# AI and Cryptography

## Lecture 2 & 3 – AI Methods to Design Cryptographic Primitives

**Luca Mariot**

Semantics, Cybersecurity and Services Group, University of Twente

l.mariot@utwente.nl

Trieste, June 27, 2023

# Topics and Reading Material

**Main topics:**

- Boolean functions and S-boxes for symmetric crypto
- Genetic Algorithms to optimize Boolean functions
- S-boxes based on Cellular Automata
- Other representations: orthogonal arrays
- Evolving algebraic constructions

**References:**

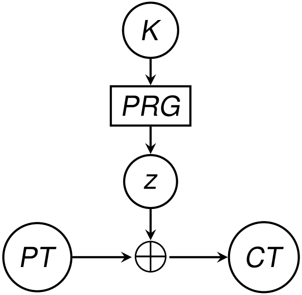
- C. Carlet. Boolean Functions for Cryptography and Coding Theory [C21]
- Survey papers: [MJBC22] and [DJMP23] (see references)

# This Lecture

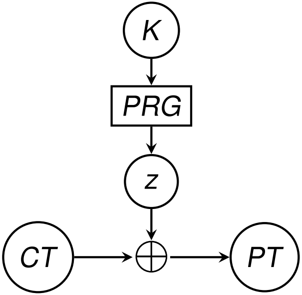
- Boolean Functions and S-boxes
- Evolutionary Algorithms
- Evolutionary Design of Boolean Functions and S-boxes
- Other Representations: orthogonal arrays
- Evolving Secondary Constructions

# Vernam-like Stream Cipher

- **PRG**: Pseudorandom generator that stretches a short secret key $K$ into an arbitrary long keystream $z$




(a) Encryption



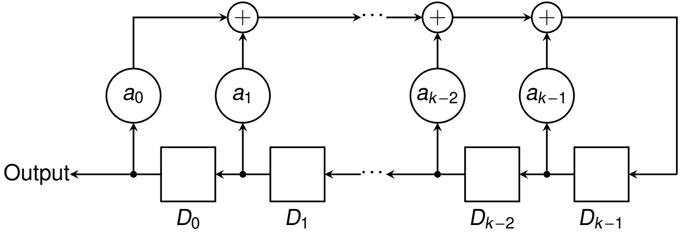

(b) Decryption

- **Question:** how to build a PRG in practice?

# Linear Feedback Shift Registers (LFSR)

- Device computing the *binary linear recurring sequence*

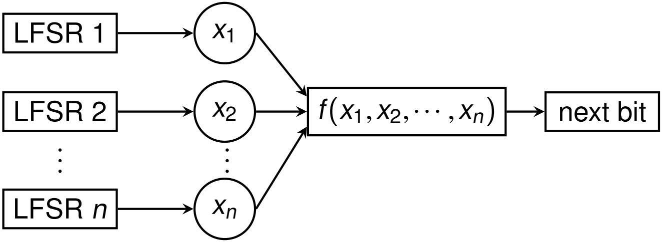
$$s_{n+k} = a + a_0 s_n + a_1 s_{n+1} + \cdots + a_{k-1} s_{n+k-1}$$




- **Too weak** as a PRG: $2k$ consecutive bits of keystream are enough to recover the LFSR initialization

# An Example of PRG: The Combiner Model

- a Boolean function $f : \{0,1\}^n \to \{0,1\}$ combines the outputs of $n$ LFSR [C21]




- Security of the combiner ⇔ **cryptographic properties** of $f$

# Boolean Functions - Basic Representations

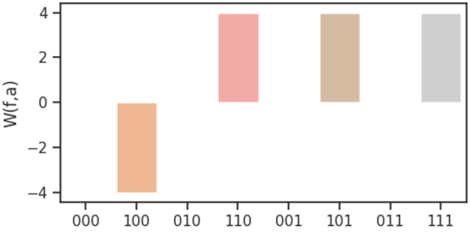
- Truth table: a $2^n$-bit vector $\Omega_f$ specifying $f(x)$ for all $x \in \{0,1\}^n$

| $(x_1, x_2, x_3)$ | 000 | 100 | 010 | 110 | 001 | 101 | 011 | 111 |
|---|---|---|---|---|---|---|---|---|
| $\Omega_f$ | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

- Algebraic Normal Form (ANF): Sum (XOR) of products (AND)

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \oplus x_2 x_3$$

- Walsh Transform: correlation with linear functions $a \cdot x$, $W(f,a) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus a \cdot x}$ for all $a \in \{0,1\}^n$



# Cryptographic Properties: Balancedness

- Hamming weight $w_H(f)$: number of 1s in $\Omega_f$
- A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is balanced if $w_H(f) = 2^{n-1}$
- Walsh characterization: $f$ balanced $\Leftrightarrow \hat{F}(0) = 0$

| $(x_1, x_2, x_3)$ | 000 | 100 | 010 | 110 | 001 | 101 | 011 | 111 |
|---|---|---|---|---|---|---|---|---|
| $\Omega_f$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

$$\Downarrow$$

$f$ is balanced

- Unbalanced functions present a statistical bias that can be exploited for *distinguishing attacks*

