Zero Knowledge Proofs

Polynomial Commitments
based on error-correcting codes

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, **Yupeng Zhang**





Stanford University

Berkeley UNIVERSITY OF CALIFORNIA

GEORGETOWN UNIVERSITY

TEXAS A&M UNIVERSITY

Recall: common paradigm for efficient SNARK

A polynomial commitment scheme

A polynomial interactive oracle proof (IOP)

SNARK for general circuits

Last time: KZG polynomial commitment

Univariate polynomials of degree <= *d*

$gp = (g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d})$

Prover

$\boldsymbol{f(x)}$

Verifier

$\boldsymbol{com_f}$

$com_f = g^{f(\tau)}$

u

v, proof $\pi = g^{q(\tau)}$

$$f(x) - f(u) = (x-u)q(x)$$

$$e(com_f / g^{v}, g) = e(g^{\tau - u}, \pi)$$

Last time: other PC based on discrete-log

Scheme	Prover	Proof size	Verifier	Trusted Setup	Crypto primitive
Bullet -proofs	$O_\lambda(d)$	$O_\lambda(\log d)$	$O_\lambda(d)$	×	discrete-log
Hyrax	$O_\lambda(d)$	$O_\lambda(\sqrt{d})$	$O_\lambda(\sqrt{d})$	×	discrete-log
Dory	$O_\lambda(d)$	$O_\lambda(\log d)$	$O_\lambda(\log d)$	×	pairing
Dark	$O_\lambda(d)$	$O_\lambda(\log d)$	$O_\lambda(\log d)$	×	unknown order group

Poly-commit based on error-correcting codes

Motivations:

- ✓ Plausibly post-quantum secure
- ✓ No group exponentiations (prover only uses hashes, additions and multiplications)
- ✓ Small global parameters

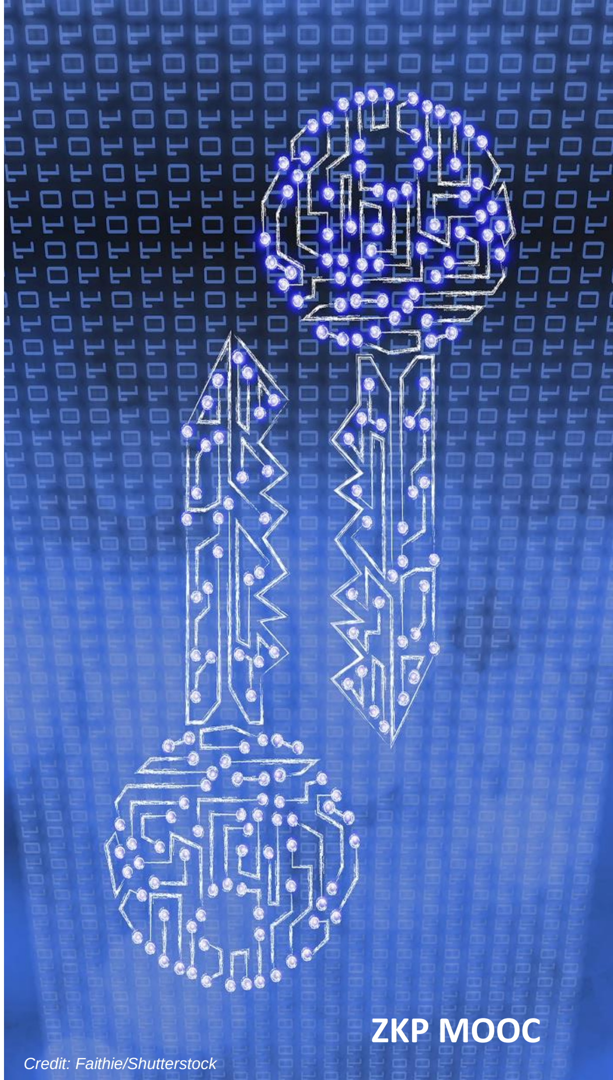
Drawbacks:

- ✗ Large proof size
- ✗ Not homomorphic and hard to aggregate

Plan of this lecture

- Background on error-correcting codes
- Polynomial commitment based on error-correcting codes
- Linear-time encodable code based on expanders

Background

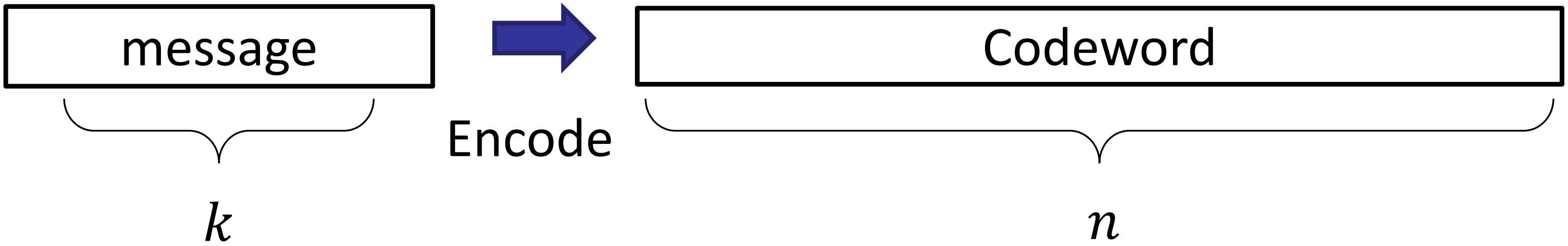


Credit: Faithie/Shutterstock

Error-correcting code

$[n, k, \Delta]$ code:

- Enc(m): Encode a message of size k to a codeword of size n
- Minimum distance (Hamming) between any two codewords is Δ



Example: repetition code

Binary with $k = 2$ and $n = 6$

- Enc(00) = 000000, Enc(01) = 000111
- Enc(10) = 111000, Enc(11) = 111111
- Minimum distance $\Delta = 3$

Can correct 1 error during the transmission

e.g. 010111→ 01   Dec(c): decode algorithm (not used in poly-commit)

Rate and relative distance

Rate: $\frac{k}{n}$ Relative distance: $\frac{\Delta}{n}$

E.g. repetition code with rate $\frac{1}{a}$, $\Delta = a$, relative distance: $\frac{1}{k}$

Trade-off between the rate and the distance of a code

Linear code

Any linear combination of codewords is also a codeword

⇒ Encoding can always be represented as vector-matrix multiplication between m and the generator matrix

⇒ minimum distance is the same as the codeword with the least number of non-zeros (weight).

