Cryptography by Cellular Automata
or
How Fast Can Complexity Emerge in Nature?

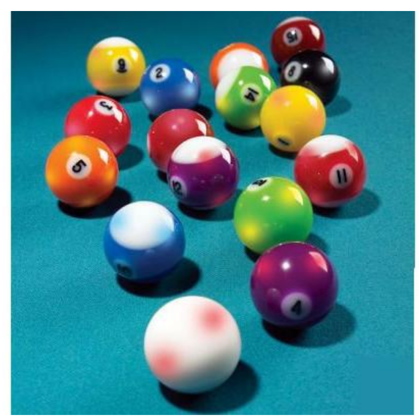
Benny Applebaum	Princeton $\rightarrow$ Weizmann
Yuval Ishai	Technion/UCLA
Eyal Kushilevitz	Technion

First Symposium on Innovations in Computer Science (ICS 2010)

Computation in the Physical World

Computation in the physical world is **spatially local**

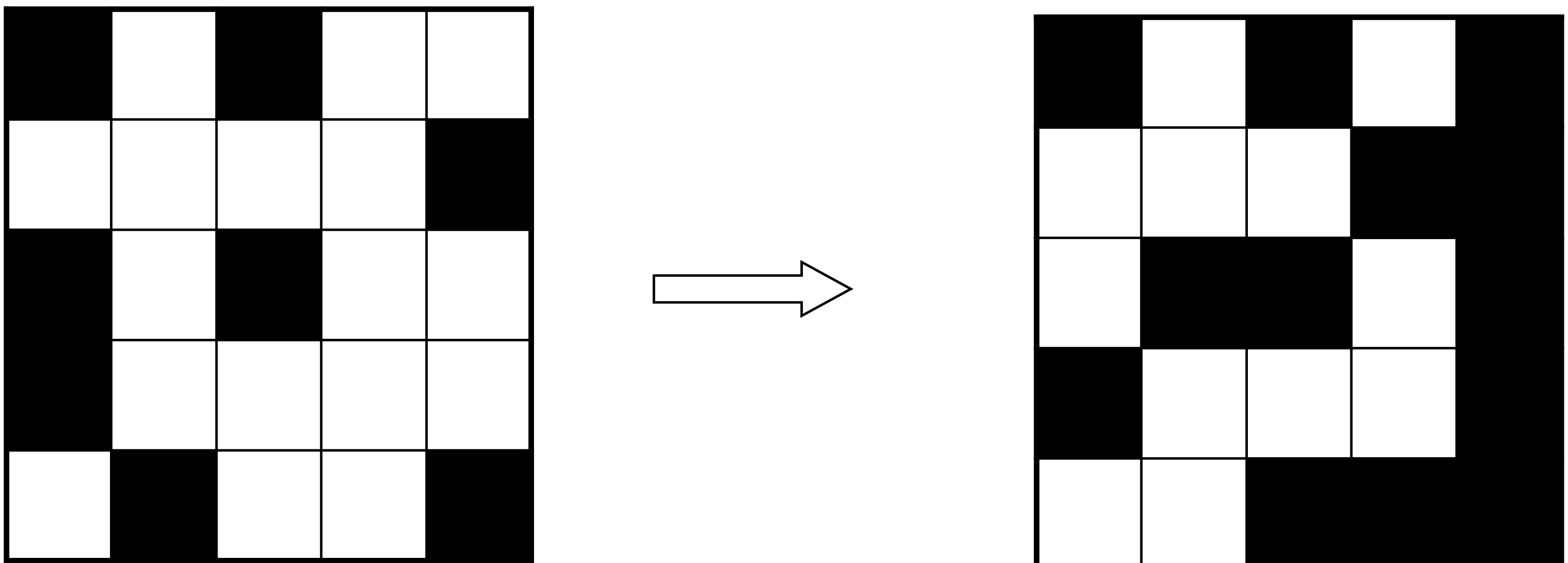
In a single time unit, information can only travel a bounded distance in space.