# Cryptographic Properties: Algebraic Degree

- Algebraic degree $d$: the degree of the multivariate polynomial representing the ANF of $f$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus x_3$$

$$\Downarrow$$

$$f \text{ has degree } d = 2$$

- *Linear* functions $\omega \cdot x = \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n$ have degree $d = 1$
- Boolean functions of high degree make the attack based on Berlekamp-Massey algorithm less effective

# Cryptographic Properties: Nonlinearity

- Nonlinearity $nl(f)$: Hamming distance of $f$ from linear functions
- Walsh characterization:

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{\omega \in \mathbb{F}_2^n} \left\{ \left| \hat{F}(\omega) \right| \right\}$$

| $(x_1, x_2, x_3)$ | 000 | 100 | 010 | 110 | 001 | 101 | 011 | 111 |
|---|---|---|---|---|---|---|---|---|
| $\Omega_f$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| $W_f$ | 0 | 0 | 0 | 0 | −4 | 4 | 4 | 4 |

$$\Downarrow$$

$$nl(f) = 2^{3-1} - \frac{1}{2} \cdot 4 = 2$$

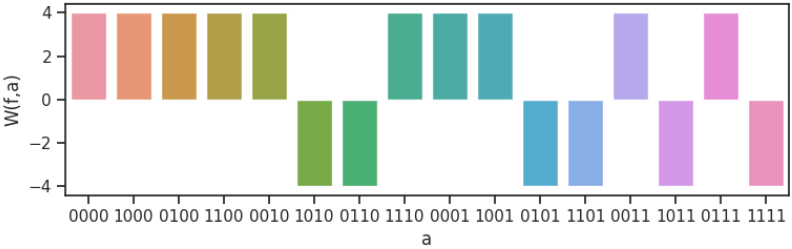
- Functions with high nonlinearity resist fast-correlation attacks

# Bent Functions

- *Parseval's Relation*, valid on any Boolean function:

$$\sum_{a \in \{0,1\}^n} [W(f,a)]^2 = 2^{2n} \text{ for all } f : \{0,1\}^n \to \{0,1\}$$

- Bent functions: $W(f,a) = \pm 2^{\frac{n}{2}}$ for all $a \in \{0,1\}^n$
  - Reach the highest possible *nonlinearity*
  - Exist only for $n$ even and they are *unbalanced*




Example: $f(x_1, x_2, x_3, x_4) = x_1 x_3 + x_1 x_4 + x_2 x_4$

# Cryptographic Properties: Resiliency

- $t$-Resiliency: when fixing any $t$ variables, the restriction of $f$ stays balanced
- Walsh characterization:

$$\hat{F}(\omega) = 0 \ \forall \omega : w_H(\omega) \leq t$$

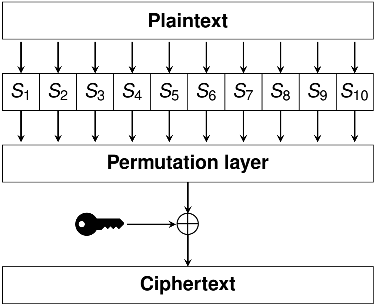
| $(x_1, x_2, x_3)$ | 000 | 100 | 010 | 110 | 001 | 101 | 011 | 111 |
|---|---|---|---|---|---|---|---|---|
| $\Omega_f$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| $\hat{F}(\omega)$ | 0 | 0 | 0 | 0 | $-4$ | 4 | 4 | 4 |

$$\Downarrow$$

$$F(001) = -4 \Rightarrow f \text{ is NOT 1-resilient}$$

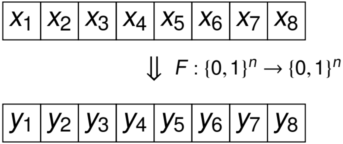
- Resilient functions of high order $t$ resist to correlation attacks

# S-boxes in SPN Ciphers




(a) Substitution-Permutation Network (SPN)

Zoom in on a **S-box** $S_i$:




(b) S-box $S_i$

S-boxes $F : \{0,1\}^n \rightarrow \{0,1\}^n$ are **vectorial** Boolean functions

# S-Boxes: General definitions

- The output of an $(n, m)$-function is defined by $m$ *coordinate functions* $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$.
- Hence, an S-box $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be represented by a $m \times 2^n$ *truth table*, where row $i$ is the truth table of $f_i$.
- Example: $n = m = 3$ (the 3-WAY S-box)