Example: Reed-Solomon Code

Encode: $\mathbb{F}_p^k \rightarrow \mathbb{F}_p^n$

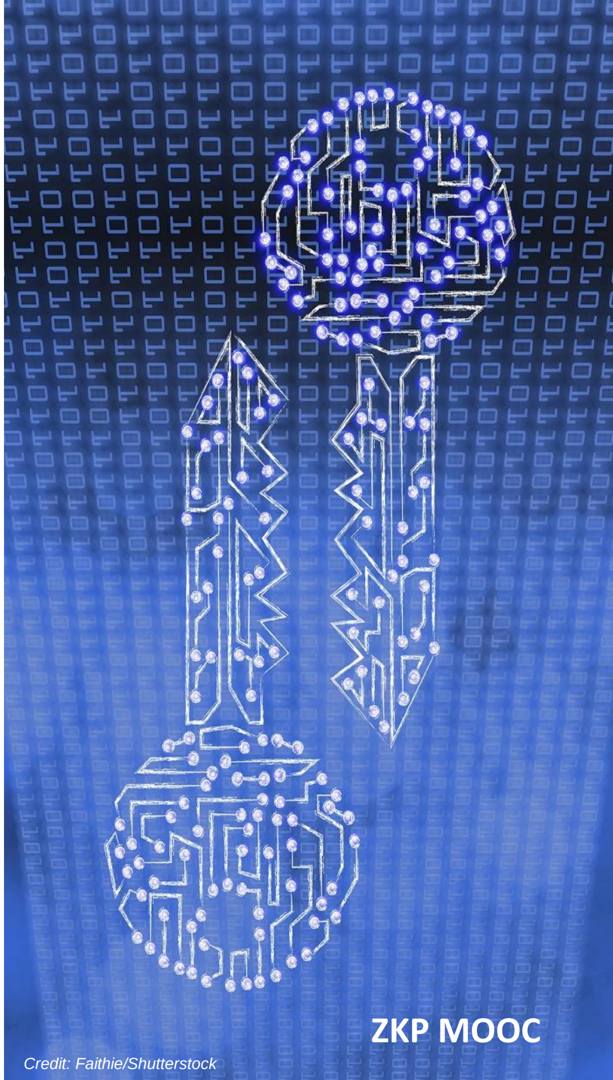
- View the message as a unique degree k-1 univariate polynomial
- The codeword is the evaluations at n points

  E.g., $(\omega, \omega^2, \ldots, \omega^n)$ for n-th root-of-unity $\omega^n = 1 \bmod p$
- Distance $\Delta = n - k + 1$

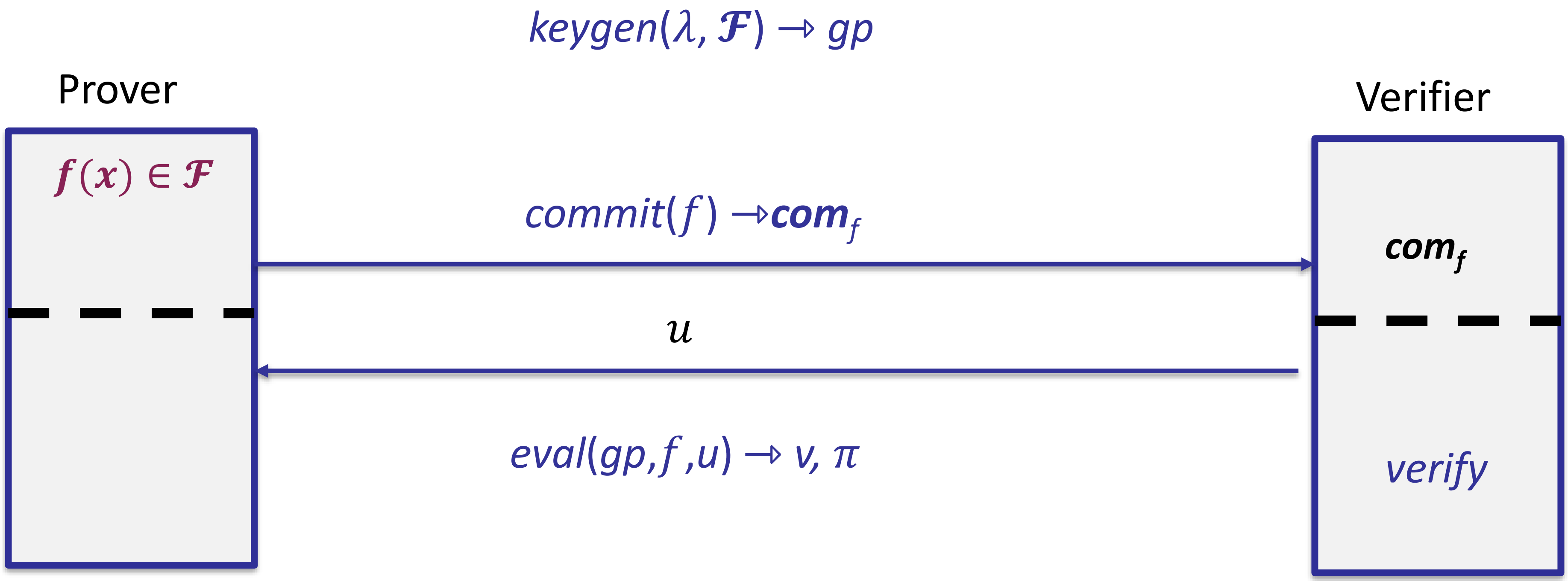
  a degree k-1 polynomial has at most k-1 roots

  E.g, $n = 2k$, rate is 1/2, and relative distance is 1/2
- Encoding time: $O(n \log n)$ using the fast Fourier transform (FFT)

Polynomial commitment based on linear codes



Recall: polynomial commitment



Polynomial coefficients in a matrix

$$\sqrt{d}\left\{\underbrace{\begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,\sqrt{d}} \\ f_{2,1} & f_{2,2} & & f_{2,\sqrt{d}} \\ \vdots & & \ddots & \vdots \\ f_{\sqrt{d},1} & f_{\sqrt{d},2} & \cdots & f_{\sqrt{d},\sqrt{d}} \end{pmatrix}}_{\sqrt{d}}\right.$$

$$f(u) = \sum_{i=1}^{\sqrt{d}} \sum_{j=1}^{\sqrt{d}} f_{i,j} u^{i-1+(j-1)\sqrt{d}}$$

Polynomial evaluation

$$f(u) = \left[1, u, u^2, \dots, u^{\sqrt{d}-1}\right] \times \begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,\sqrt{d}} \\ f_{2,1} & f_{2,2} & & f_{2,\sqrt{d}} \\ \vdots & & \ddots & \vdots \\ f_{\sqrt{d},1} & f_{\sqrt{d},2} & \cdots & f_{\sqrt{d},\sqrt{d}} \end{pmatrix} \times \begin{bmatrix} 1 \\ u^{\sqrt{d}} \\ u^{2\sqrt{d}} \\ \dots \\ u^{d-\sqrt{d}} \end{bmatrix}$$

$$f(u) = \sum_{i=1}^{\sqrt{d}} \sum_{j=1}^{\sqrt{d}} f_{i,j} u^{i-1+(j-1)\sqrt{d}}$$

