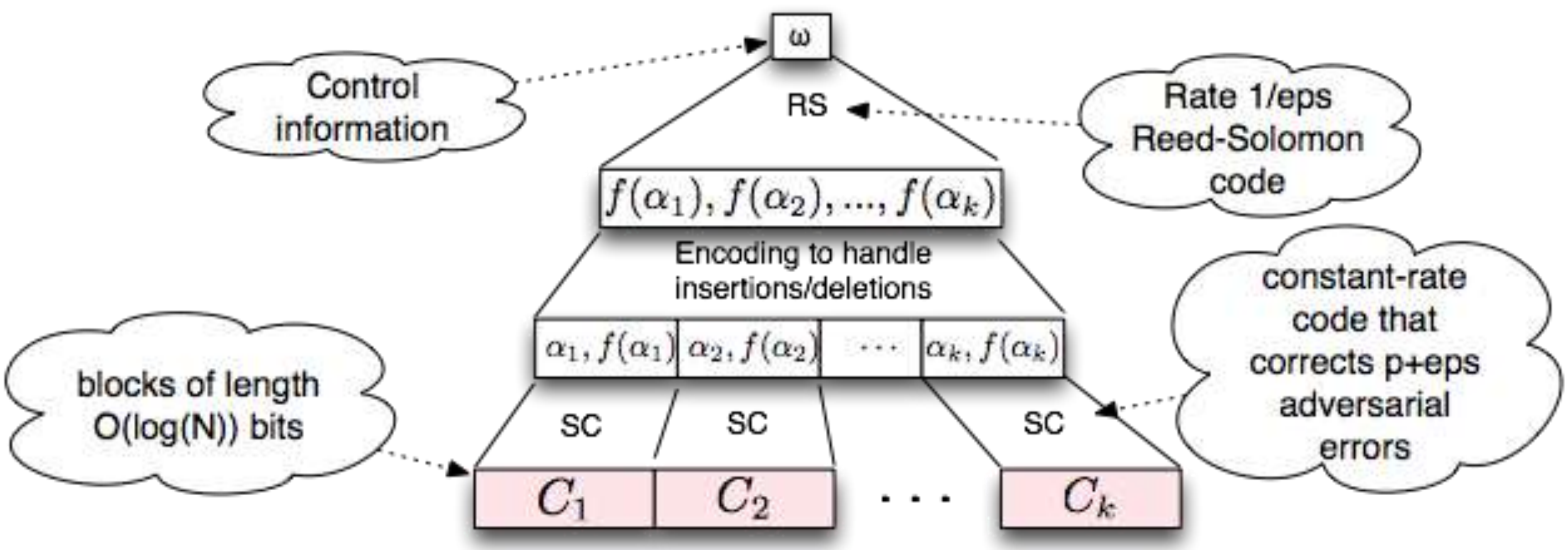
Overall construction

- Two main pieces
  - Scrambled “payload” codeword: $\pi^{-1}(\mathrm{REC}(m)) + \Delta$
    - π is a $\log^2(n)$-wise independent permutation,
    - Δ is a $\log^2(n)$-wise independent bit string
    - Broken into **blocks** of length log(n)