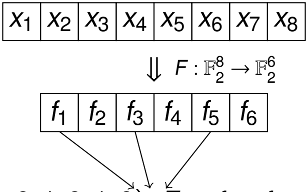
| $(x_1, x_2, x_3)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| $dec(x_1, x_2, x_3)$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| $F(x_1, x_2, x_3)$ | 0 | 5 | 6 | 1 | 3 | 2 | 4 | 7 |
| $f_1(x_1, x_2, x_3)$ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| $f_2(x_1, x_2, x_3)$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| $f_3(x_1, x_2, x_3)$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

# Component Functions

- Given $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ and a vector $v \in \mathbb{F}_2^m$, the *component function* $v \cdot F$ is defined for all $x \in \mathbb{F}_2^n$ as:

$$v \cdot F(x) = \bigoplus_{i=1}^{m} v_i f_i(x)$$

- Example with $n = 8$, $m = 6$ and $v = (1, 0, 1, 0, 1, 0)$:




$$(1, 0, 1, 0, 1, 0) \cdot F = f_1 \oplus f_3 \oplus f_5$$

- Component functions are thus *linear combinations* of coordinate functions.

# Walsh-Hadamard Transform (WHT)

- The Walsh-Hadamard Transform (WHT) of a $(n, m)$-function is the WHT of all its component functions $v \cdot F$, that is

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) \oplus a \cdot x}, \text{ for all } a \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m$$

- Example: $n = m = 3$ (the 3-WAY S-box)

| $(x_1, x_2, x_3)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| $F(x)$ | 000 | 101 | 110 | 001 | 011 | 010 | 100 | 111 |
| $W_F(a, 000)$ | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $W_F(a, 001)$ | 0 | 4 | 0 | −4 | 0 | 4 | 0 | 4 |
| $W_F(a, 010)$ | 0 | 0 | 0 | 0 | 4 | −4 | 4 | 4 |
| $W_F(a, 011)$ | 0 | 4 | 0 | 4 | −4 | 0 | 4 | 0 |
| $W_F(a, 100)$ | 0 | 0 | 4 | 4 | 0 | 0 | −4 | 4 |
| $W_F(a, 101)$ | 0 | −4 | 4 | 0 | 0 | 4 | 4 | 0 |
| $W_F(a, 110)$ | 0 | 0 | −4 | 4 | 4 | 4 | 0 | 0 |
| $W_F(a, 111)$ | 0 | 4 | 4 | 0 | 4 | 0 | 0 | −4 |

# Balancedness

- $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is *balanced* if $|F^{-1}(y)| = 2^{n-m}$ for all $y \in \mathbb{F}_2^m$.
- $F$ is balanced iff for all $v \in \mathbb{F}_2^m \setminus \{0\}$, the component function $v \cdot F$ is balanced.
- Balanced functions with $m = n$ are *invertible* (or bijective) S-boxes, since $|F^{-1}(y)| = 2^{n-n} = 1$.
- Example: $n = m = 3$, the 3-Way S-box

| $(x_1, x_2, x_3)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| $F(x)$ | 000 | 101 | 110 | 001 | 011 | 010 | 100 | 111 |

$$\Downarrow$$

$F$ is balanced (bijective)

# Nonlinearity

- Given $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$, recall that the Walsh-Hadamard transform for component $v \cdot F$ is, for all $a \in \mathbb{F}_2^n$:

$$W_f(a, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) \oplus a \cdot x}$$

- Hence, the nonlinearity of component $v \cdot F$ is:

$$nl(v \cdot F) = 2^{n-1} - \frac{1}{2} \max_{a \in \mathbb{F}_2^n} \left\{ \left| W_F(a, v) \right| \right\}$$

- The *nonlinearity* of a S-box $F$ is defined as the *minimum nonlinearity* among all its component functions $v \in \mathbb{F}_2^m \setminus \{0\}$:

$$nl(F) = min_{v \in \mathbb{F}_2^m \setminus \{0\}} \{ nl(v \cdot F) \}$$

# Nonlinearity – Example

- Example: $n = m = 3$, nonlinearity of the 3-WAY S-box

| $(x_1, x_2, x_3)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 | $nl$ |
|---|---|---|---|---|---|---|---|---|---|
| $F(x)$ | 000 | 101 | 110 | 001 | 011 | 010 | 100 | 111 | |
| $W_F(a, 001)$ | 0 | 4 | 0 | −4 | 0 | 4 | 0 | 4 | 2 |
| $W_F(a, 010)$ | 0 | 0 | 0 | 0 | 4 | −4 | 4 | 4 | 2 |
| $W_F(a, 011)$ | 0 | 4 | 0 | 4 | −4 | 0 | 4 | 0 | 2 |
| $W_F(a, 100)$ | 0 | 0 | 4 | 4 | 0 | 0 | −4 | 4 | 2 |
| $W_F(a, 101)$ | 0 | −4 | 4 | 0 | 0 | 4 | 4 | 0 | 2 |
| $W_F(a, 110)$ | 0 | 0 | −4 | 4 | 4 | 4 | 0 | 0 | 2 |
| $W_F(a, 111)$ | 0 | 4 | 4 | 0 | 4 | 0 | 0 | −4 | 2 |