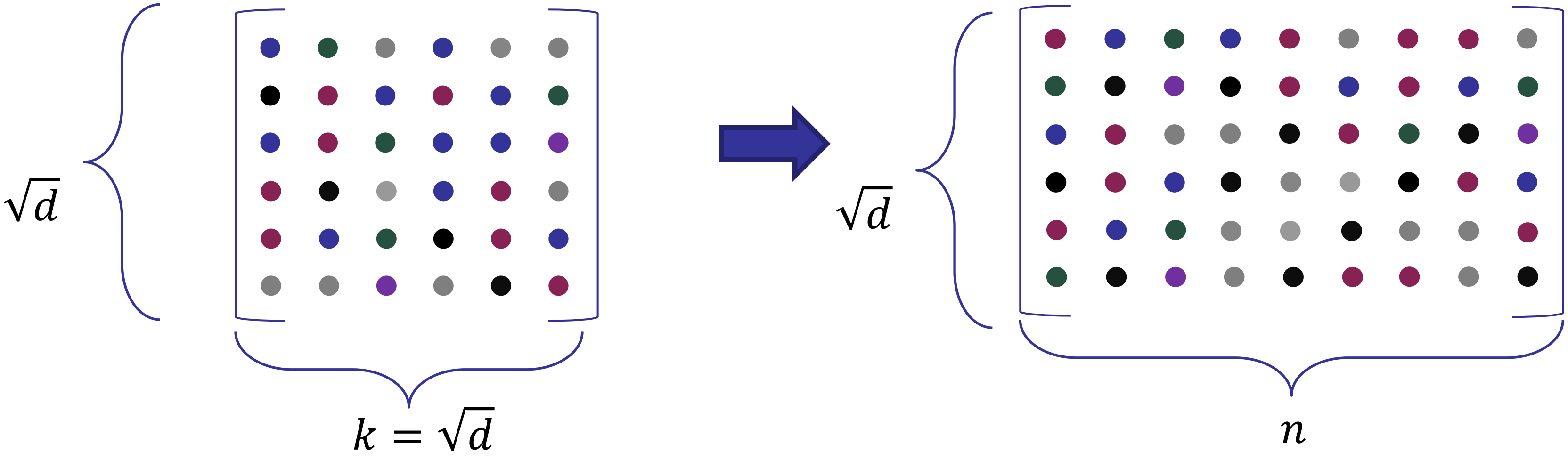
Reducing to Vec-Mat product

$$\left[1, u, u^2, \ldots, u^{\sqrt{d}-1}\right] \times \begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,\sqrt{d}} \\ f_{2,1} & f_{2,2} & & f_{2,\sqrt{d}} \\ \vdots & & \ddots & \vdots \\ f_{\sqrt{d},1} & f_{\sqrt{d},2} & \cdots & f_{\sqrt{d},\sqrt{d}} \end{pmatrix} = \underbrace{[\qquad\qquad\qquad]}_{\sqrt{d}}$$

Argument for Vec-Mat product

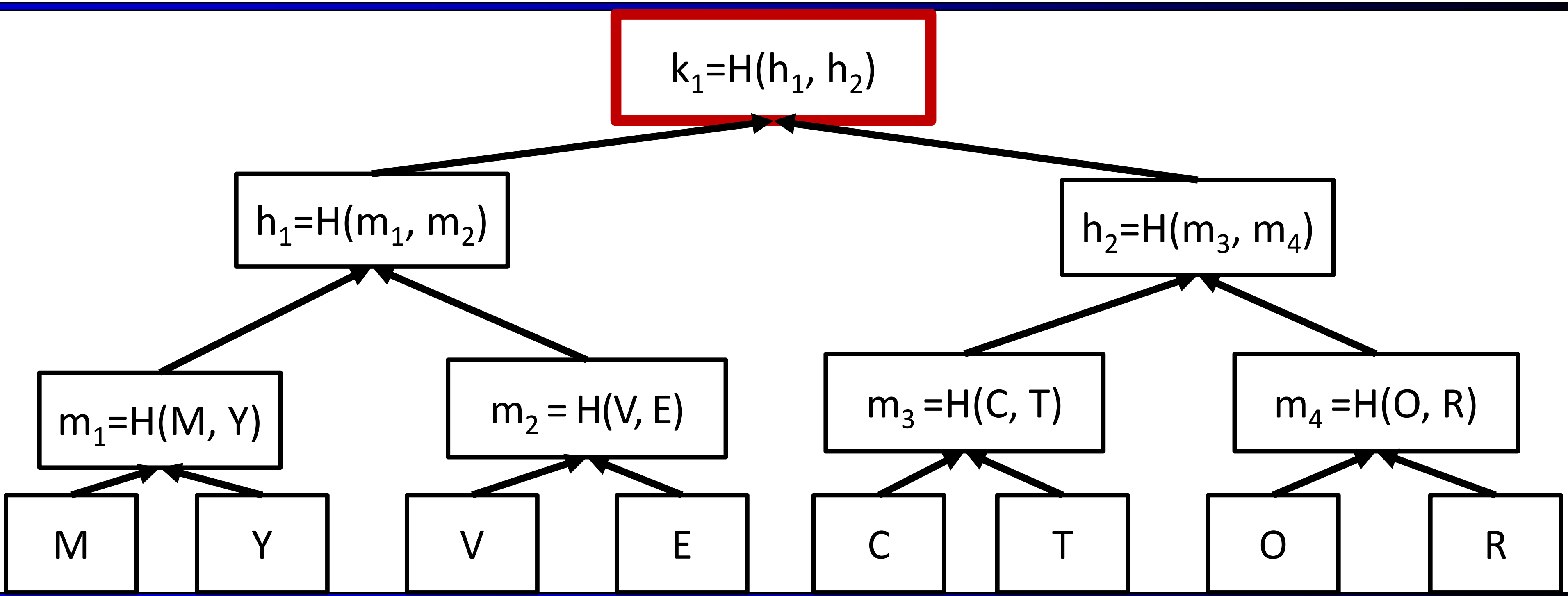
→ Polynomial commitment with $\sqrt{d}$ proof size

