Hey! You Can't Do That With My Code!

William J. Martin

Department of Mathematical Sciences
and
Department of Computer Science
Worcester Polytechnic Institute

Twelfth Rocky Mountain Discrete Mathematics Day(s)
Denver CO, June 20, 2009

Outline

A New Era in Coding Theory

Is algebraic coding theory dead?

A New Era in Coding Theory

Is algebraic coding theory dead?

New Efficient Error-Correcting Codes

- turbo codes
- belief propagation
- low-density parity check codes
- proof by simulation (now made rigorous)
- large minimum distance no longer needed

Of Course Codes are Still Fundamental

- Coding Theory
- Design Theory

Of Course Codes are Still Fundamental

- Coding Theory
- *"Distinguishability"*

- Design Theory
- *"Approximation"*

Of Course Codes are Still Fundamental

- Coding Theory
- *"Distinguishability"*
- E.g., linear codes in Hamming scheme $H(n, q)$

- Design Theory
- *"Approximation"*
- E.g, orthogonal arrays of strength t

A Hint of Delsarte Theory

- Coding Theory
- *"Distinguishability"*
- E.g., linear codes in Hamming scheme $H(n, q)$

- Design Theory
- *"Approximation"*
- E.g, orthogonal arrays of strength t

Theorem: The dual of any additive code with minimum distance d is an OA of strength $t = d - 1$.

An Apology

- Perhaps the most exciting development in algebraic coding theory since 1990 is the theory of **quantum error-correcting codes**

An Apology

- Perhaps the most exciting development in algebraic coding theory since 1990 is the theory of **quantum error-correcting codes**
- This is **not** what I want to talk about today.

Part I: (T, M, S)-Nets

Using Codes to Estimate Integrals

If orthogonal arrays can be used to approximate Hamming space, can they also be used to approximate other spaces?

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/ Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)
- **1996:** ordered orthogonal arrays (Mullen/Schmid)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)
- **1996:** ordered orthogonal arrays (Mullen/Schmid)
- **1996:** Constructions from algebraic curves (Niederreiter/Xing)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)
- **1996:** ordered orthogonal arrays (Mullen/Schmid)
- **1996:** Constructions from algebraic curves (Niederreiter/Xing)
- **1999:** MacWilliams identities, LP bounds, association scheme (WJM/Stinson)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)
- **1996:** ordered orthogonal arrays (Mullen/Schmid)
- **1996:** Constructions from algebraic curves (Niederreiter/Xing)
- **1999:** MacWilliams identities, LP bounds, association scheme (WJM/Stinson)
- **late 90s+:** Many new constructions (Adams/Edel/Bierbrauer/et al.)

Key Results

- **1967:** Sobol' sequences (I. Sobol') [also Halton/Faure/Hammersley sequences]
- **1987:** (T, M, S)-nets (Niederreiter)
- **1996:** generalized orthogonal arrays (Lawrence)
- **1996:** ordered orthogonal arrays (Mullen/Schmid)
- **1996:** Constructions from algebraic curves (Niederreiter/Xing)
- **1999:** MacWilliams identities, LP bounds, association scheme (WJM/Stinson)
- **late 90s+:** Many new constructions (Adams/Edel/Bierbrauer/et al.)
- **2004+:** Improved bounds (Schmid/Schürer/Bierbrauer/Barg/Purkayastha/Trinker/Visentin)

What is a (T, M, S)-Net?

Harald Niederrieter

A (T, M, S)-*net in base* q

What is a (T, M, S)-Net?

Harald Niederrieter

A (T, M, S)-*net in base* q is a set $\mathcal{N}$ of q^M points in the half-open S-dimensional Euclidean cube $[0, 1)^S$

What is a (T, M, S)-Net?

Harald Niederrieter

A (T, M, S)-*net in base* q is a set $\mathcal{N}$ of q^M points in the half-open S-dimensional Euclidean cube $[0, 1)^S$ with the property that every *elementary interval*

$$\left[\frac{a_1}{q^{d_1}}, \frac{a_1+1}{q^{d_1}}\right) \times \left[\frac{a_2}{q^{d_2}}, \frac{a_2+1}{q^{d_2}}\right) \times \cdots \times \left[\frac{a_S}{q^{d_S}}, \frac{a_S+1}{q^{d_S}}\right)$$

of volume q^{T-M}

What is a (T, M, S)-Net?

Harald Niederrieter

A (T, M, S)-*net in base* q is a set $\mathcal{N}$ of q^M points in the half-open S-dimensional Euclidean cube $[0, 1)^S$ with the property that every *elementary interval*

$$\left[\frac{a_1}{q^{d_1}}, \frac{a_1+1}{q^{d_1}}\right) \times \left[\frac{a_2}{q^{d_2}}, \frac{a_2+1}{q^{d_2}}\right) \times \cdots \times \left[\frac{a_S}{q^{d_S}}, \frac{a_S+1}{q^{d_S}}\right)$$

of volume q^{T-M} (i.e., with $d_1 + d_2 + \cdots + d_S = M - T$)

What is a (T, M, S)-Net?

Harald Niederrieter

A (T, M, S)-*net in base* q is a set $\mathcal{N}$ of q^M points in the half-open S-dimensional Euclidean cube $[0, 1)^S$ with the property that every *elementary interval*

$$\left[\frac{a_1}{q^{d_1}}, \frac{a_1+1}{q^{d_1}}\right) \times \left[\frac{a_2}{q^{d_2}}, \frac{a_2+1}{q^{d_2}}\right) \times \cdots \times \left[\frac{a_S}{q^{d_S}}, \frac{a_S+1}{q^{d_S}}\right)$$

of volume q^{T-M} (i.e., with $d_1 + d_2 + \cdots + d_S = M - T$) contains exactly q^T points from $\mathcal{N}$.

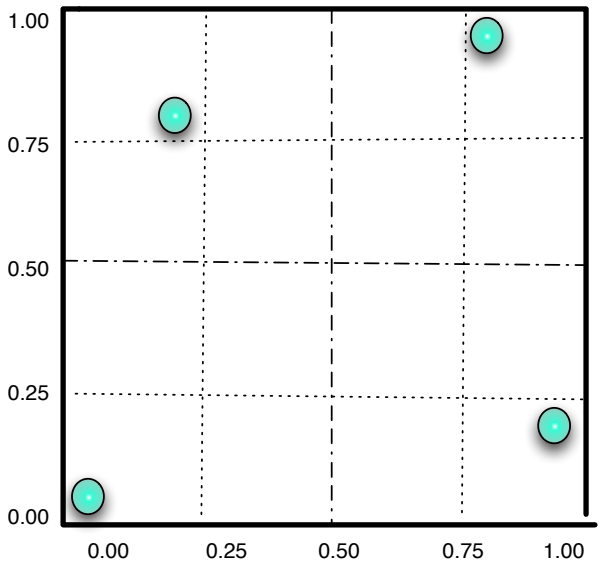
Simple Example of a (T, M, S)-Net

- binary code with minimum distance three
- four points in $[0, 1)^2$

Simple Example of a (T, M, S)-Net

- binary code with minimum distance three
- $C = \{000000,\ 111001,\ 001110,\ 110111\}$
- four points in $[0, 1)^2$
- $\mathcal{N} = \{(0, 0),\ (7/8, 1/8),\ (1/8, 3/4),\ (3/4, 7/8)\}$

Simple Example of a (T, M, S)-Net

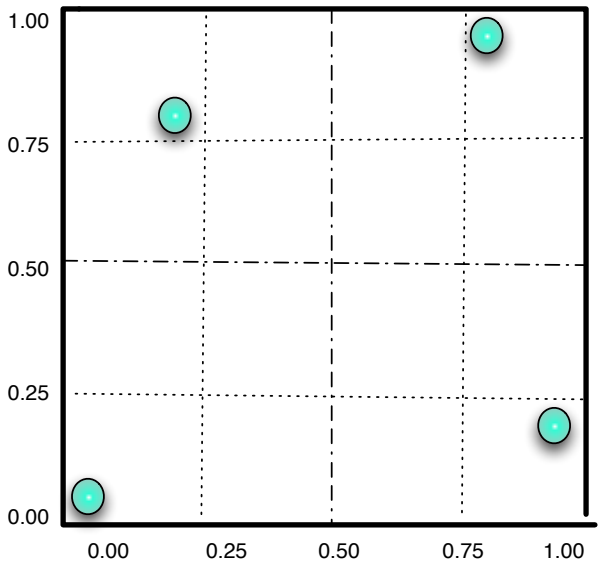
- binary code with minimum distance three
- $C = \{000000,\ 111001,\ 001110,\ 110111\}$
- partition into two groups of three coords, insert decimal points
- four points in $[0, 1)^2$
- $\mathcal{N} = \{(0, 0),\ (7/8, 1/8),\ (1/8, 3/4),\ (3/4, 7/8)\}$
- 