Cellular Automata

Spatially local computation is nicely captured by Cellular Automaton

- Grid of **n** cells each has a state = value from a fixed alphabet (e.g., binary)
- Configuration = State of all cells
- Next state of a cell computed via a local rule applied to its **d**-neighborhood
- Different cells can have different local rules but rules are fixed over time



Parity-CA with d=1

Brief History of Cellular Automata

- 50's defined by von Neumann-Ulam showed self-replication/universality
- 60's Zuse: "the universe is the output of a giant CA" (birth of digital physics)
- 70's popularized by Conway via the game of life
- 80's studied by Wolfram, Vichniac, Toffoli, Margolus via simulations
- 90's-today: study of complex systems
  - "simple rules lead to high complexity"
  - commonly used in physics, chemistry, biology, economics, sociology…
  - special conferences/journals (defeats TM's in a google test)





Motivation

- CAs exhibit complex computational and dynamical phenomena
  (self-replication, universality, synchronization, fractality, chaos,…)

- Main Question: How fast/common "complexity" is?

- Let's take complexity to be computational intractability

Can we infer the past from the present?

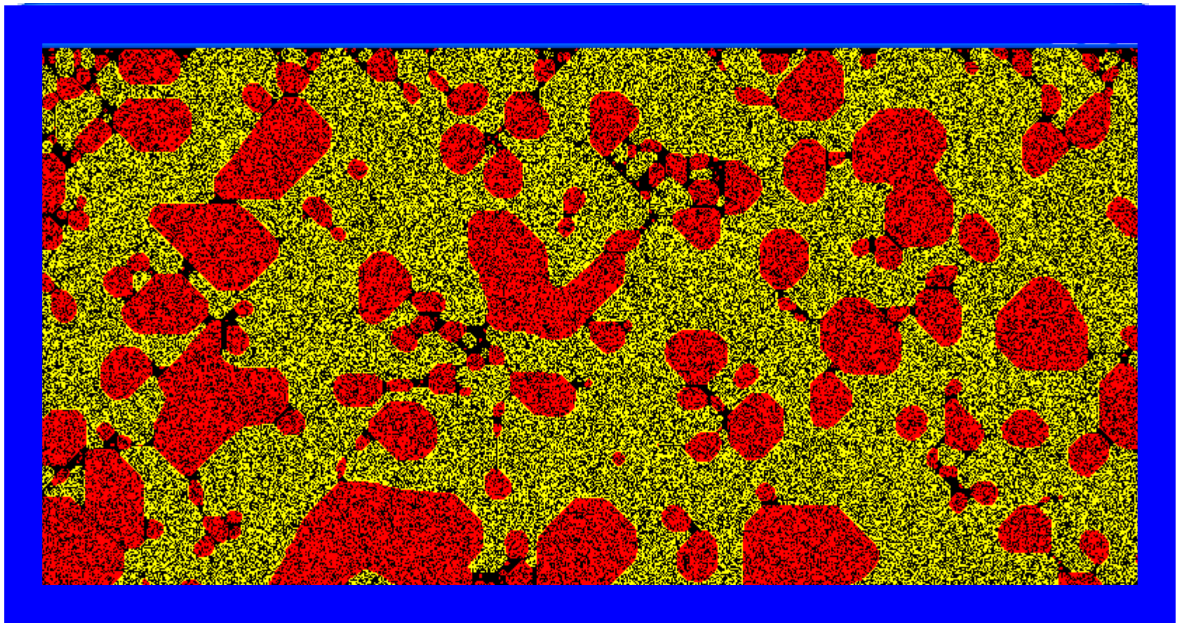
- **t-Inversion problem**

  - Initialize the CA to a random configuration $x = (x_1,\ldots,x_n)$.

  - Let the CA evolve for **t** steps to a configuration y.

  - Goal: given y find x or some other consistent initial configuration x'.

x y



Can we infer the past from the present?

- **t-Inversion problem**
  - Initialize the CA to a random configuration $x = (x_1,\dots,x_n)$.
  - Let the CA evolve for **t** steps to a configuration y.
  - Goal: given y find x or some other consistent initial configuration x'.

	t=O(1) (spatially local CSP)	t=poly(n)
Arbitrary **x**	• NP-hard [Cook-71] • Poly-time approximation [Lipton-Tarjan-77] • $2^{\mathrm{sqrt}(n)}$ time exact solution • NC^1 solution for 1-Dimensional CA	
Random **x**	?	Hard by universality

Can we get hardness for typical configurations in constant time?