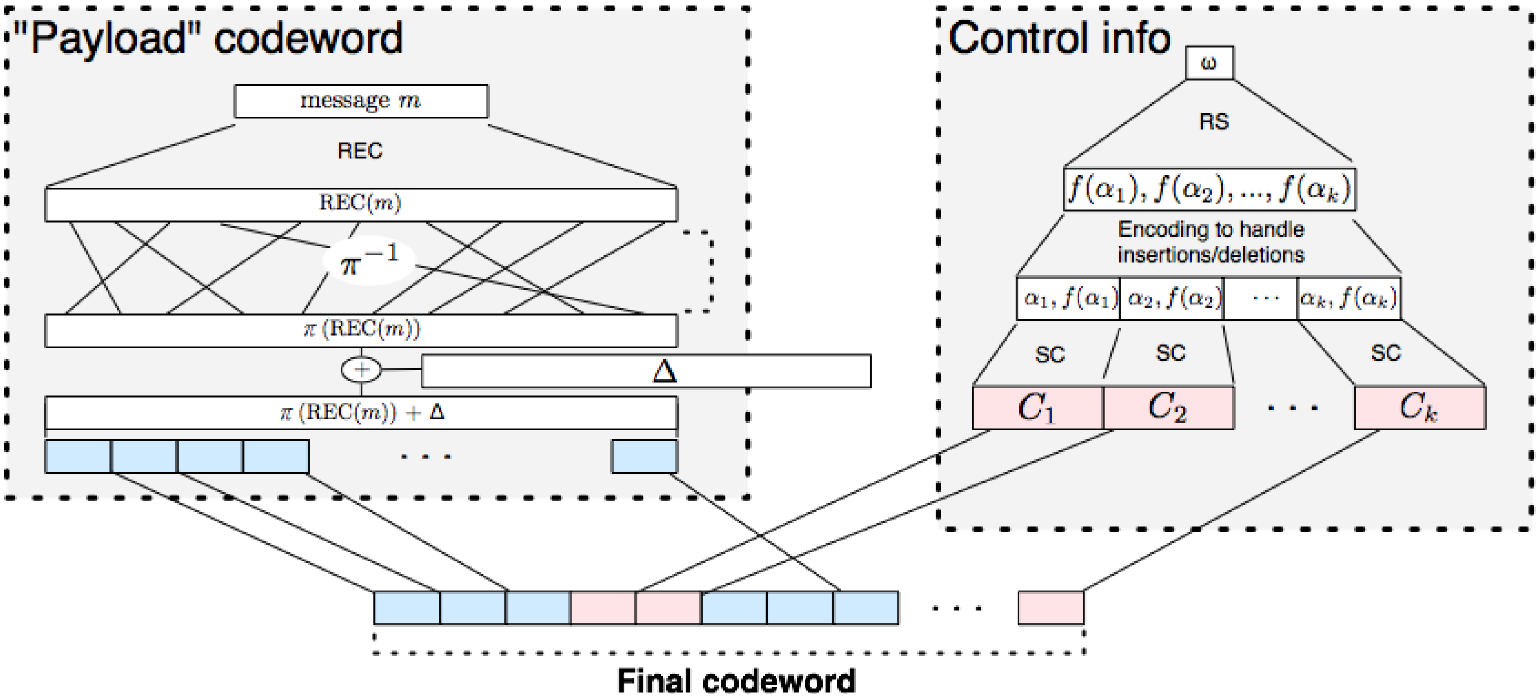
Overall construction

- Two main pieces
  - Scrambled payload codeword: $\pi^{-1}(\mathrm{REC}(m)) + \Delta$
  - Control information: ω = (~~π, Δ,T~~) Standard "sampler"
    - T is a (pseudorandom) subset of blocks in {1,..., n/log(n)}
    - Encode ω via low-rate Reed-Solomon-code into "control blocks"
    - Encode each control block via small LDC+AMD code