Simple Example of a (T, M, S)-Net

- binary code with minimum distance three
- $C = \{000000,\ 111001,\ 001110,\ 110111\}$
- partition into two groups of three coords, insert decimal points
-

0	0	0	0	0	0
1	1	1	0	0	1
0	0	1	1	1	0
1	1	0	1	1	1

- four points in $[0, 1)^2$
- $\mathcal{N} = \{(0, 0),\ (7/8, 1/8),\ (1/8, 3/4),\ (3/4, 7/8)\}$
-

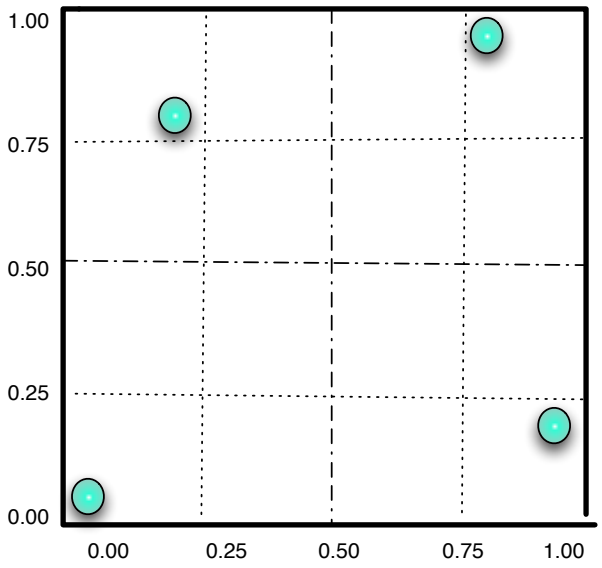


Simple Example of a (T, M, S)-Net

- binary code with minimum distance three
- $C = \{000000,\ 111001,\ 001110,\ 110111\}$
- partition into two groups of three coords, insert decimal points
-

.0	0	0	.0	0	0
.1	1	1	.0	0	1
.0	0	1	.1	1	0
.1	1	0	.1	1	1

- four points in $[0, 1)^2$
- $\mathcal{N} = \{(0, 0),\ (7/8, 1/8),\ (1/8, 3/4),\ (3/4, 7/8)\}$
-



Orthogonal Array Property

- We consider an $m \times n$ array A over $\mathbb{F}_q$

Orthogonal Array Property

- We consider an $m \times n$ array A over $\mathbb{F}_q$
- **"OA property":** for a subset T of the columns, does the projection of A onto these columns contain every $|T|$-tuple over $\mathbb{F}_q$ equally often?

Orthogonal Array Property

- We consider an $m \times n$ array A over $\mathbb{F}_q$
- **"OA property":** for a subset T of the columns, does the projection of A onto these columns contain every $|T|$-tuple over $\mathbb{F}_q$ equally often?
- **orthogonal array of strength** t: A has the OA property with respect to any set T of t or fewer columns

Orthogonal Array Property

- We consider an $m \times n$ array A over $\mathbb{F}_q$
- **"OA property":** for a subset T of the columns, does the projection of A onto these columns contain every $|T|$-tuple over $\mathbb{F}_q$ equally often?
- **orthogonal array of strength** t: A has the OA property with respect to any set T of t or fewer columns
- **ordered orthogonal array:** Now assume $n = s\ell$ and columns are labelled $\{(i, j) : 1 \leq i \leq s, 1 \leq j \leq \ell\}$.

Ordered Orthogonal Arrays

- **"OA property"** with respect to column set T: projection of A onto these columns contains every $|T|$-tuple over $\mathbb{F}_q$ equally often

Ordered Orthogonal Arrays

- **"OA property"** with respect to column set T: projection of A onto these columns contains every $|T|$-tuple over $\mathbb{F}_q$ equally often
- **ordered orthogonal array:** Now assume $n = s\ell$ and columns are labelled $\{(i, j) : 1 \leq i \leq s, 1 \leq j \leq \ell\}$

Ordered Orthogonal Arrays

- **"OA property"** with respect to column set T: projection of A onto these columns contains every $|T|$-tuple over $\mathbb{F}_q$ equally often
- **ordered orthogonal array:** Now assume $n = s\ell$ and columns are labelled $\{(i, j) : 1 \leq i \leq s, 1 \leq j \leq \ell\}$
- a set T of columns is *"left-justified"* if it contains $(i, j-1)$ whenever it contains (i, j) with $j > 1$

Ordered Orthogonal Arrays

- **"OA property"** with respect to column set T: projection of A onto these columns contains every $|T|$-tuple over $\mathbb{F}_q$ equally often
- **ordered orthogonal array:** Now assume $n = s\ell$ and columns are labelled $\{(i,j) : 1 \leq i \leq s, 1 \leq j \leq \ell\}$
- a set T of columns is *"left-justified"* if it contains $(i, j-1)$ whenever it contains (i,j) with $j > 1$
- **ordered orthogonal array of strength** t: A enjoys the OA property for every left-justified set of t or fewer columns

Ordered Orthogonal Arrays

- **"OA property"** with respect to column set T: projection of A onto these columns contains every $|T|$-tuple over $\mathbb{F}_q$ equally often
- **ordered orthogonal array:** Now assume $n = s\ell$ and columns are labelled $\{(i,j) : 1 \leq i \leq s, 1 \leq j \leq \ell\}$
- a set T of columns is *"left-justified"* if it contains $(i, j-1)$ whenever it contains (i,j) with $j > 1$
- **ordered orthogonal array of strength** t: A enjoys the OA property for every left-justified set of t or fewer columns
- **Lawrence/Mullen/Schmid:** $\exists (T, M, S)$-net in base $q \Leftrightarrow \exists OOA$ over $\mathbb{F}_q$ with q^m rows, $s = S$, $\ell = t = M - T$.

The Theorem of Mullen & Schmid and (indep.) Lawrence

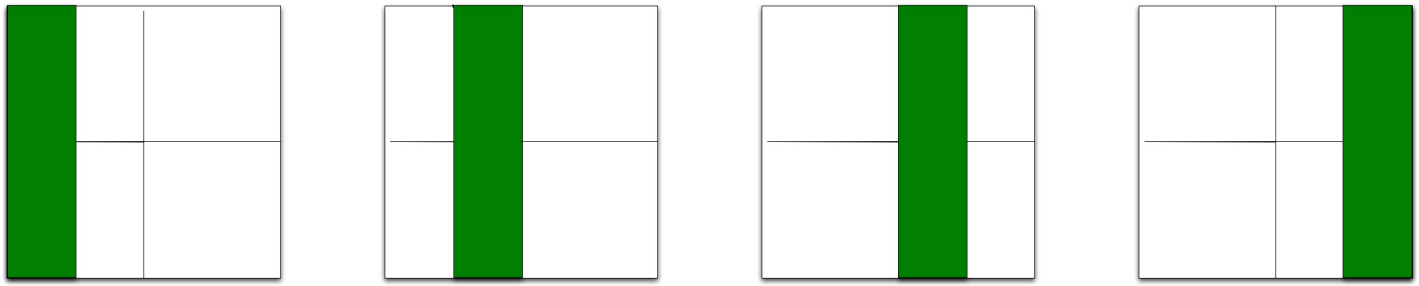


Theorem (1996): $\exists (T, M, S)$-net in base $q \Leftrightarrow \exists OOA$ over $\mathbb{F}_q$ with q^m rows, $s = S$, $\ell = t = M - T$