$$\Downarrow$$

Nonlinearity of $F$: $nl = 2$

# Differential Uniformity

- Given $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the *delta difference table* of $F$ with respect to $a \in \mathbb{F}_2^n \setminus \{0\}$ and $b \in \mathbb{F}_2^m$ is:

$$\Delta_F(a,b) = \left\{ x \in \mathbb{F}_2^n : D_a F(x) = b \right\}$$

- Let $\delta_F(a,b) = |\Delta_F(a,b)|$. The *differential uniformity* of $F$ is:

$$\delta_F = \max_{\substack{a \in \mathbb{F}_2^n \setminus \{0\} \\ b \in \mathbb{F}_2^m}} \delta_F(a,b)$$

- S-boxes should have low differential uniformity to resist *differential cryptanalysis attacks*.

# Differential Uniformity – Example

- Example: $n = m = 3$, differential uniformity of the 3-WAY S-box

| $(x_1, x_2, x_3)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| $F(x)$ | 000 | 101 | 110 | 001 | 011 | 010 | 100 | 111 |

$$\Downarrow$$

| $\delta_F(a, b)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 001 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 |
| 010 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 |
| 011 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 0 |
| 100 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 2 |
| 101 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 0 |
| 110 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 |
| 111 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 2 |

$\Rightarrow$ differential uniformity of $F$: $\delta_f = 2$ (APN function)

# Trade-offs

Most of these properties cannot be satisfied simultaneously!

- *Covering Radius bound*: $nl \leq 2^{n-1} - 2^{\frac{n}{2}-1}$
- *Siegenthaler's bound*: $d \leq n - t - 1$
- *Tarannikov's bound*: $nl \leq 2^{n-1} - 2^{t+1}$

**Number** of Boolean functions of $n$ variables: $2^{2^n}$

| $n$ | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| $2^{2^n}$ | 256 | 65536 | $4.3 \cdot 10^9$ | $1.8 \cdot 10^{19}$ | $3.4 \cdot 10^{38}$ | $1.2 \cdot 10^{77}$ |

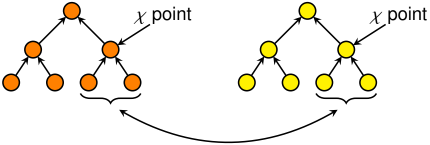
$\Rightarrow$ too huge for exhaustive search when $n > 5$!

**Number** of $(n,m)$-functions: $m2^{2^n}$

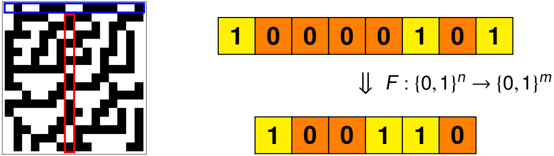
# This Lecture

# AI approaches to design symmetric primitives

- "Traditional" approach: ad-hoc and **algebraic constructions** to choose primitives with specific security properties
- "AI" approach: support the designer in choosing the primitives using **AI methods/models** from the following domains:
  - Optimization (Evolutionary algorithms, swarm intelligence...)




  - Computational models (cellular automata, neural networks...)



# Combinatorial Optimization

- Combinatorial Optimization Problem: map $\mathcal{P} : \mathcal{I} \to \mathcal{S}$ from a set $\mathcal{I}$ of *problem instances* to a family $\mathcal{S}$ of *solution spaces*
- $S = \mathcal{P}(I)$ is a finite set equipped with a *fitness function* $fit : S \to \mathbb{R}$, giving a score to candidate solutions $x \in S$
- Optimization goal: find $x^* \in S$ such that:

| **Minimization:** | **Maximization:** |
|---|---|
| $x^* = argmin_{x \in S}\{fit(x)\}$ | $x^* = argmax_{x \in S}\{fit(x)\}$ |

- Heuristic optimization algorithm: iteratively **tweaks** a set of candidate solutions using *fit* to drive the search

# Genetic Algorithms (GA) – Genetic Programming (GP)

Optimization algorithms loosely based on evolutionary principles, introduced respectively by **J. Holland** (1975) and **J. Koza** (1989)

- Evolve in parallel a **population** of solutions.
- **Black-box optimization**: use only the fitness function to optimize the solutions.
- Use **Probabilistic operators** to evolve the solutions

**GA Encoding**: individual ⇒ fixed-length bitstring

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |

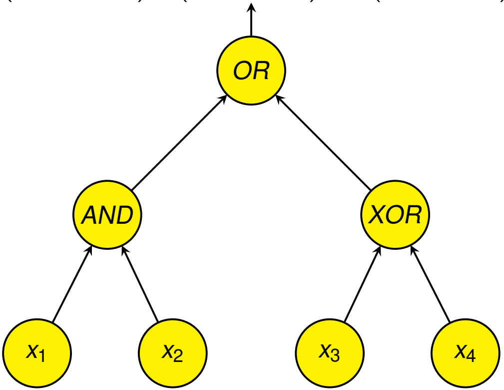
$$\Downarrow$$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus x_3$$