Is it possible to compute one-way function in O(1) steps?

Can we predict the future based on partial observations of the present?

- **t-Prediction problem**

  - Initialize the CA to a random configuration $x = (x_1,\ldots,x_n)$.

  - Let the CA evolve for **t** steps and collect **k>n** intermediate values $(y_1,\ldots, y_k)$ from several sites during computation.

  - Goal: Predict a value of the sequence based on previous values

X

Can we predict the future based on partial observations of the present?

- **t-Prediction problem**

  - Initialize the CA to a random configuration $x = (x_1, \ldots, x_n)$.

  - Let the CA evolve for **t** steps and collect **k>n** intermediate values $(y_1, \ldots, y_k)$ from several sites during computation.

  - Goal: Predict a value of the sequence based on previous values

Can we generate pseudorandomess in O(1) steps?

- [Wolfram86] conjectured that the **prediction problem** is hard for **t=n** suggested a heuristic construction of a psuedorandom generator

- Many other heuristic candidates but t>poly(n) [Guan87,Habutsu-Nishio-Sasase-Mori91, Meier91, Gutowitz93, Nandi-Kar-Chaudhuri-94]

Our Results: Intractability is **Common** and **Fast**

- **DRLC Assumption**:

Can't Decode Random binary Linear Code of rate 1/6 w/noise level 1/4.

- **Thm.** The inversion and prediction problem are intractable even for **t=1**.

  - Construct explicit CAs for which problems are hard.

  - Tight Security: If DRLC is exponentially hard, get $2^{\mathrm{sqrt}(n)}$ hardness.

Our Results: Intractability is **Common** and **Fast**

• **DRLC Assumption**:

Can't Decode Random binary Linear Code of rate 1/6 w/noise level 1/4.

• **Thm.** The inversion and prediction problem are intractable even for **t=1**.

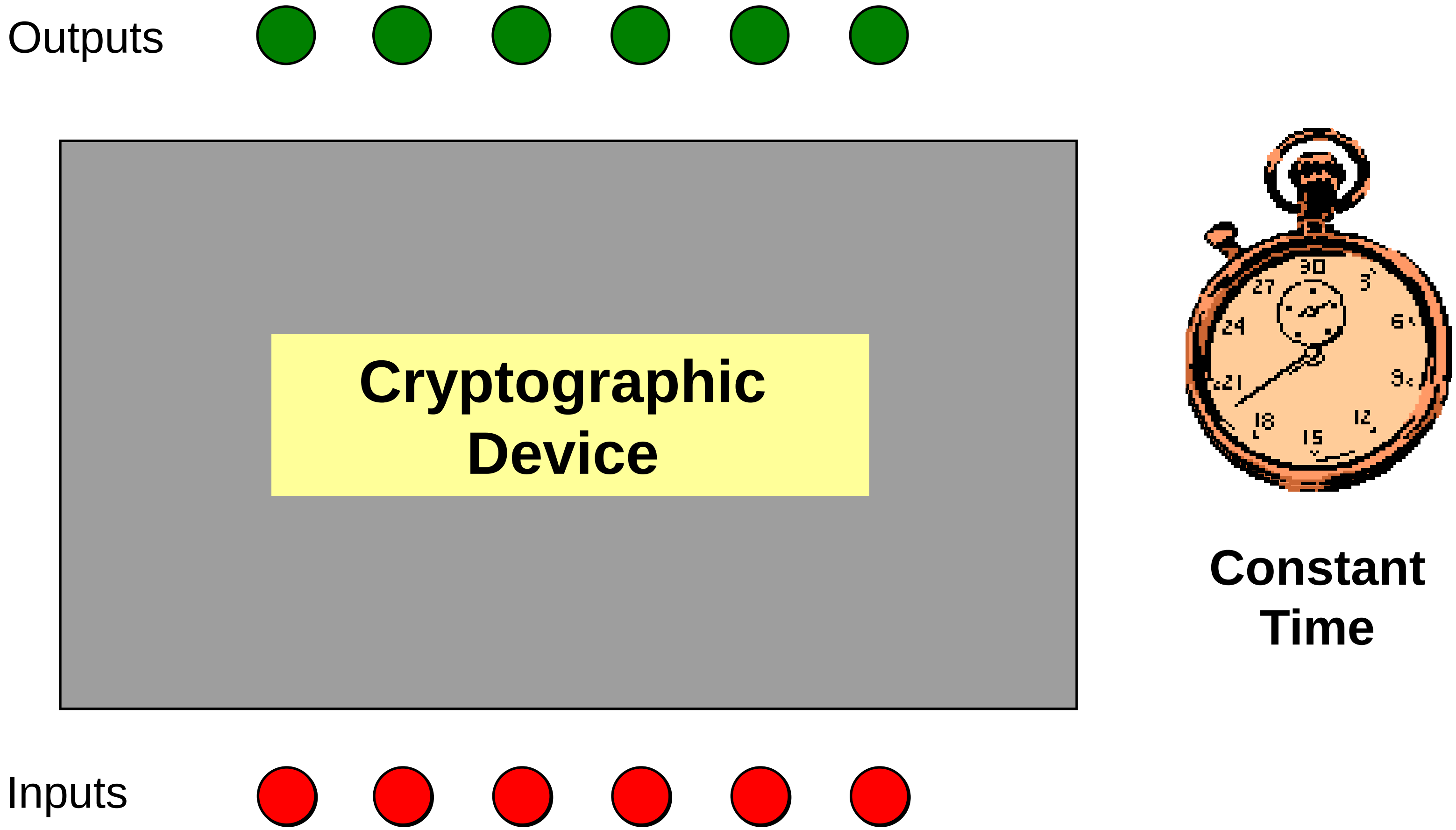
- Construct explicit CAs for which problems are hard.
- Tight Security: If DRLC is exponentially hard, get $2^{\mathrm{sqrt}(n)}$ hardness.