Idea of Proof

$$\mathcal{N} = \left\{\left(\frac{0}{4}, \frac{0}{4}\right), \left(\frac{1}{4}, \frac{3}{4}\right), \left(\frac{2}{4}, \frac{2}{4}\right), \left(\frac{3}{4}, \frac{1}{4}\right)\right\}$$

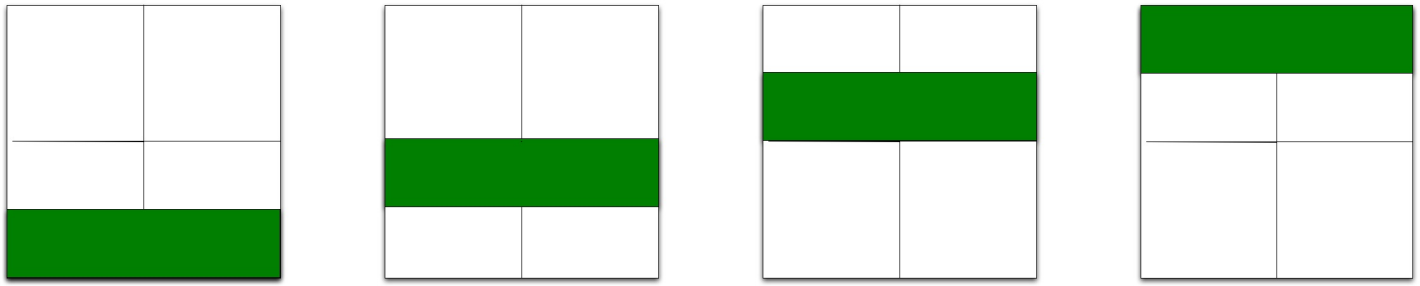
$$T = \{(1,1), (1,2)\}$$



Idea of Proof

$$\mathcal{N} = \{(.00, .00), (.01, .11), (.10, .10), (.11, .01)\}$$

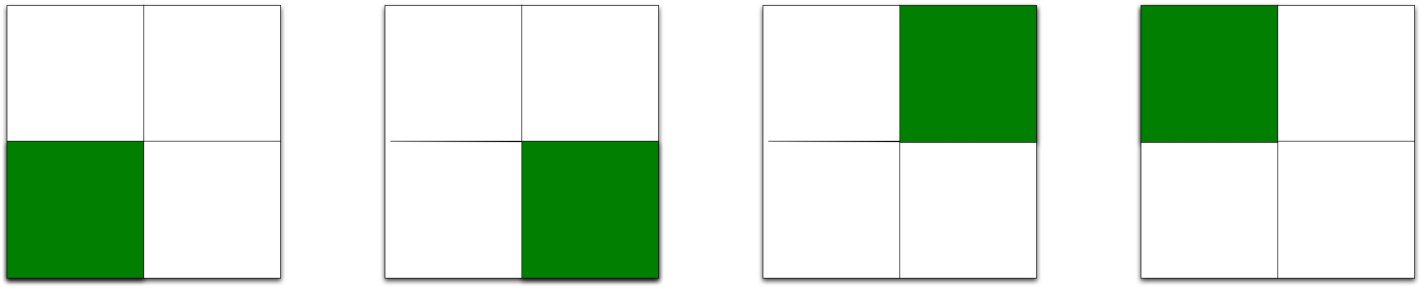
$$T = \{(2, 1), (2, 2)\}$$



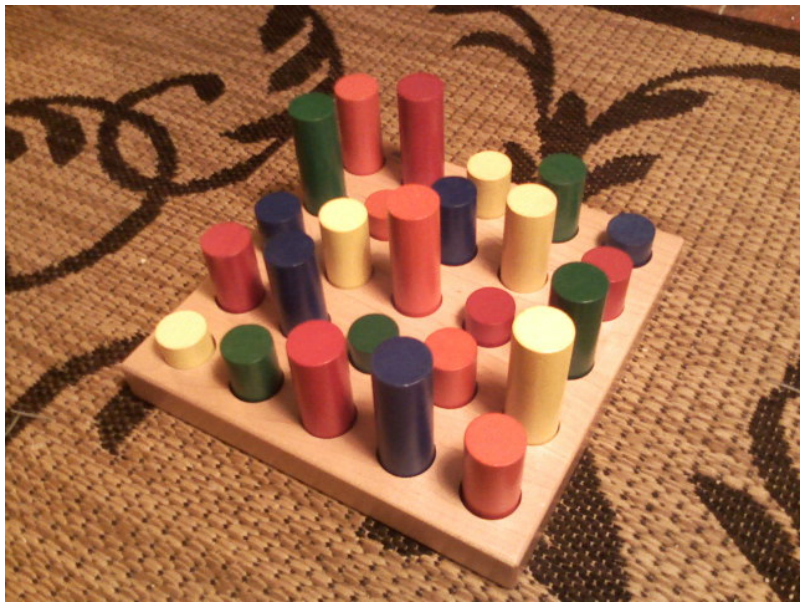
Idea of Proof

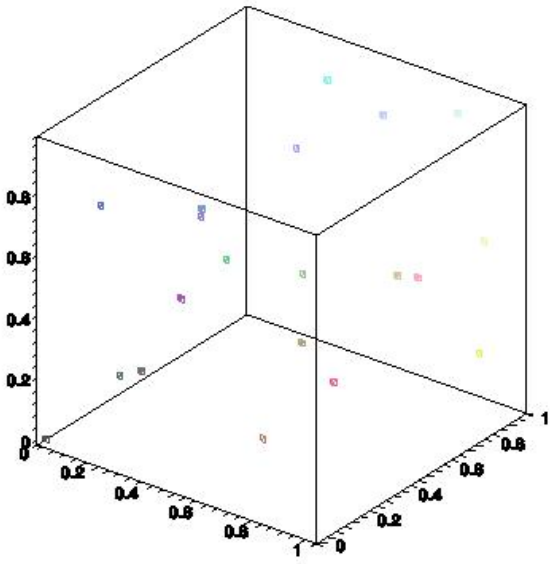
$$\mathcal{N} = \{(.00, .00), (.01, .11), (.10, .10), (.11, .01)\}$$

$$T = \{(1, 1), (2, 1)\}$$



Nets from Many Sources





two mutually orthogonal latin squares of order five (color/height)

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$
- Reed-Solomon code has a codeword for each polynomial $f(x)$ of degree $\leq k$:

$$c_f = [f(P_1), f(P_2), \ldots, f(P_s)]$$

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$
- Reed-Solomon code has a codeword for each polynomial $f(x)$ of degree $\leq k$:

$$c_f = [f(P_1), f(P_2), \ldots, f(P_s)]$$

- a non-zero polynomial of degree at most k has at most k roots

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$
- Reed-Solomon code has a codeword for each polynomial $f(x)$ of degree $\leq k$:

$$c_f = [f(P_1), f(P_2), \ldots, f(P_s)]$$

- a non-zero polynomial of degree at most k has at most k roots
- . . . counting multiplicities!