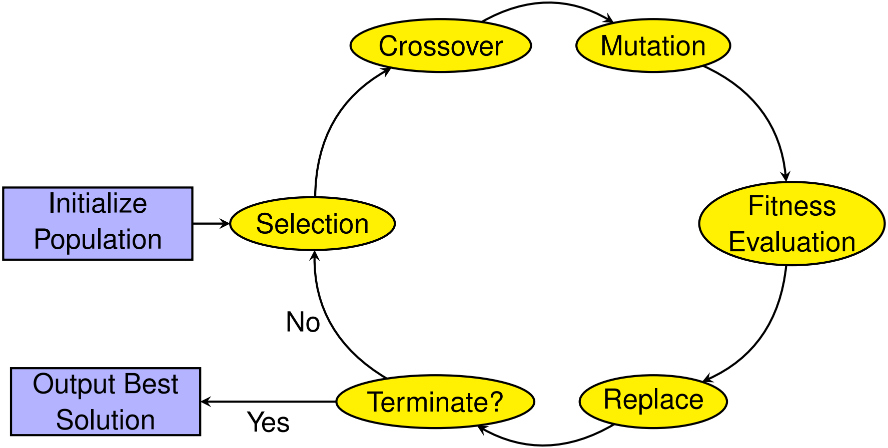
# Genetic Algorithms (GA) – Genetic Programming (GP)

- **GP Encoding**: an individual is represented by a tree
  - Terminal nodes: input variables of a program
  - Internal nodes: operators (e.g. AND, OR, NOT, XOR, ...)

$$f(x_1, x_2, x_3, x_4) = (x_1\ AND\ x2)\ OR\ (x_3\ XOR\ x_4)$$

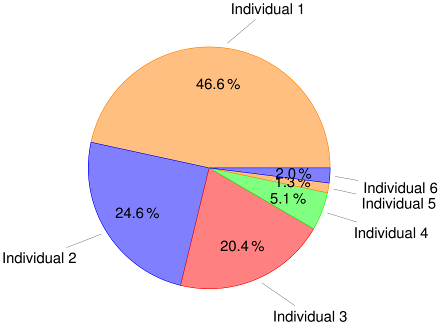


# The EA Loop


Crossover
Mutation
Initialize Population
Selection
Fitness Evaluation
No
Output Best Solution
Terminate?
Replace
Yes

**Roulette-Wheel Selection (RWS)**: the probability of selecting an individual is proportional to its fitness

**Tournament Selection (TS)**: Randomly sample $t$ individuals from the population and select the fittest one.



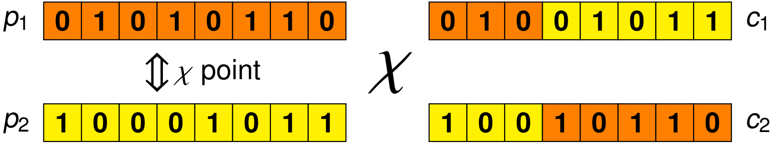

**Generational Breeding**: Draw as many pairs as population size

**Steady-State Breeding**: Select only a single pair

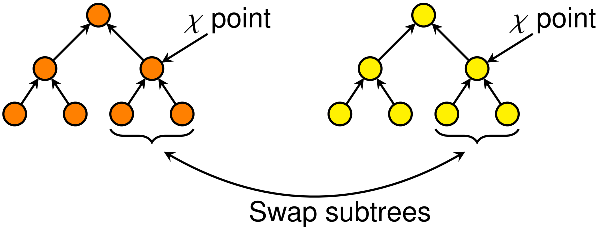
# Crossover

**Idea**: Recombine the genes of two parents individuals to create the offspring (Exploitation)

**GA Example:** One-Point Crossover



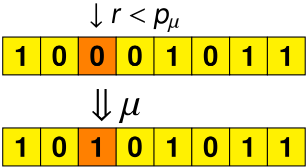

**GP Example:** Subtree Crossover



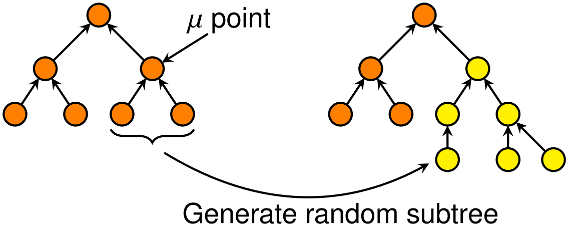
# Mutation

**Idea**: Introduce new genetic material in the offspring (Exploration)

**GA Example**: Bit-flip mutation




**GP Example**: Subtree mutation



# Replacement and Termination

- **Elitism**: keep the best individual from the previous generation
- **Termination**: several criteria such as budget of fitness evaluations, solutions diversity, ...