• Also **Characterization Thm** for crypto by CA in single step:

- Possible: Public/Symmetric-key Encryption, Commitment, Identification
- Impossible: Signatures, Decryption, $n+\Omega(n)$ pseudorandom sequence

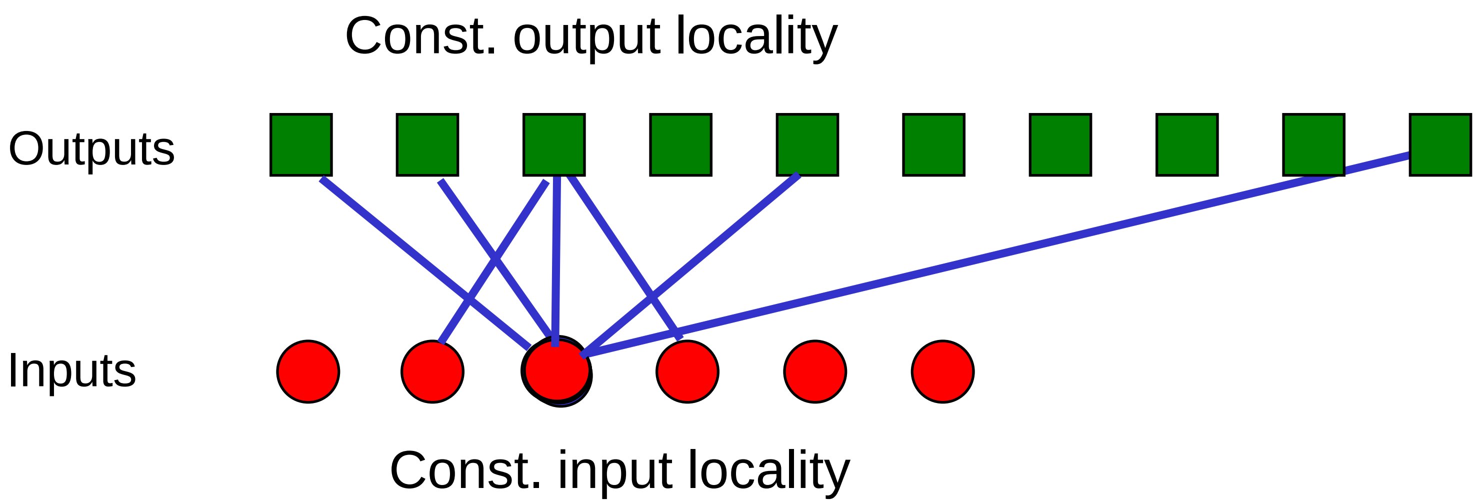
Application: Crypto with Constant Latency