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$
- Reed-Solomon code has a codeword for each polynomial $f(x)$ of degree $\leq k$:

$$c_f = [f(P_1), f(P_2), \ldots, f(P_s)]$$

- a non-zero polynomial of degree at most k has at most k roots
- . . . counting multiplicities!
- So take SM-tuple $(M = k + 1)$

$$\left[f(P_1), f'(P_1), \ldots, f^{(k)}(P_1) | \ldots\ldots | f(P_s), f'(P_s), \ldots, f^{(k)}(P_s)\right]$$

to get a powerful (T, M, S)-net

Niederreiter/Xing Construction (Simplified)

- Let $N = \{P_1, \ldots, P_s\}$ be a subset of $\mathbb{F}_q$ of size s, let $k \geq 0$
- Reed-Solomon code has a codeword for each polynomial $f(x)$ of degree $\leq k$:

$$c_f = [f(P_1), f(P_2), \ldots, f(P_s)]$$

- a non-zero polynomial of degree at most k has at most k roots
- . . . counting multiplicities!
- So take SM-tuple $(M = k + 1)$

$$\left[f(P_1), f'(P_1), \ldots, f^{(k)}(P_1) | \ldots\ldots | f(P_s), f'(P_s), \ldots, f^{(k)}(P_s)\right]$$

to get a powerful (T, M, S)-net

- They show that the same works over algebraic curves (global function fields)

Codes for the Rosenbloom-Tsfasman Metric

- the dual of a linear OA is an error-correcting code

Codes for the Rosenbloom-Tsfasman Metric

- the dual of a linear OA is an error-correcting code
- the dual of a linear OOA is a code for the Rosenbloom-Tsfasman metric

Codes for the Rosenbloom-Tsfasman Metric

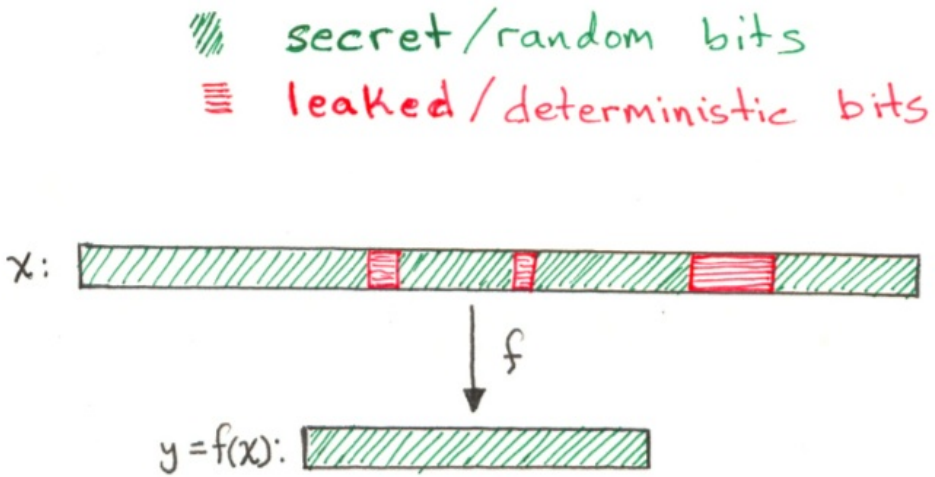
- the dual of a linear OA is an error-correcting code
- the dual of a linear OOA is a code for the Rosenbloom-Tsfasman metric
- **Research Problem:** Are there any non-trivial perfect codes in the Rosenbloom-Tsfasman metric?

Part II: Resilient Functions

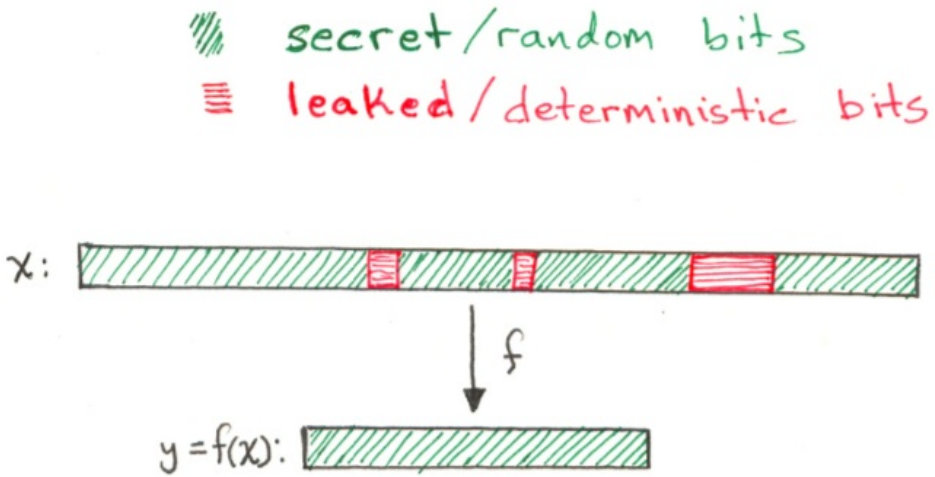
Resilient Functions

How can a code be used to bolster randomness?

Resilient Functions



Resilient Functions



We have a secret string x. An opponent learns t bits of x, but we don't know which ones.
After applying function f, we guarantee that our opponents knows nothing.

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)
- **1988:** Privacy amplification by public discussion (Bennett/Brassard/Robert)

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)
- **1988:** Privacy amplification by public discussion (Bennett/Brassard/Robert)
- **1993:** Equivalent to large set of OA (Stinson)

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)
- **1988:** Privacy amplification by public discussion (Bennett/Brassard/Robert)
- **1993:** Equivalent to large set of OA (Stinson)
- **1995:** First non-linear examples (Stinson/Massey)

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)
- **1988:** Privacy amplification by public discussion (Bennett/Brassard/Robert)
- **1993:** Equivalent to large set of OA (Stinson)
- **1995:** First non-linear examples (Stinson/Massey)
- **1997:** All-or-nothing transforms (Rivest)

Key Results

- **1985:** The bit extraction problem (Chor/Goldreich/Håstad/Friedman/Rudich/Smolensky)
- **1988:** Privacy amplification by public discussion (Bennett/Brassard/Robert)
- **1993:** Equivalent to large set of OA (Stinson)
- **1995:** First non-linear examples (Stinson/Massey)
- **1997:** All-or-nothing transforms (Rivest)
- **1999+:** Applications to fault-tolerant distributed computing, key distribution, quantum cryptography, etc.

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code
- Define $f : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^k$ via

$$f(x) = Gx$$

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code
- Define $f : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^k$ via

$$f(x) = Gx$$

- If $t \leq d - 1$ entries of x are deterministic and the rest are random and fully independent (denote $\mathcal{D}_{T,A}$)

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code
- Define $f : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^k$ via

$$f(x) = Gx$$

- If $t \leq d - 1$ entries of x are deterministic and the rest are random and fully independent (denote $\mathcal{D}_{T,A}$)
- . . . then $f(x)$ is uniformly distributed over $\mathbb{F}_q^k$

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code
- Define $f : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^k$ via

$$f(x) = Gx$$

- If $t \leq d - 1$ entries of x are deterministic and the rest are random and fully independent (denote $\mathcal{D}_{T,A}$)
- . . . then $f(x)$ is uniformly distributed over $\mathbb{F}_q^k$
- **Why?** Any linear combination of entries of $f(x)$ is a dot product of x with some codeword

The Linear Case (Chor, et al.)

- Let G be a generator matrix for an $[n, k, d]_q$-code
- Define $f : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^k$ via

$$f(x) = Gx$$

- If $t \leq d - 1$ entries of x are deterministic and the rest are random and fully independent (denote $\mathcal{D}_{T,A}$)
- . . . then $f(x)$ is uniformly distributed over $\mathbb{F}_q^k$
- **Why?** Any linear combination of entries of $f(x)$ is a dot product of x with some codeword
- So any non-trivial linear function of entries involves at least one random input position

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**
- Device must tap some physical source of known behavior

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**
- Device must tap some physical source of known behavior
- Even the best sources of randomness have "quiet" periods

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**
- Device must tap some physical source of known behavior
- Even the best sources of randomness have "quiet" periods
- Assuming 80% of input bits are random samples and 20% are from quiet periods

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**
- Device must tap some physical source of known behavior
- Even the best sources of randomness have "quiet" periods
- Assuming 80% of input bits are random samples and 20% are from quiet periods
- Resilient function collapses samples to strings one-tenth the size

True Random Bit Generators (Sunar/Stinson/WJM)

- Random bits are **expensive**
- Device must tap some physical source of known behavior
- Even the best sources of randomness have "quiet" periods
- Assuming 80% of input bits are random samples and 20% are from quiet periods
- Resilient function collapses samples to strings one-tenth the size
- What if quiet period is longer than expected?