Encoding the polynomial

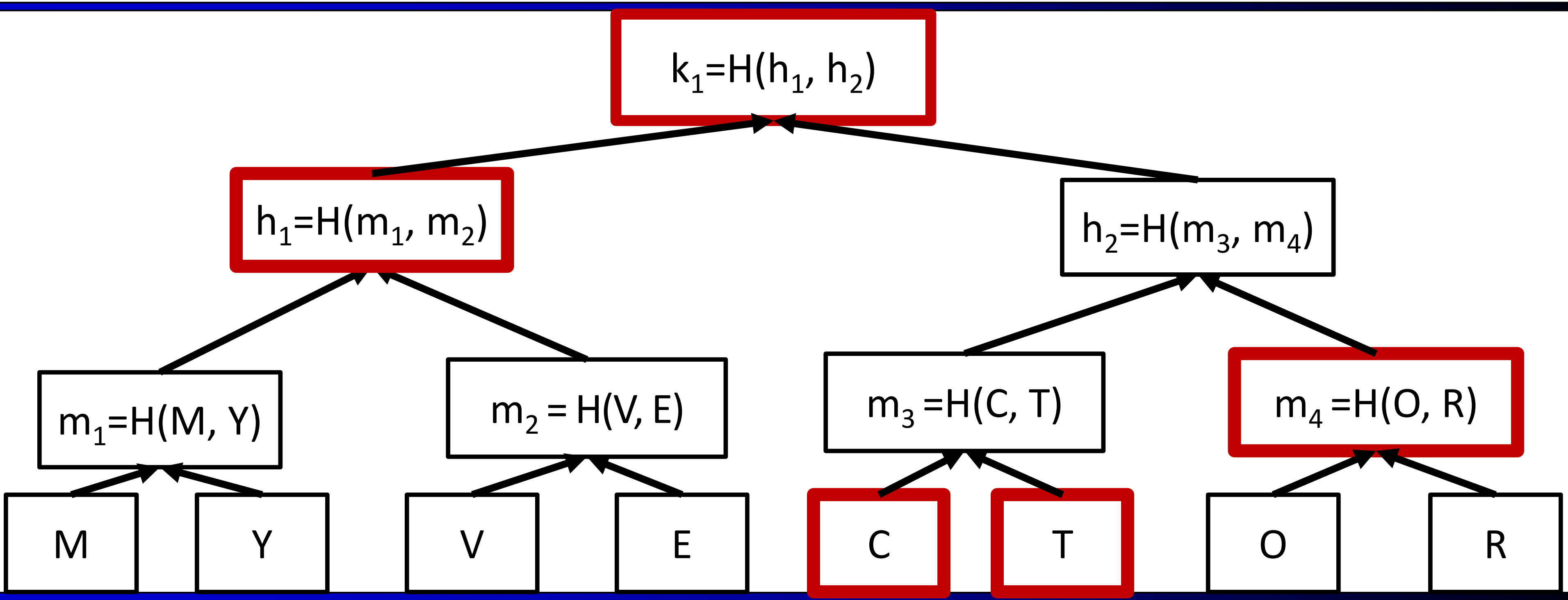


Encode each row with a linear code

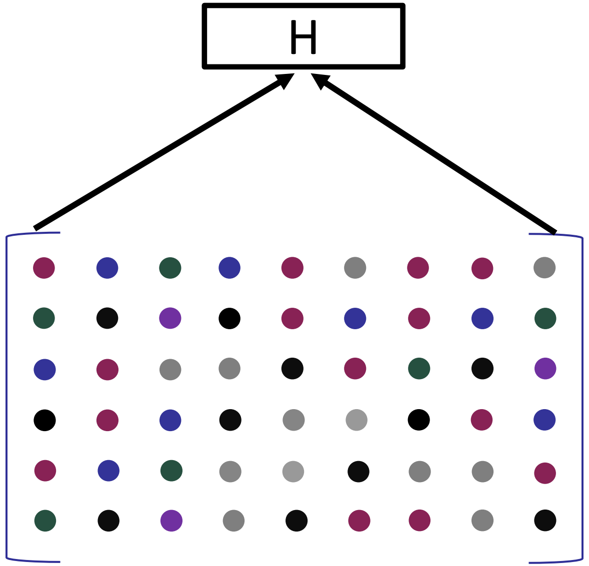
Recall: Merkle tree commitment



Recall: Merkle tree opening



Committing the polynomial

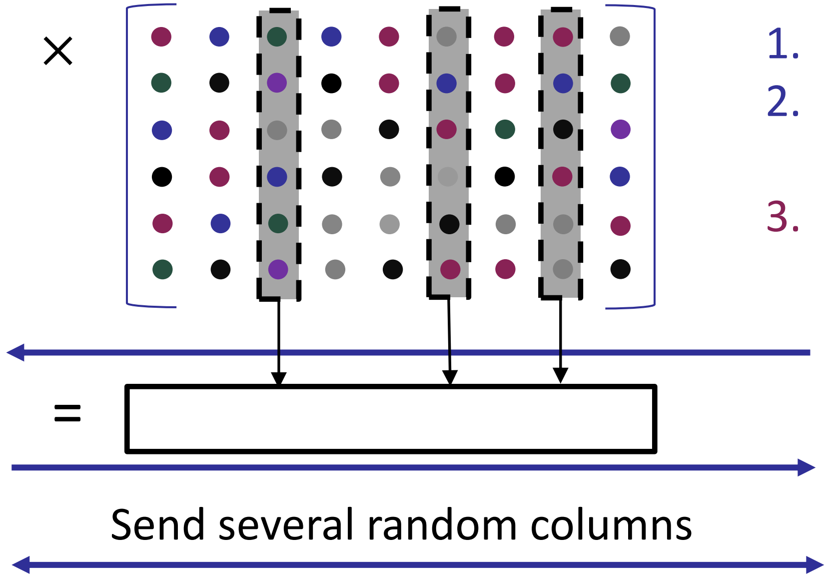


Commit to each column of the encoded matrix using Merkle tree

Step 1: Proximity test

Test if the committed matrix indeed consists of $\sqrt{d}$ codewords

$[r_1, r_2, r_3, \dots, r_{\sqrt{d}}] \times$



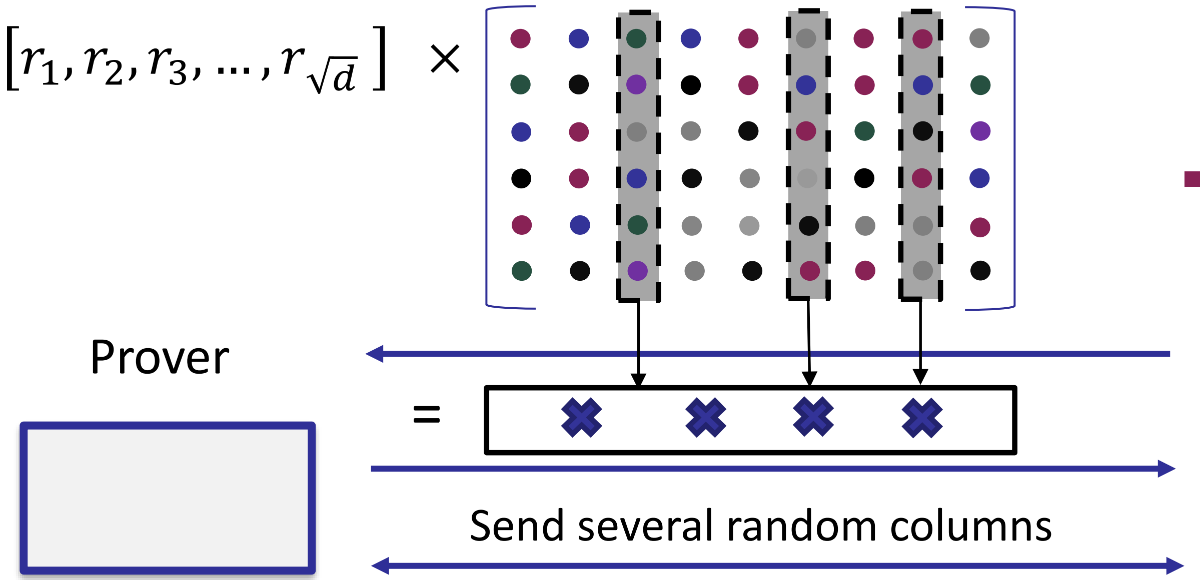
1. The vector is a codeword
2. Columns are as committed in Merkle tree
3. Inner product between *r* and each column is consistent