Vision: const. time independent of security/input length



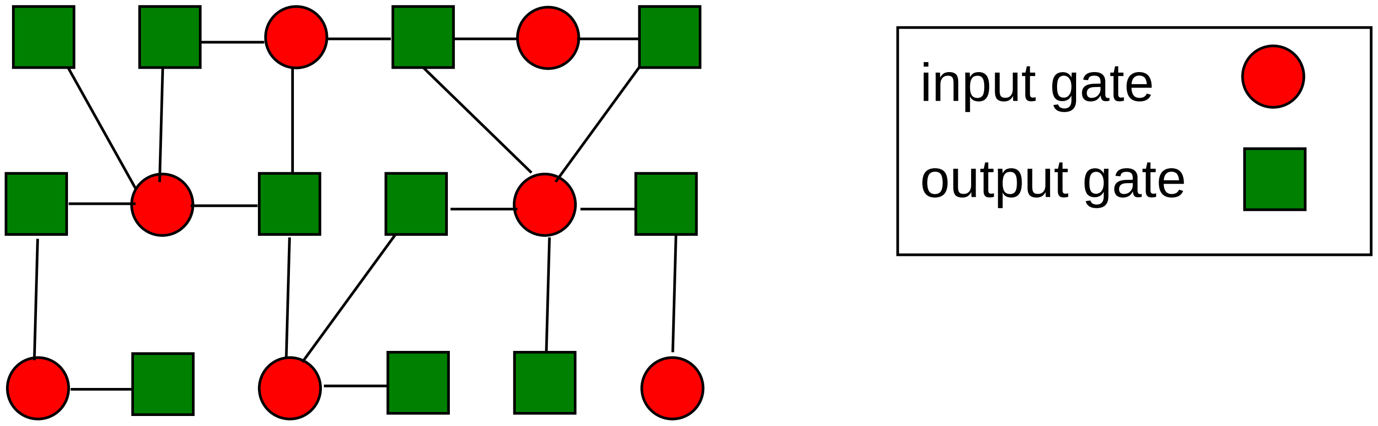
Previous Works

- Crypto with constant output locality (NC^0) [A-Ishai-Kushilevitz04]
- But fan-out is large ⇒ long time (replication cost)
- Crypto with constant output locality & input locality [AIK07]
- But long wires ⇒ long time
- This work: CA based primitives ⇒ crypto with spatial locality



Crypto with Spatial Locality

- Crypto with short wires embedding s.t. distance(input,output)= O(1)
  - similar model studied in VLSI by [Chazelle Monier 85]
- Input/output locality vs. Spatial locality: **qualitative** difference
  - Spatial locality ⇒ Graph is bad expander !
  - Seems bad for security…[Gol00,MST03,Alekhnovich03, AIK06]
  - Leads to actual attacks (PTAS, sub-exp attack, separation for some primitives)
- Inherently kills all previous constructions/approaches



About the Proof

Thm. Assume DRLC is hard.

Then, ∃ CA for which single-step inversion is average-case hard.

Proof approach:

DRLC ⇒ "Almost-linear" one-way function ⇒ Spatially-Local one-way function ⇒ OWF computable by CA in a single step

Almost-linear OWF

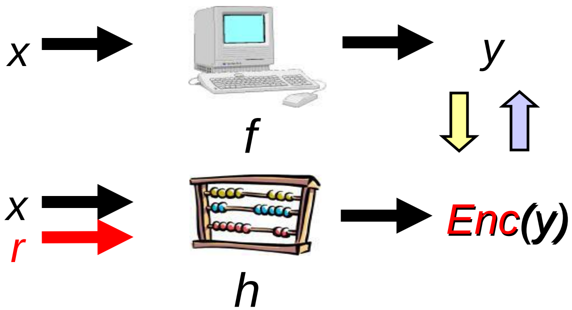
f

Spatially-Local OWF

h

By [AIK04] suffices to take h to be a randomized encoding of f

- for every x: the distribution h(x,r) "encodes" the string f(x)