Higher Weights (Generalized Hamming Weights)

- Start with a binary linear $[n, k, d]$-code

Higher Weights (Generalized Hamming Weights)

- Start with a binary linear $[n, k, d]$-code
- Define $A_h^{(\ell)}$ as number of linear subcodes C', $\dim C' = \ell$, $|\operatorname{supp} C'| = h$

Higher Weights (Generalized Hamming Weights)

- Start with a binary linear $[n, k, d]$-code
- Define $A_h^{(\ell)}$ as number of linear subcodes C', $\dim C' = \ell$, $|\operatorname{supp} C'| = h$
- E.g. $A_h^{(1)} = A_h$ for $h > 0$, $\quad A_h^{(\ell)} = 0$ for $h < d$ except $A_0^{(0)} = 1$

Higher Weights (Generalized Hamming Weights)

- Start with a binary linear $[n, k, d]$-code
- Define $A_h^{(\ell)}$ as number of linear subcodes C', $\dim C' = \ell$, $|\operatorname{supp} C'| = h$
- E.g. $A_h^{(1)} = A_h$ for $h > 0$, $\quad A_h^{(\ell)} = 0$ for $h < d$ except $A_0^{(0)} = 1$
- The number of i-subsets of coordinates that contain the support of exactly 2^r codewords is shown to be

$$B_{i,r} = \sum_{\ell=0}^{k} \sum_{h=0}^{n} (-1)^{\ell-r} 2^{\binom{\ell-r}{2}} \binom{n-h}{i-h} \begin{bmatrix} \ell \\ r \end{bmatrix} A_h^{(\ell)}$$

Higher Weights (Generalized Hamming Weights)

- Start with a binary linear $[n, k, d]$-code
- Define $A_h^{(\ell)}$ as number of linear subcodes C', $\dim C' = \ell$, $|\operatorname{supp} C'| = h$
- E.g. $A_h^{(1)} = A_h$ for $h > 0$, $\quad A_h^{(\ell)} = 0$ for $h < d$ except $A_0^{(0)} = 1$
- The number of i-subsets of coordinates that contain the support of exactly 2^r codewords is shown to be

$$B_{i,r} = \sum_{\ell=0}^{k} \sum_{h=0}^{n} (-1)^{\ell-r} 2^{\binom{\ell-r}{2}} \binom{n-h}{i-h} \begin{bmatrix} \ell \\ r \end{bmatrix} A_h^{(\ell)}$$

- **Lemma** (Sunar/WJM): Let X be a random variable taking values in $\{0,1\}^n$ according to a probability distribution $\mathcal{D}_{T,A}$. Then

$$\text{Prob}[H_{\text{out}} = k - r \mid |T| = i] \;=\; B_{i,r} \binom{n}{i}^{-1}.$$

A Research Problem

Higher weight enumerators are known only for very few codes:

- MDS codes: partial information only (Dougherty, et al.)

A Research Problem

Higher weight enumerators are known only for very few codes:

- MDS codes: partial information only (Dougherty, et al.)
- Golay codes (Sunar/WJM, probably earlier)

A Research Problem

Higher weight enumerators are known only for very few codes:

- MDS codes: partial information only (Dougherty, et al.)
- Golay codes (Sunar/WJM, probably earlier)
- Hamming codes

Can we work out these statistics for the other standard families of codes?

Part III: Fuzzy Extractors

Codes for Biometrics

How can we eliminate noise if we are not permitted to choose our codewords?

Selected References

- **1990s:** Ad-hoc mix of protocols (e.g., quantum oblivious transfer, crypto over noisy channels)

Selected References

- **1990s:** Ad-hoc mix of protocols (e.g., quantum oblivious transfer, crypto over noisy channels)
- **1987,1994:** Patents for iris recognition systems

Selected References

- **1990s:** Ad-hoc mix of protocols (e.g., quantum oblivious transfer, crypto over noisy channels)
- **1987,1994:** Patents for iris recognition systems
- **2008:** definition of "fuzzy extractor" (Dodis/Ostrovsky/Reyzin/Smith)

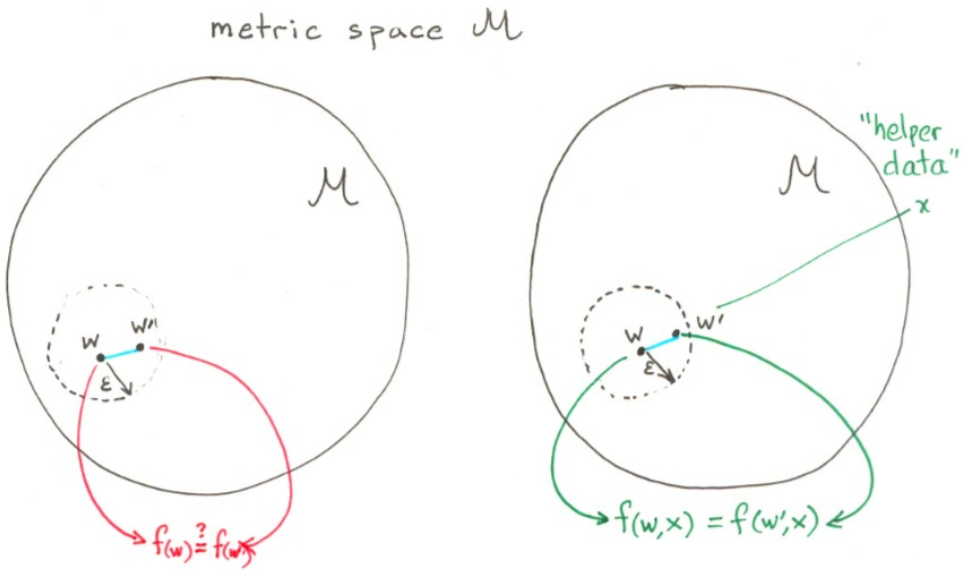
Selected References

- **1990s:** Ad-hoc mix of protocols (e.g., quantum oblivious transfer, crypto over noisy channels)
- **1987,1994:** Patents for iris recognition systems
- **2008:** definition of "fuzzy extractor" (Dodis/Ostrovsky/Reyzin/Smith)
- **2009:** CD fingerprinting (Hammouri/Dana/Sunar)

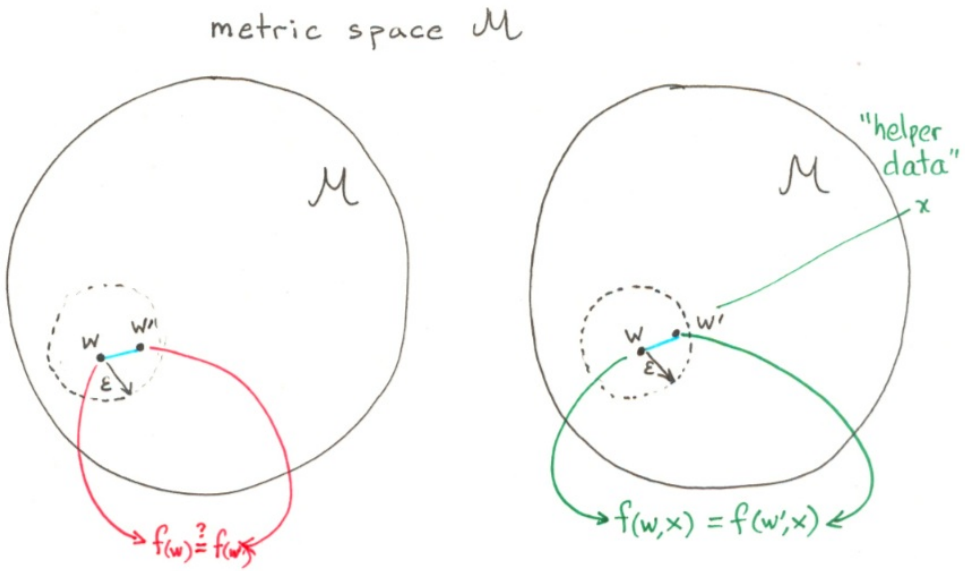
Selected References

- **1990s:** Ad-hoc mix of protocols (e.g., quantum oblivious transfer, crypto over noisy channels)
- **1987,1994:** Patents for iris recognition systems
- **2008:** definition of "fuzzy extractor" (Dodis/Ostrovsky/Reyzin/Smith)
- **2009:** CD fingerprinting (Hammouri/Dana/Sunar)
- **2009:** physically unclonable functions (WPI team)