Prover

Verifier

H

Soundness (Intuition)



Suppose the prover cheats

- If the vector is correctly computed → it is not a codeword → ~~check 1~~
- If the vector is false → many different locations from the correct answer
  - By check 2, columns are as committed
  - Probability of passing check 3 is small

Ligero [AHIV'2017] and [BCGGHJ'2017]

- Ligero [AHIV'2017] : Interleaved test. Reed-Solomon code
- [BCGGHJ'2017] : Ideal linear commitment model. Linear-time encodable code → first SNARK with linear prover time

In the formal proof [AHIV'2017]

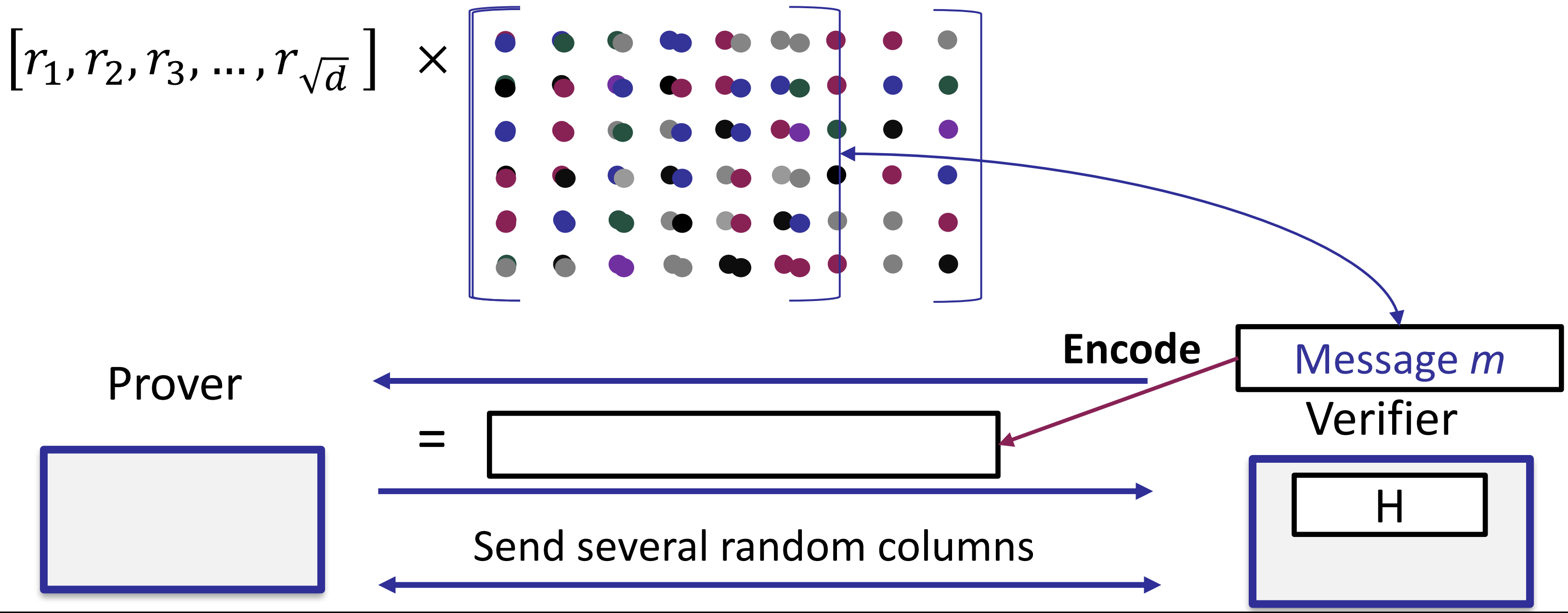
If the committed matrix C is e-far from any codeword for $e < \frac{\Delta}{4}$

$\rightarrow \Pr[w = r^T C$ is e-close to any codeword$] \leq \frac{e+1}{\mathbb{F}}$

If $w = r^T C$ is e-far from any codeword

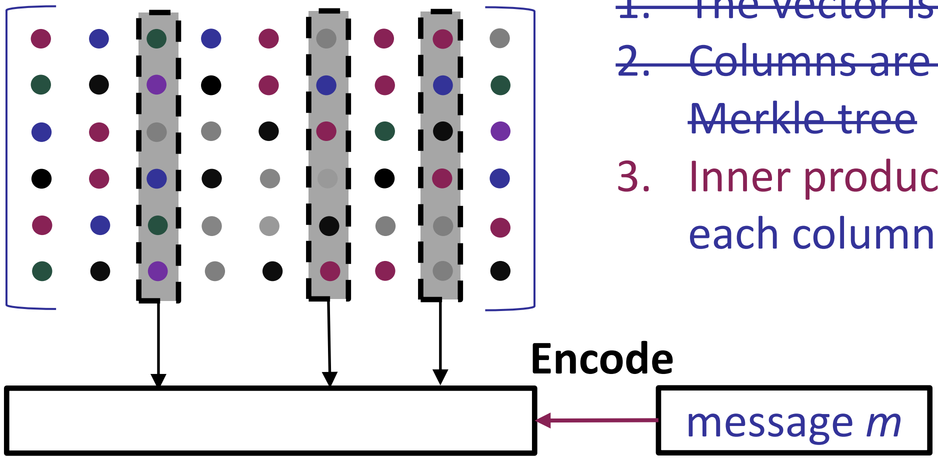
$\rightarrow \Pr[$check 3 is true for t random columns$] \leq \left(1 - \frac{e}{n}\right)^t$

One optimization



Step 2: Consistency check

$\left[1, u, u^2, \ldots, u^{\sqrt{d}-1}\right] \times$



Send several random columns

1. ~~The vector is a codeword~~
2. ~~Columns are as committed in Merkle tree~~
3. Inner product between $\vec{u}$ and each column is consistent

Prover

Verifier

H

Soundness (intuition)

- By the proximity test, the committed matrix C is close to a codeword
- There exists an extractor that extracts F by Merkle tree commitment and decoding C, s.t. $\vec{u} \times F = m$ with probability $1 - \epsilon$

Poly-commit based on linear code

- Keygen: sample a hash function
- Commit: encode the coefficient matrix of f row-wise with a linear code, compute the Merkle tree commitment
- Eval and Verify:
  - Proximity test: random linear combination of all rows, check its consistency with t random columns
  - Consistency test: $\vec{u} \times F = m$, encode m and check its consistency with t random columns
  - $f(u) = \langle m, \vec{u}' \rangle$

Properties of the polynomial commitment

- Keygen: O(1), transparent setup!
- Commit:
  - Encoding: O(d logd) field multiplications using RS code, O(d) using linear-time encodable code
  - Merkle tree: O(d) hashes, O(1) commitment size
- Eval: O(d) field multiplications

  (non-interactive via Fiat Shamir)
- Proof size: O($\sqrt{d}$)
- Verifier time: O($\sqrt{d}$)