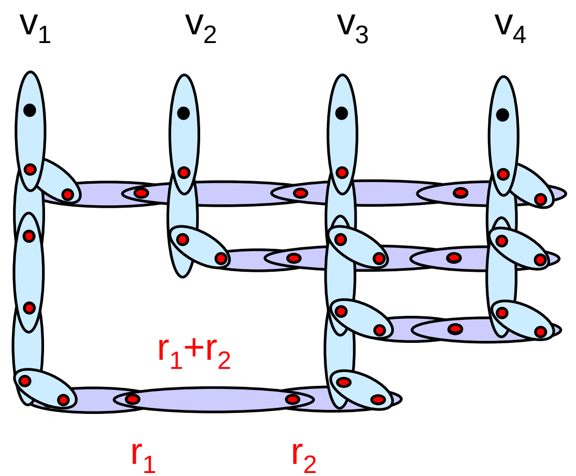


Spatial Encoding for Linear Functions

- **Want**: Spatial Encoding for "almost linear function"
- **Warm-up**: encode a fixed linear function $g_M(v)=Mv$

$$g_M(v) = \begin{matrix} v_1+v_4 \\ v_2+v_3+v_4 \\ v_3+v_4 \\ v_1+v_3 \end{matrix}$$

Randomized Encoding of g



Black nodes= original inputs

Red nodes= random inputs

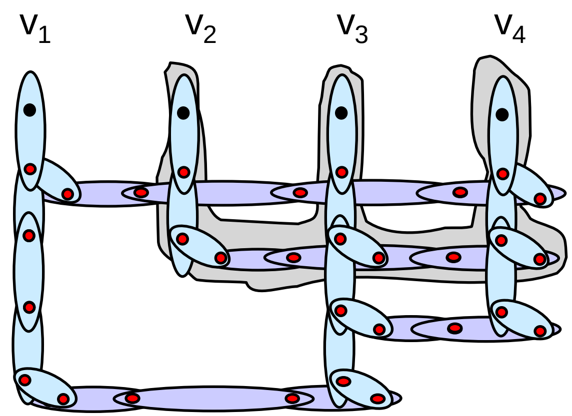
Hyper-Edges = outputs

Spatial Encoding for Linear Functions

- **Want**: Spatial Encoding for "almost linear function"
- **Warm-up**: encode a fixed linear function $g_M(v)=Mv$

$$g_M(v) = \begin{array}{c} \mathbf{v_1+v_4} \\ \boxed{\mathbf{v_2+v_3+v_4}} \\ \mathbf{v_3+v_4} \\ \mathbf{v_1+v_3} \end{array} \Leftarrow$$

Randomized Encoding of g



Black nodes= original inputs

Red nodes= random inputs

Hyper-Edges = outputs

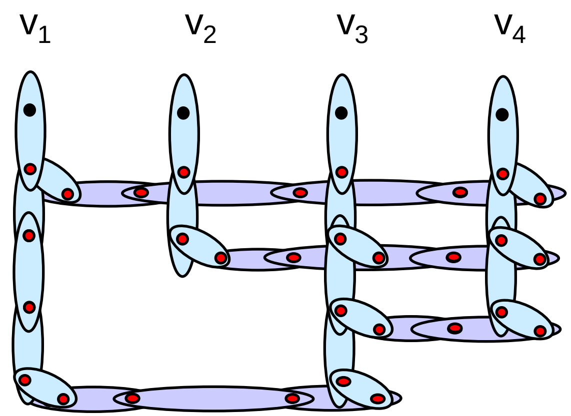
- Clearly encoding is spatially local
- To decode g(v) sum up the "right edges"

Spatial Encoding for Linear Functions

- **Want**: Spatial Encoding for "almost linear function"
- **Warm-up**: encode a fixed linear function $g_M(v)=Mv$

$$g_M(v) = \begin{matrix} \mathbf{v_1+v_4} \\ \mathbf{v_2+v_3+v_4} \\ \mathbf{v_3+v_4} \\ \mathbf{v_1+v_3} \end{matrix} \Longrightarrow$$