Fuzzy Extractors

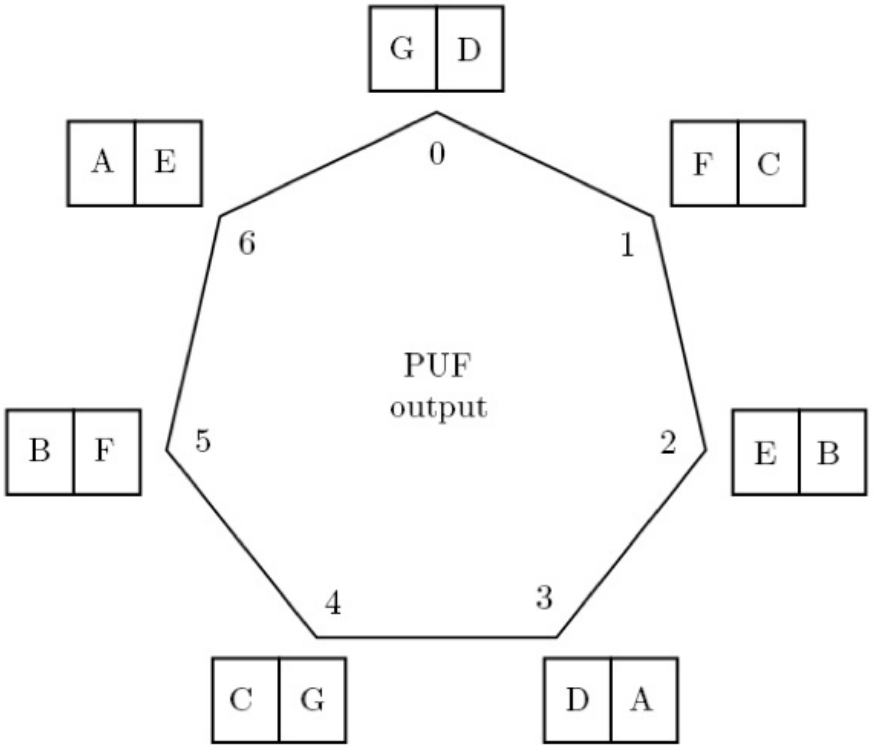


Fuzzy Extractors

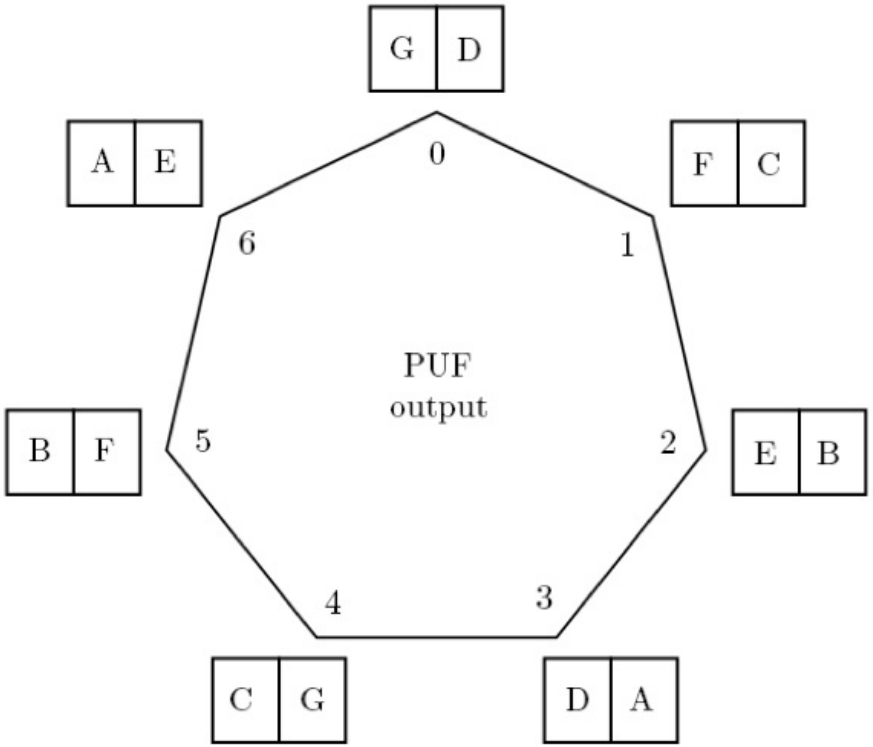


Metric space $\mathcal{M}$ and function $f : \mathcal{M} \times \{0,1\}^* \to \{0,1\}^*$ such that $f(w', x) = f(w, x)$ provided x valid for w and $d(w', w) < \epsilon$.

Fuzzy Extractor: Toy Example



Fuzzy Extractor: Toy Example



Baseline reading $w = 3$ is obtained from temporal reading $w' = 2$ and hint $x = D$.

But w is not recoverable from either w' or x alone.

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G
- For each user, generate a random k-bit string m

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G
- For each user, generate a random k-bit string m
- For baseline reading w, helper data is $x = w + mG$

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G
- For each user, generate a random k-bit string m
- For baseline reading w, helper data is $x = w + mG$
- New reading w' is assumed to be within distance $d/2$ of w in large Hamming space

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G
- For each user, generate a random k-bit string m
- For baseline reading w, helper data is $x = w + mG$
- New reading w' is assumed to be within distance $d/2$ of w in large Hamming space
- To recover m from x and w', decode $w' + x = mG + (w - w')$

Code-Offset Construction (Dodis, et al.)

Fuzzy extractor for Hamming metric:

- Start with a binary $[n, k, d]$-code with generator matrix G
- For each user, generate a random k-bit string m
- For baseline reading w, helper data is $x = w + mG$
- New reading w' is assumed to be within distance $d/2$ of w in large Hamming space
- To recover m from x and w', decode $w' + x = mG + (w - w')$
- Provided k and d are both linear in n, recovery of m from just x or w' is hard

A Research Problem

Fuzzy extractors are known for several metrics:

- Hamming

A Research Problem

Fuzzy extractors are known for several metrics:

- Hamming
- Set difference (fuzzy vault scheme of Juels/Sudan)

A Research Problem

Fuzzy extractors are known for several metrics:

- Hamming
- Set difference (fuzzy vault scheme of Juels/Sudan)
- Edit distance

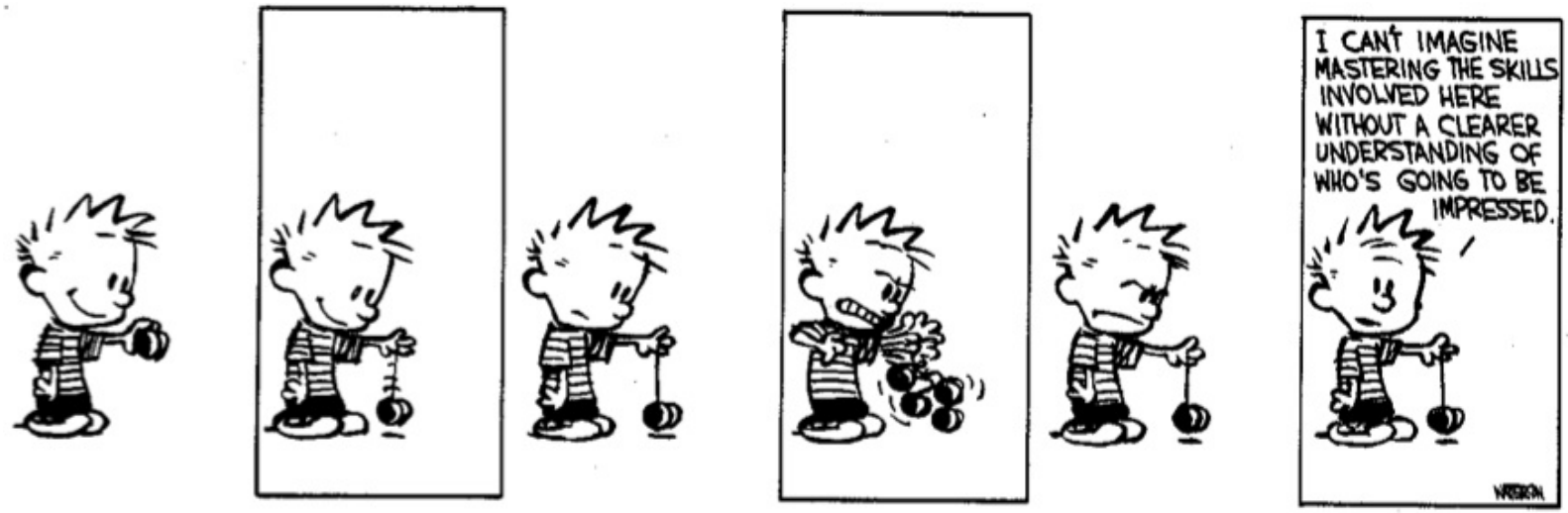
A Research Problem

Fuzzy extractors are known for several metrics:

- Hamming
- Set difference (fuzzy vault scheme of Juels/Sudan)
- Edit distance

Can we build efficient fuzzy extractors for the Euclidean metric?