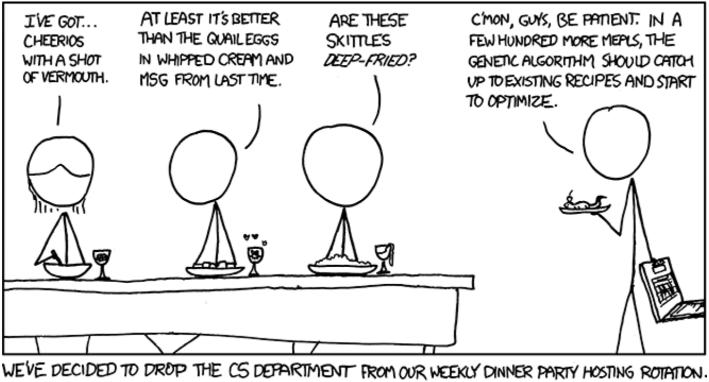



Image credit: `https://xkcd.com/720/`

# This Lecture

Boolean Functions and S-boxes

Evolutionary Algorithms

Evolutionary Design of Boolean Functions and S-boxes

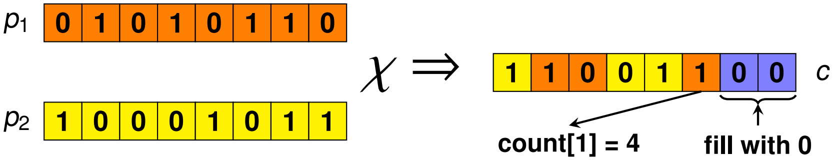
Other Representations: orthogonal arrays

Evolving Secondary Constructions

# Direct Search of Boolean Functions [MCD98]

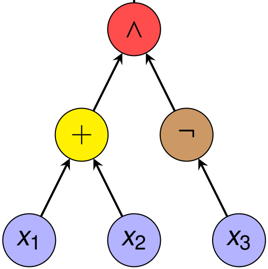
- GA encoding: represent the truth tables as $2^n$-bit strings
- Fitness function measuring nonlinearity, algebraic degree, and deviation from correlation-immunity
- Specialized crossover and mutation operators for preserving balancedness

**Crossover Idea:** Use *counters* to keep track of the multiplicities of zeros and ones [MCD98, MMT20]



# Evolving Boolean Functions with GP

- The truth table is synthesized from a GP tree:



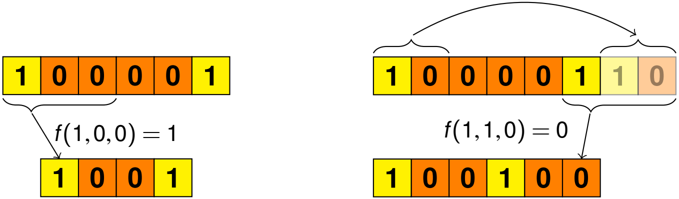

| $x_1$ | $x_2$ | $x_3$ | $f(x)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

- Difficult to enforce constraints on balancedness
- But, GP has better performance than GA with direct search [**?**]

# Cellular Automata

- One-dimensional Cellular Automaton (CA): a discrete parallel computation model composed of a finite array of $n$ cells

Example: $n = 6$, $d = 3$, $f(s_i, s_{i+1}, s_{i+2}) = s_i \oplus s_{i+1} \oplus s_{i+2}$ (rule 150)

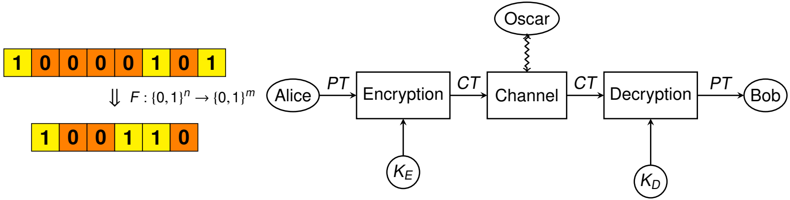



No Boundary CA – NBCA

Periodic Boundary CA – PBCA

- Each cell updates its state $s \in \{0,1\}$ by evaluating a local rule $f : \{0,1\}^d \rightarrow \{0,1\}$ on itself and the $d-1$ cells on its right

**General Research Goal**: Investigate cryptographic primitives defined by Cellular Automata

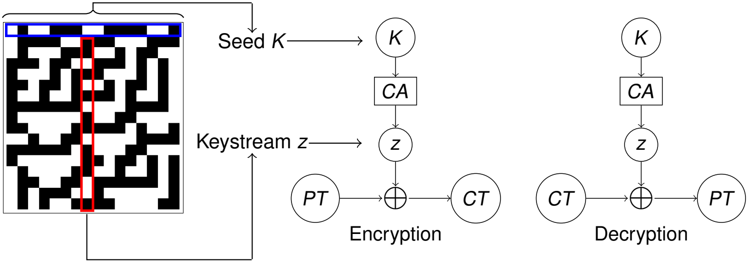



Why CA, anyway?