Randomized Encoding of g



Black nodes= original inputs

Red nodes= random inputs

Hyper-Edges = outputs

- Clearly encoding is spatially local
- To decode g(v) sum up the "right edges"
- Encoding is uniform under the above constraint

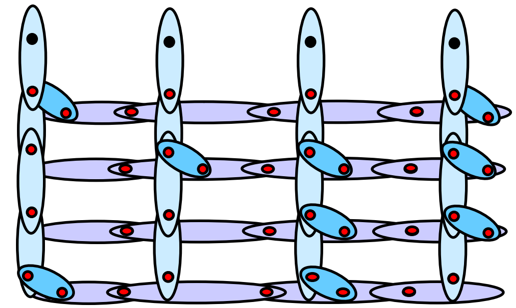
Useful Extensions

- More Generally:

M

1	0	0	**1**
0	**1**	**1**	**1**
0	0	**1**	**1**
1	0	**1**	0

$g_M(x)=Mx$



Encoding of g_M

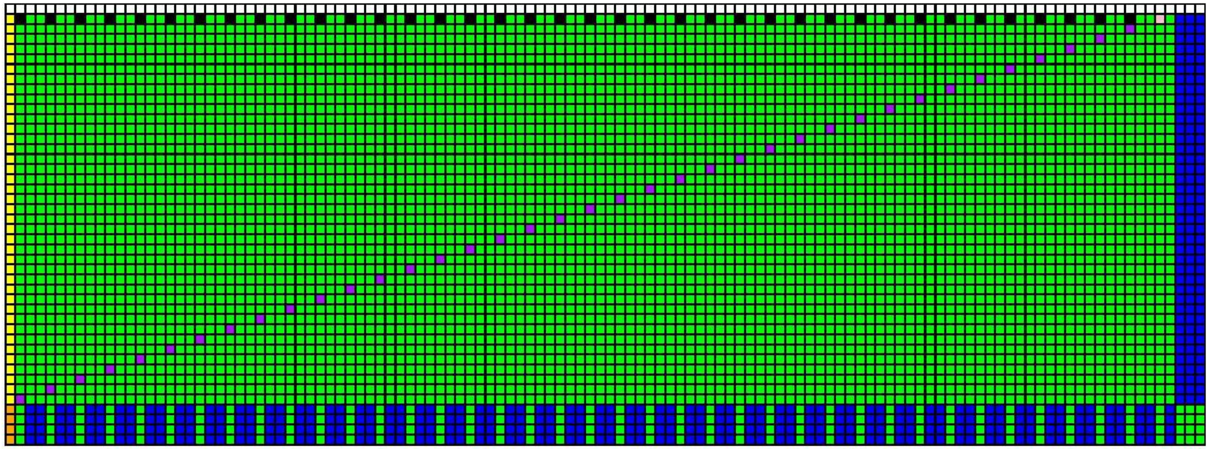
- Can encode the universal linear function L(M,v)=(M, Mv)

- Extends to “almost linear functions”

Conclusion

- In some CAs intractability is fast and common
  - What about a random CA?
- Spatially local functions can be used for crypto
  - approximation is easy but crypto is possible (unique example?)
- Well known: “Expansion leads to intractability”
  - proof complexity, inapproximability, property testing, SDP lower bounds
- New theme in crypto as well [Gol00,MST03,Alekhnovich03]
- This work: in crypto even weak expansion suffices

Thanks !



CA which computes a one-way function

Different colors correspond to different rules

	y_1	y_2	y_3
blue	x_1+b_1	$x_2+w_2+x_3\ x_1$	x_3
green	x_1+b_1	x_2+w_2	x_3
purple	x_1+b_1	$x_2+w_2+x_1$	x_3
black	$a_1c_1+x_1$	x_2+w_2	x_3
pink	$a_1c_1+x_1$	$x_2+w_2+x_1$	x_3
yellow	x_1+b_1	w_2	x_3
orange	x_1+b_1	w_2+x_1	x_3
white	0	0	0

c	
b	x
a	w