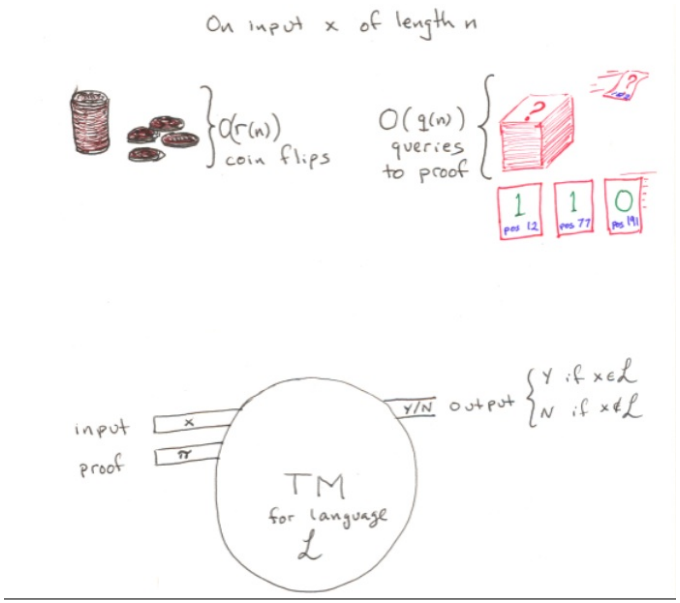
Part IV: The PCP Theorem



Codes and Computational Complexity

How does coding theory make computational problems more robust?

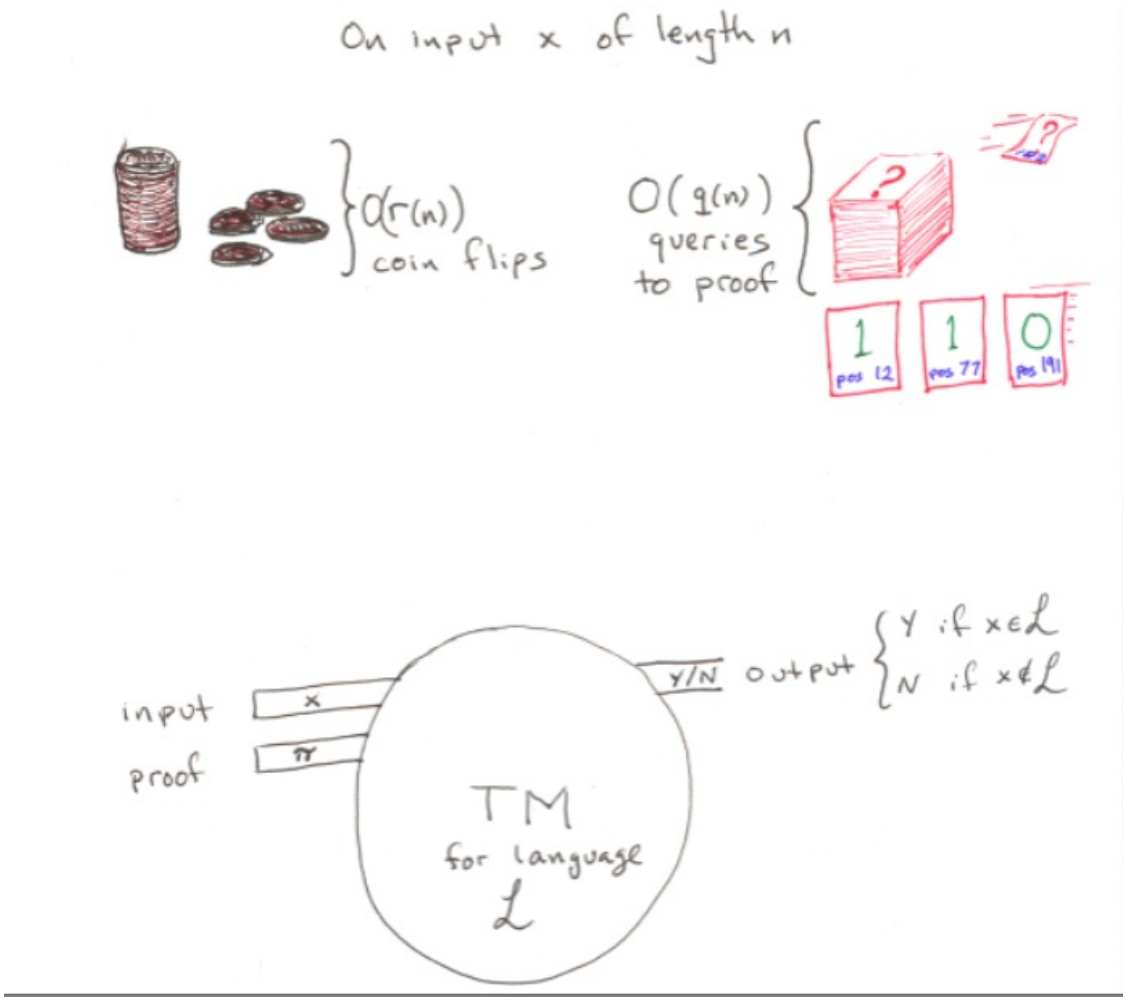
Probabilistically Checkable Proofs



Randomized Turing Machine uses $O(r(n))$ random bits and makes $O(q(n))$ queries to the proof π.

If π is a correct proof, x is accepted with probability one; if not, x is rejected with probability at least $\frac{1}{2}$.

Probabilistically Checkable Proofs



Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)
- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra, cf. Feige, et al.)

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)
- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra, cf. Feige, et al.)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)
- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra, cf. Feige, et al.)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)
- **1990s:** Implications for hardness of approximation

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)
- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra, cf. Feige, et al.)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)
- **1990s:** Implications for hardness of approximation
- **1999:** New hardness-based proof of PCP Theorem (Dinur)

Key Results

- **1980s:** Cryptography, IP, AM, constrained NP verifiers, ...
- **1990:** $IP = PSPACE$ (Shamir), $MIP = NEXP$ (Babai/Fortnow/Lund)
- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra, cf. Feige, et al.)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)
- **1990s:** Implications for hardness of approximation
- **1999:** New hardness-based proof of PCP Theorem (Dinur)
- **2001:** $NP = PCP_{1-\epsilon,\frac{1}{2}}[O(\log n), 3]$ (Håstad)

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$
- V is randomized: for $|x| = n$, it uses at most $r(n)$ random bits

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$
- V is randomized: for $|x| = n$, it uses at most $r(n)$ random bits
- V has random access to the proof: for $|x| = n$, it queries at most $q(n)$ bits from π

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$
- V is randomized: for $|x| = n$, it uses at most $r(n)$ random bits
- V has random access to the proof: for $|x| = n$, it queries at most $q(n)$ bits from π
- for $x \in \mathcal{L}$, there exists π such that $\text{Prob}[V^{\pi}(x) = 1] = 1$