1. **Security from Complexity**: CA can yield very complex dynamical behaviors, depending on the local rule
2. **Efficient implementation**: Leverage CA parallelism and locality for lightweight cryptography

# CA-based Crypto History: Wolfram's PRNG

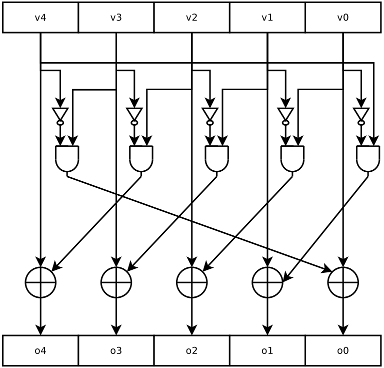
- CA-based Pseudorandom Generator (PRG) [W86]: central cell of rule 30 CA used as a stream cipher keystream




- Security claims based mainly on statistical/empirical tests
- This CA-based PRNG was later shown to be vulnerable, improvements by choosing larger local rules [LM14]

# Real world CA-Based Crypto: KECCAK $\chi$ S-box

- Local rule: $\chi(x_1, x_2, x_3) = x_1 \oplus (1 \oplus (x_2 \cdot x_3))$ (rule 210)
- Invertible for every odd size $n$ of the CA




- Used as a PBCA with $n = 5$ in the KECCAK specification of SHA-3 standard [BDPV11]

# Problem Statement

- Goal: Find PBCA of length $n$ and diameter $d = n$:
  - with cryptographic properties on par with those of other real-world ciphers [MPLJ19]
  - with low implementation cost [PMYJM17]
- Considered S-boxes sizes: from $n = 4$ to $n = 8$
- Genetic Programming to address this problem
- **Fitness function**: optimize *both* crypto (nonlinearity, differential uniformity) and implementation properties (GE measure)

# Results

Table: Statistical results and comparison.

| S-box size | $T\_max$ | GP | | | $N_F$ | $\delta_F$ |
|---|---|---|---|---|---|---|
| | | Max | Avg | Std dev | | |
| $4 \times 4$ | 16 | **16** | 16 | 0 | 4 | 4 |
| $5 \times 5$ | 42 | **42** | 41.73 | 1.01 | 12 | 2 |
| $6 \times 6$ | 86 | **84** | 80.47 | 4.72 | 24 | 4 |
| $7 \times 7$ | 182 | **182** | 155.07 | 8.86 | 56 | 2 |
| $8 \times 8$ | 364 | 318 | 281.87 | 13.86 | 82 | 20 |

- From $n = 4$ to $n = 7$, one obtains CA rules inducing S-boxes with optimal crypto properties
- Only for $n = 8$ the performances of GP are consistently worse wrt to the theoretical optimum

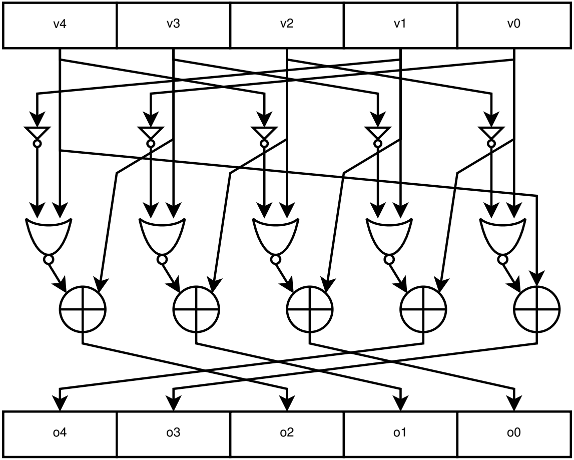
# A Posteriori Analysis – Implementation Properties, $n = 5$

Table: Power is in *nW*, area in *GE*, and latency in *ns*. *DPow*: dynamic power, *LPow*: cell leakage power

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Size | 5×5 | Rule | Keccak | | | | |
| DPow. | 321.684 | LPow: | 299.725 | Area: | 17 | Latency:0.14 | |
| Size | 5×5 | Rule | ((v2 NOR NOT(v4)) XOR v1) | | | | |
| DPow. | 324.849 | LPow: | 308.418 | Area: | 17 | Latency:0.14 | |
| Size | 5×5 | Rule | ((v4 NAND (v2 XOR v0)) XOR v1) | | | | |
| DPow. | 446.782 | LPow: | 479.33 | Area: | 24.06 | Latency:0.2 | |
| Size | 5×5 | Rule | (IF(v1, v2, v4) XOR (v0 NAND NOT(v3))) | | | | |
| DPow. | 534.015 | LPow: | 493.528 | Area: | 26.67 | Latency:0.17 | |