Performance the poly-commit [GLSTW'21]

degree $d = 2^{25}$, linear-time encodable code

- Commit: 36s
- Eval: 3.2s
- Proof size: 49MB
- Verifier time: 0.7s

[Bootle-Chiesa-Groth'20] and Brakedown [GLSTW'21]

- [Bootle-Chiesa-Groth'20]: Tensor query IOP $\langle f, \ (\vec{u} \otimes \vec{u}')\rangle$
  - Generalizes to multiple dimensions with proof size $O(n^{\epsilon})$ for constant $\epsilon < 1$

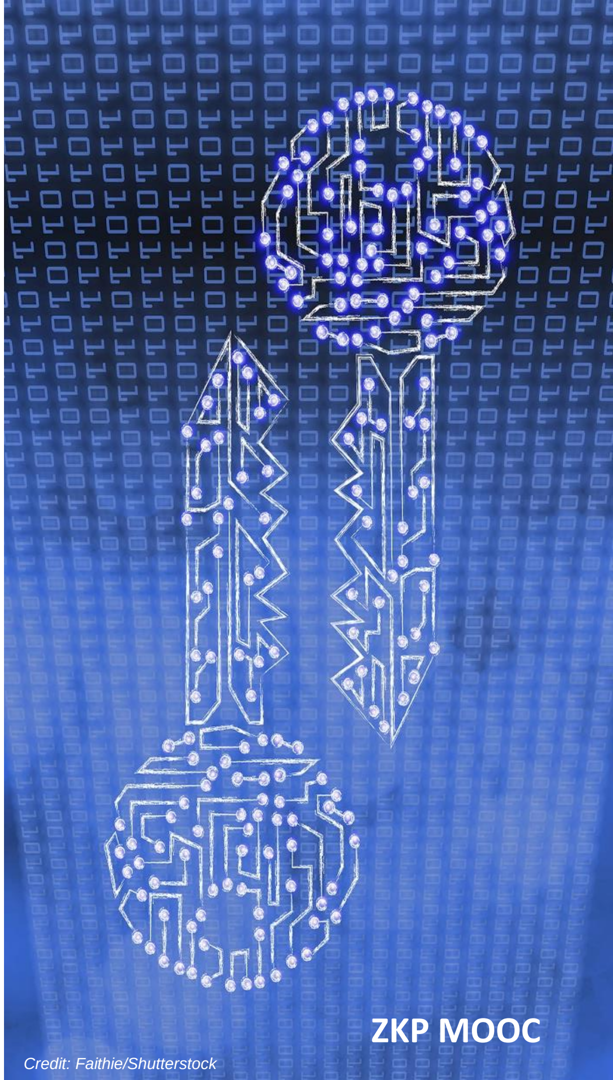
- Brakedown [GLSTW'21]: polynomial commitment based on tensor query
  - Knowledge soundness without efficient decoding algorithm

[Bootle-Chiesa-Liu'21] and Orion [Xie-Zhang-Song'22]

- [Bootle-Chiesa-Liu'21]
  - Proof size polylog(n) with a proof composition of tensor IOP and PCP of proximity [Mie'09]
- Orion [Xie-Zhang-Song'22]
  - Proof size $O(\log^2 n)$ with a proof composition of the code-switching technique [Ron-Zewi-Rothblum'20]

    (5.7MB for $d = 2^{25}$)