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$
- V is randomized: for $|x| = n$, it uses at most $r(n)$ random bits
- V has random access to the proof: for $|x| = n$, it queries at most $q(n)$ bits from π
- for $x \in \mathcal{L}$, there exists π such that $\text{Prob}[V^{\pi}(x) = 1] = 1$
- for $x \notin \mathcal{L}$, for all π such that $\text{Prob}[V^{\pi}(x) = 0] \geq \frac{1}{2}$

PCP

- Our Turing machine V (the "Verifier" of the proof) has input x on tape and a "proof" π claiming $x \in \mathcal{L}$ as auxiliary input
- output is $V^{\pi}(x) \in \{0, 1\}$
- V is randomized: for $|x| = n$, it uses at most $r(n)$ random bits
- V has random access to the proof: for $|x| = n$, it queries at most $q(n)$ bits from π
- for $x \in \mathcal{L}$, there exists π such that $\text{Prob}[V^{\pi}(x) = 1] = 1$
- for $x \notin \mathcal{L}$, for all π such that $\text{Prob}[V^{\pi}(x) = 0] \geq \frac{1}{2}$
- the set of languages for which such a polynomial time V exists is denoted $PCP[r(n), q(n)]$

Recap PCP Theorems

- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra)

Recap PCP Theorems

- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)

Recap PCP Theorems

- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)
- **1999:** New hardness-based PCP Theorem (Dinur, employing expanding constraint graphs)

Recap PCP Theorems

- **1992:** $NP \subseteq PCP[O(\log n), O(\log n)]$ (Arora/Safra)
- **1992:** $NP = PCP[O(\log n), O(1)]$ (Arora/Lund/Motwani/Sudan/Szegedy)
- **1999:** New hardness-based PCP Theorem (Dinur, employing expanding constraint graphs)
- **2001:** $NP = PCP_{1-\epsilon,\frac{1}{2}}[O(\log n), 3]$ (Håstad) where probability of accepting a correct proof can be made arbitrarily close to one

Constraint Satisfaction Problems (CSP)

- alphabet Σ

Constraint Satisfaction Problems (CSP)

- alphabet Σ
- graph G, vertices are "variables", edges are "constraints"

Constraint Satisfaction Problems (CSP)

- alphabet Σ
- graph G, vertices are "variables", edges are "constraints"
- clauses $c(e) \subseteq \Sigma \times \Sigma$

Constraint Satisfaction Problems (CSP)

- alphabet Σ
- graph G, vertices are "variables", edges are "constraints"
- clauses $c(e) \subseteq \Sigma \times \Sigma$
- mapping $a : V(G) \to \Sigma$ is valid if $(a(x), a(y)) \in c(e)$ whenever $e = (x, y)$

Constraint Satisfaction Problems (CSP)

- alphabet Σ
- graph G, vertices are "variables", edges are "constraints"
- clauses $c(e) \subseteq \Sigma \times \Sigma$
- mapping $a : V(G) \to \Sigma$ is valid if $(a(x), a(y)) \in c(e)$ whenever $e = (x, y)$
- E.g., 3-coloring:
  $c(e) = \{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)\}$ for all e

Constraint Satisfaction Problems (CSP)

- alphabet Σ
- graph G, vertices are "variables", edges are "constraints"
- clauses $c(e) \subseteq \Sigma \times \Sigma$
- mapping $a : V(G) \to \Sigma$ is valid if $(a(x), a(y)) \in c(e)$ whenever $e = (x, y)$
- E.g., 3-coloring:
  $c(e) = \{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)\}$ for all e
- So CSP is **NP**-hard

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables
- assignment for each constraint is encoded using a Hadamard code

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables
- assignment for each constraint is encoded using a Hadamard code
- for proof π of length n, encode as a small-degree polynomial

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables
- assignment for each constraint is encoded using a Hadamard code
- for proof π of length n, encode as a small-degree polynomial
- choose $q \approx n^{1/m}$ and encode π as values of m-variate polynomial over $\mathbb{F}_q$

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables
- assignment for each constraint is encoded using a Hadamard code
- for proof π of length n, encode as a small-degree polynomial
- choose $q \approx n^{1/m}$ and encode π as values of m-variate polynomial over $\mathbb{F}_q$
- $m = \log t / \log \log t$ and $q = poly \log t$

Coding Theory Key to Proof

- If we are only going to sample a small portion of π, then an error-plagued version of π will also work with high probability
- so the proof π is "self-correcting"; may as well make it a codeword
- Target problem: CSP with t constraints each involving w variables
- assignment for each constraint is encoded using a Hadamard code
- for proof π of length n, encode as a small-degree polynomial
- choose $q \approx n^{1/m}$ and encode π as values of m-variate polynomial over $\mathbb{F}_q$
- $m = \log t / \log \log t$ and $q = poly \log t$
- $x : H^m \to \{0, 1\}$ where $H \subseteq \mathbb{F}_q$ has size $n^{1/m}$ has low-degree extension

Hardness of Approximation Results

- (Håstad) **max-3SAT** is **NP**-hard to approximate within a ratio of $> 7/8$

Hardness of Approximation Results

- (Håstad) **max-3SAT** is **NP**-hard to approximate within a ratio of $> 7/8$
- It is **NP**-hard to decide if a CSP has $\mathit{unsat}(\mathcal{C}) = 1$ or $\mathit{unsat}(\mathcal{C}) \leq 1/2$

Hardness of Approximation Results

- (Håstad) **max-3SAT** is **NP**-hard to approximate within a ratio of $> 7/8$
- It is **NP**-hard to decide if a CSP has $\mathit{unsat}(\mathcal{C}) = 1$ or $\mathit{unsat}(\mathcal{C}) \leq 1/2$
- For some $\epsilon > 0$, there is no $(1 + \epsilon)$-approximate algorithm for Steiner Tree unless $\mathbf{P} = \mathbf{NP}$

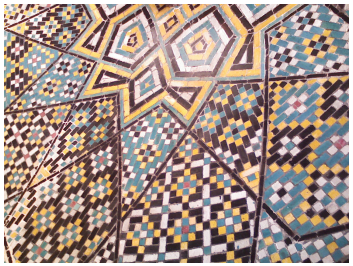
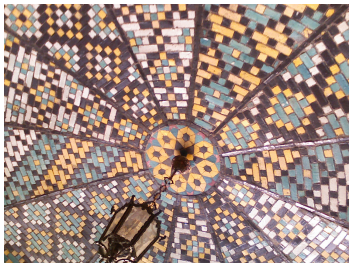
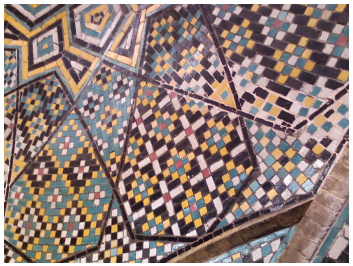
Hardness of Approximation Results

- (Håstad) **max-3SAT** is **NP**-hard to approximate within a ratio of $> 7/8$
- It is **NP**-hard to decide if a CSP has $\mathit{unsat}(\mathcal{C}) = 1$ or $\mathit{unsat}(\mathcal{C}) \leq 1/2$
- For some $\epsilon > 0$, there is no $(1 + \epsilon)$-approximate algorithm for Steiner Tree unless $\mathbf{P} = \mathbf{NP}$
- For some $\epsilon > 0$, there is no $(1 + \epsilon)$-approximate algorithm for Minimum Vertex Cover unless $\mathbf{P} = \mathbf{NP}$

Hardness of Approximation Results

- (Håstad) **max-3SAT** is **NP**-hard to approximate within a ratio of $> 7/8$
- It is **NP**-hard to decide if a CSP has $\mathit{unsat}(\mathcal{C}) = 1$ or $\mathit{unsat}(\mathcal{C}) \leq 1/2$
- For some $\epsilon > 0$, there is no $(1 + \epsilon)$-approximate algorithm for Steiner Tree unless $\mathbf{P} = \mathbf{NP}$
- For some $\epsilon > 0$, there is no $(1 + \epsilon)$-approximate algorithm for Minimum Vertex Cover unless $\mathbf{P} = \mathbf{NP}$
- For some $c > 1$, there is no n^c-approximate algorithm for Independent Set unless $\mathbf{P} = \mathbf{NP}$

The End


Thank you all!