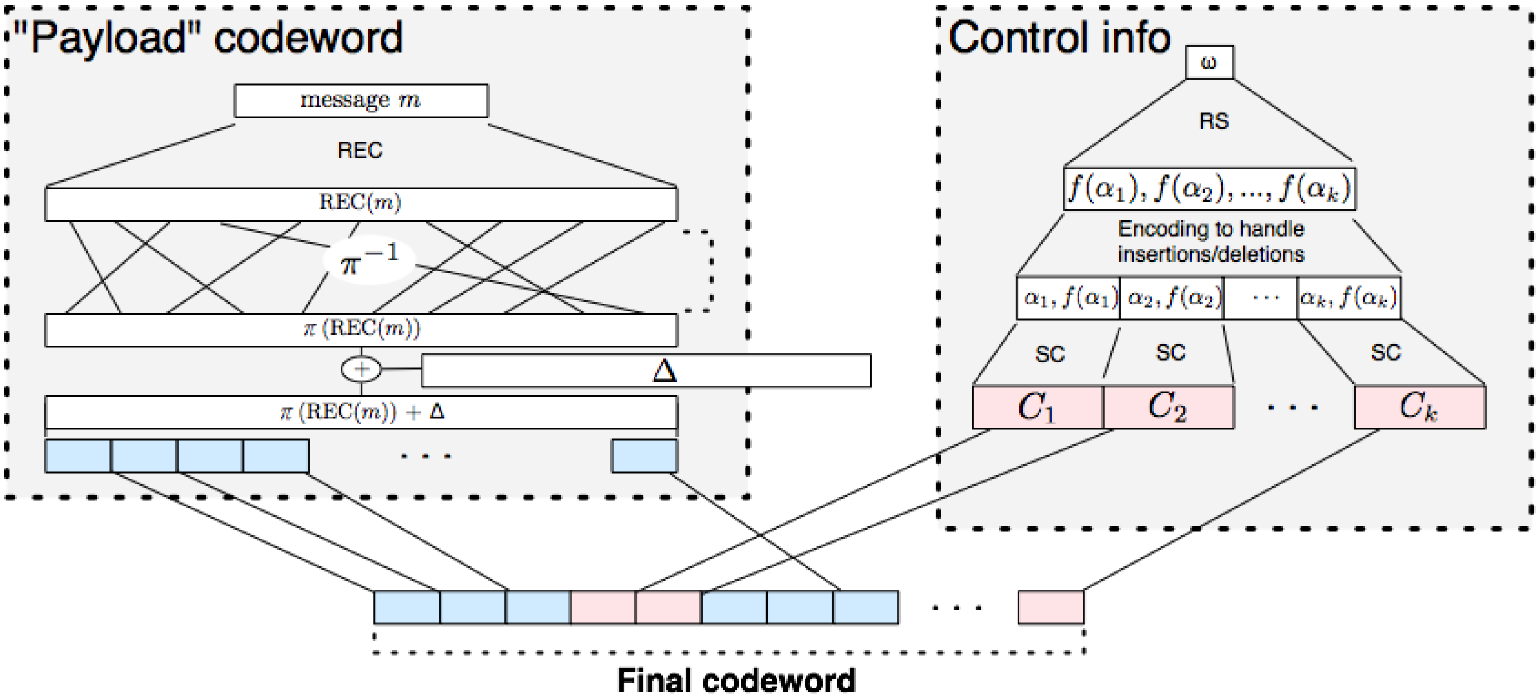
Control/payload construction

- Two main pieces
  - Scrambled payload codeword: $\pi^{-1}(\mathrm{REC}(m)) + \Delta$
  - Control information: $\omega = (\pi, \Delta, T)$
- Combine by interleaving according to T



Decoding idea

- First decode control information, block by block
- Given control info, unscramble payload part & run REC decoder



Control info recovery

- Pseudorandomness of $T \Rightarrow$ enough ($\approx \varepsilon$ n) control blocks have $< (p+\varepsilon)$ errors.
- But decoder is not handed T
  - So does *not* know which blocks are control blocks
- Decode each block up to radius $p+\varepsilon$
  - By properties of "inner" LDC+AMD construction, enough control blocks correctly decoded
  - Random offset $\Delta \Rightarrow$ payload blocks look random
    - Far from every control codeword
    - so very few mistaken for control blocks

$\Rightarrow$ Reed-Solomon decoder recovers ω correctly

Finishing decoding

- Control decoding successful $\Rightarrow$ decoder knows ω, so can
  - remove offset Δ and apply π,
  - run REC decoder (which works for $\log^2 n$-wise independent errors) on $REC(m) + \pi(e)$
  - recover m w.h.p.

Online logspace channels

- Similar high level structure; details more complicated
- Use "pseudorandom" codes to hide location of control information from channel
  - Small codes whose output looks random to channel
    - Efficiently decodable by (more powerful) decoder
  - Ensures enough control blocks have few errors
- But channel can inject many "fake" legitimate looking control blocks
  - Overcome by resorting to list decoding
  - recover small list $\{\omega_1, \omega_2, \ldots, \omega_L\}$ containing true ω

Online logspace channels: Payload decoding

- Ensure channel's error distribution is indistinguishable (in online logspace) from an oblivious distribution
  - How? Nisan's PRG to produce offset Δ that fools channel
- Given correct control info, argue events that ensured successful decoding in oblivious case also occur w.h.p. against more powerful online logspace channel
  - event $\approx$ error is "well-distributed" for REC decoder
  - Problem: this "well-distributed"-ness can't be checked in *online* logspace
  - Solution: work with a weaker condition that *can* be checked in online logspace (leads to worse o(1) failure bound)

SIZE(n^b) channels

- Replace Nisan by appropriate efficient pseudorandom generator for SIZE(n^b) circuits
  - Exists under computational assumptions (like one-way functions)

- Analysis easier than online logspace case, as one only needs polytime distinguisher

Summary

- List decoding allows communicating at optimal rate even against *adversarial* errors, but explicit constructions not known (for binary case)

- Bounding complexity of channel “new” way to capture limited adversarial behavior
  - well-motivated bridge between Shannon & Hamming

- Our results: Explicit optimal rate codes for
  - additive errors
  - List decoding against online logspace channels

Open questions

For unique decoding on *online logspace* channels

- Is better rate possible than adversarial channels for $p < ¼$?
- Better rate upper bound than $1\text{-}h(p)$ for $p < ¼$?

Online adversarial channels

- Rate upper bound of $\min\{1\text{-}4p, 1\text{-}h(p)\}$

[Langberg-Jaggi-Dey'09]

- True trade-off ?