Linear-time encodable code



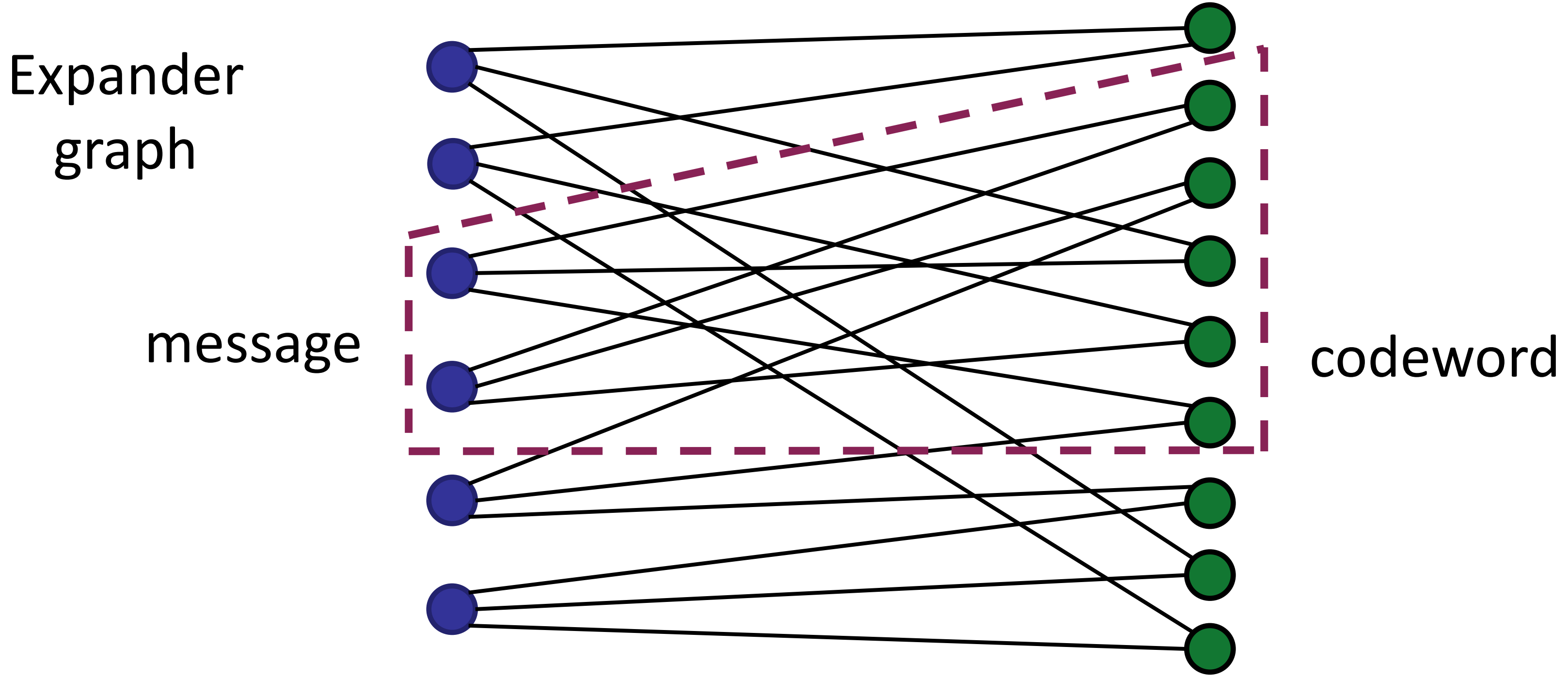
Credit: Faithie/Shutterstock

SNARKs with linear prover time

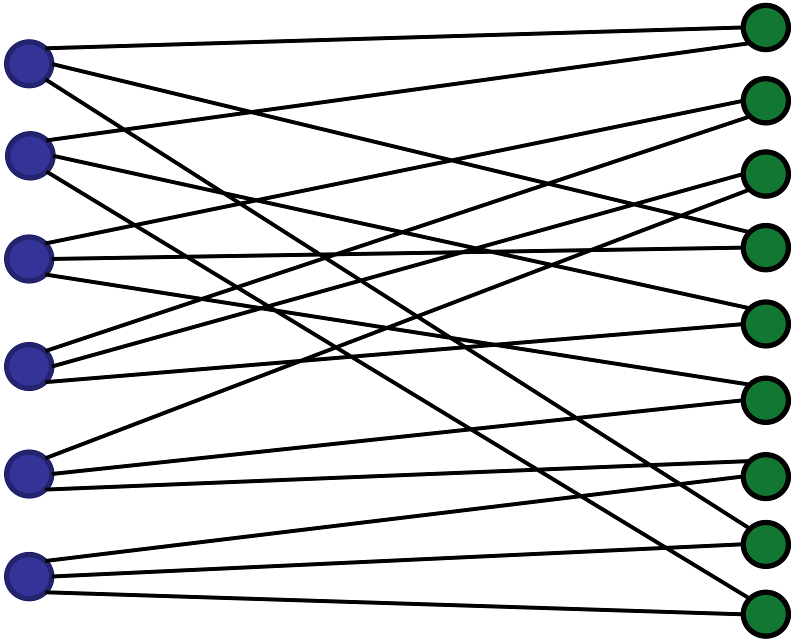
BCGGHJ 2017 → BCG2020 → BCL2021 → GLSTW 2021 → XZS2022

BCGGHJ 2017	BCG2020	BCL2021	GLSTW 2021	XZS2022
Ideal linear model $O(\sqrt{d})$ proof size	Tensor IOP $O(d^{\epsilon})$	Tensor IOP+PCPP $\text{polylog}(d)$	Polynomial commitment $O(d^{\epsilon})$	Code-switching proof composition $O(\log^2 d)$

Linear-time encodable code [Spielman'96][Druk-Ishai'14]



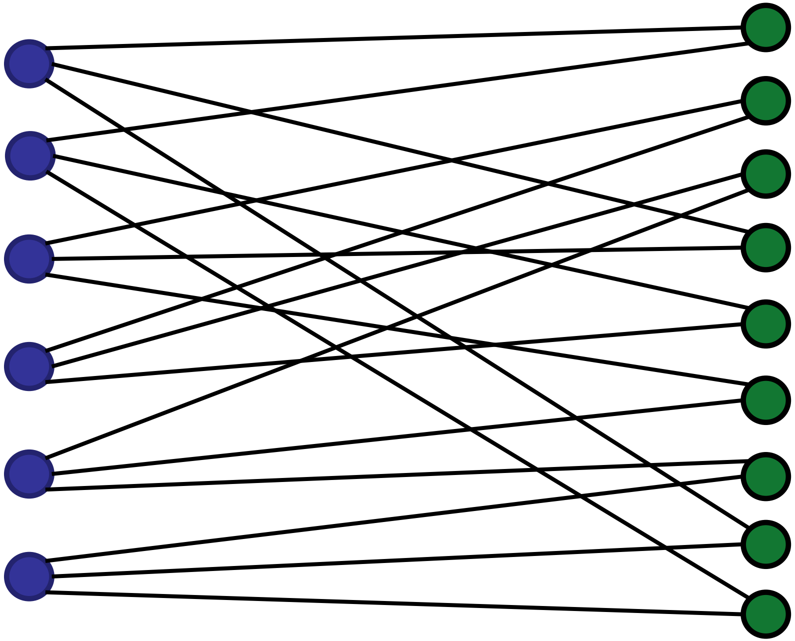
Lossless Expander



- # left nodes = $|L|$, # right nodes = $\alpha|L|$ for a constant α
- Degree of a left node = g
- For every subset S of nodes on the left, # of neighbors $|\Gamma(S)| = g|S|$,

  for $|S| \leq \frac{\alpha|L|}{g}$

Lossless Expander

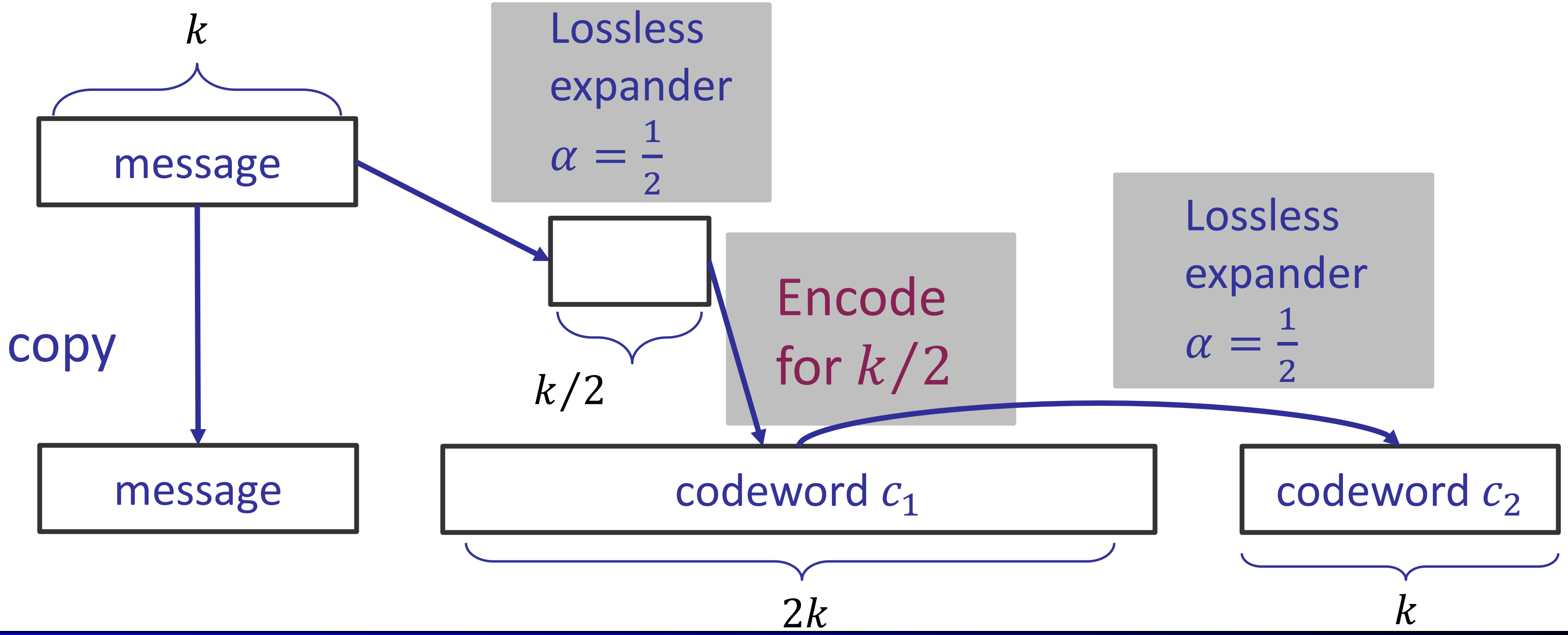


- # left nodes = $|L|$, # right nodes = $\alpha|L|$ for a constant α
- Degree of a left node = g
- For every subset S of nodes on the left, # of neighbors

$|\Gamma(S)| \geq (1-\beta)g|S|$, for $|S| \leq \frac{\delta|L|}{g}$

$(\beta \rightarrow 0, \delta \rightarrow \alpha)$

Overview of the recursive encoding



Encoding algorithm

- Message m of size k, codeword size $4k$, rate is $1/4$
- Suppose there is an encoding algorithm from $k/2$ to $2k$ with good relative distance Δ
- Suppose there are lossless expander graphs of size k and $2k$, and $\alpha = 1/2$

1. Pass m through lossless expander to get m_1 of size $k/2$
2. Encode m_1 to get c_1 of size $2k$
3. Pass c_1 through lossless expander to get c_2 of size k
4. Codeword $c = m||c_1||c_2$

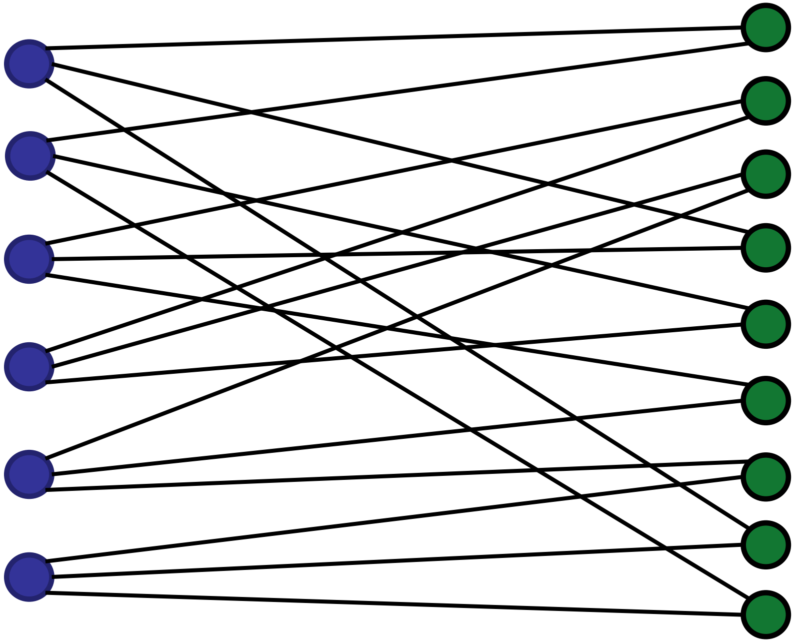
Recursive encoding

- Repeat for $k/2$, $k/4$ … until a constant size
- Use any code with good distance for a constant-size message. E.g., Reed-Solomon code

Distance of the code

constant relative distance $\Delta' = \min\{\Delta, \frac{\delta}{4g}\}$

Lossless Expander



- # left nodes = k, # right nodes = αk for a constant α
- Degree of a left node = g
- For every subset S of nodes on the left, # of neighbors

$|\Gamma(S)| \geq (1-\beta)g|S|$, for $|S| \leq \frac{\delta|L|}{g}$

$(\beta \rightarrow 0, \delta \rightarrow \alpha)$

Proof of constant relative distance [Druk-Ishai'14]

constant relative distance $\Delta' = \min\{\Delta, \frac{\delta}{4g}\}$, codeword $c = m||c_1||c_2$

1. If weight of m is larger than $4k\Delta'$ → done
2. If (weight of m) $\leq 4k\Delta'$, the condition of lossless expander holds
   - Let S be the set of nonzero nodes, $|\Gamma(S)| \geq (1-\beta)g|S|$
   - At least 1 node in $|\Gamma(S)|$ have a unique neighbor in S
   - m_1 is nonzero → (weight of c_1) $\geq 2k\Delta$
3. If it is larger than $4k\Delta'$ → done
4. Else, weight of $c_2 \geq 2k\Delta'$ because of lossless expander

Sampling of the lossless expander

- [Capalbo-Reingold-Vadhan-Wigderson’2002]: Explicit construction of lossless expander (large hidden constant)

- Random sampling: 1/poly(n) failure probability

Improvements of the code

- Brakedown [Golovnev-Lee-Setty-Thaler-Wahby'21]: random summations with better concrete distance analysis

- Orion [Xie-Zhang-Song'22]: expander testing with a negligible failure probability via maximum density of the graph

Putting everything together

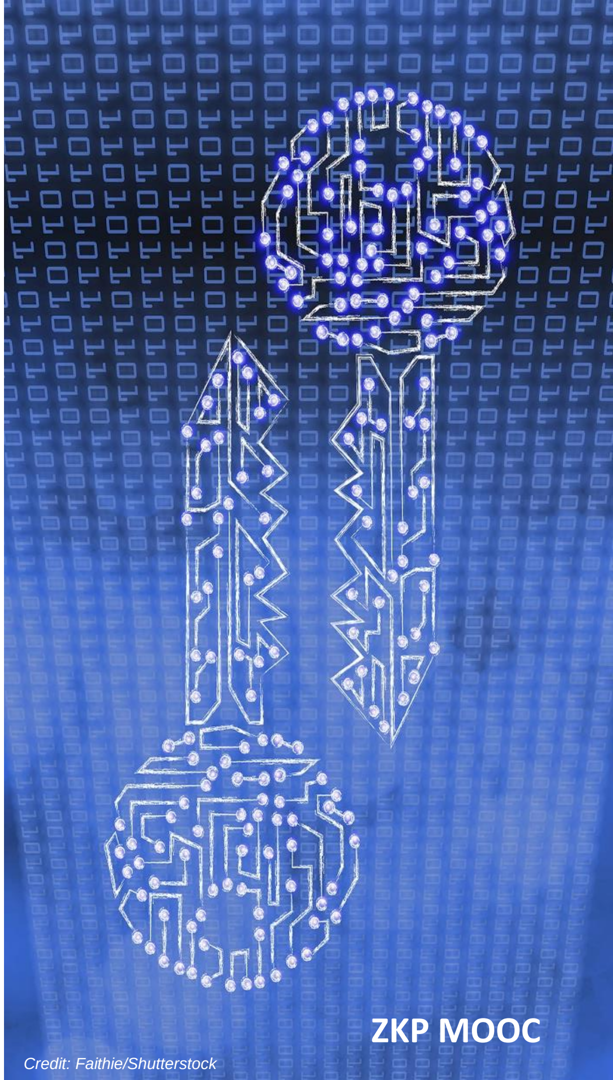
Polynomial commitment (and SNARK) based on linear code

- ✓ Transparent setup: $O(1)$
- ✓ Commit and Prover time: $O(d)$ field additions and multiplications
- ✓ Plausibly post-quantum secure
- ✓ Field agnostic

- ✗ Proof size: $O(\sqrt{d})$, MBs

End of Lecture

Next: FRI and Stark



Credit: Faithie/Shutterstock