- Results on par with the Keccak $\chi$ S-box

# Example of Optimal CA S-box found by GP



# This Lecture

Boolean Functions and S-boxes

Evolutionary Algorithms

Evolutionary Design of Boolean Functions and S-boxes

Other Representations: orthogonal arrays

Evolving Secondary Constructions

# Correlation Immunity (Recall)

- $f$ is $t$-correlation immune iff $W_f(a) = 0$ for all $a$ s.t. $1 \leq HW(a) \leq t$, where $HW$ is the Hamming weight of $a$

| $(x_1, x_2, x_3)$ | 000 | **100** | **010** | **110** | **001** | **101** | **011** | 111 |
|---|---|---|---|---|---|---|---|---|
| $\Omega_f$ | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| $\hat{F}(\omega)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |

$$\Downarrow$$

$f$ is 2-order correlation immune

- $t$-order CI functions $\Rightarrow$ Masking countermeasures of order $t$ for **Side-Channel Analysis**

- $(N, k, s, t)$ **Orthogonal Array**: $N \times k$ matrix $A$ such that each $t$-uple occurs $\lambda = N/s^t$ times in each $N \times t$ submatrix.

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Example: OA** $(8, 4, 2, 3)$

$\Rightarrow$ Each 3-bit vector $(x_1, x_2, x_3) \in \{0, 1\}^3$ appears once in the submatrix with columns 1, 3, 4

- Applications in statistics, coding theory, cryptography

# Correlation Immunity: OA Characterization

- **Support** of $f$: sets of input vectors $x$ that map to 1 under $f$

**Truth table**

| $x_1$ | $x_2$ | $x_3$ | $f(x)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | **1** |

**Support**

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$\Downarrow$$

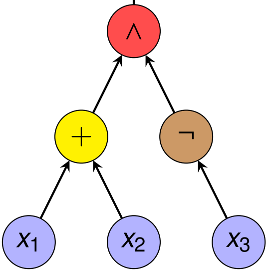
$$OA(4,3,2,2)$$

**Theorem**

*$f:\{0,1\}^n \to \{0,1\}$ is $t$-order CI $\Leftrightarrow$ Support of $f$ is an $OA(N,n,2,t)$, with $N = |Supp(f)|$*

# Solutions Encoding

- Each column is the truth table of a $n$-variable Boolean function
- For GP, the truth table is synthesized from the tree of the individual



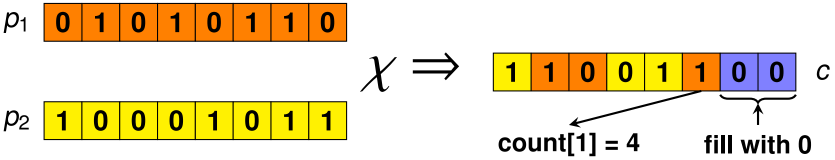

| $x_1$ | $x_2$ | $x_3$ | $f(x)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |

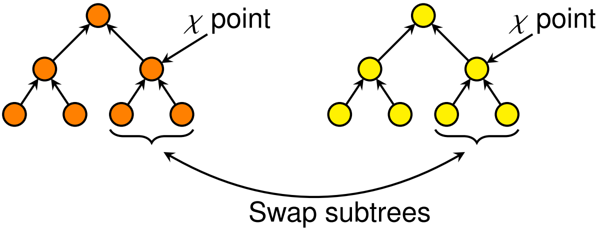
- Crossover and mutation are applied **column-wise**

# Crossover Operators

- **Classic GA and GP**: one-point and subtree crossover
- **Balanced GA**: counter-based crossover on each column

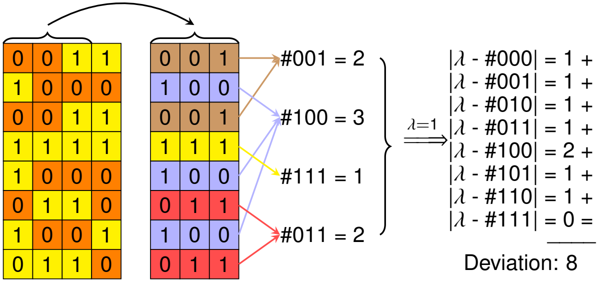



- **For GP**: Use standard subtree crossover



# Fitness Function

**Idea:** *minimize* in each $N \times t$ submatrix the number of occurrences of each $t$-uple deviating from $\lambda$




**Fitness function:** $L^p$ distance between vector $(\lambda, \cdots, \lambda)$ and the vector of deviations for each submatrix

$$fit_p(A) = \sum_{S \text{ Submatrix}} \left( \sum_{x \in \{0,1\}^t} |\lambda - \#x|^p \right)^{\frac{1}{p}}$$

# This Lecture

Boolean Functions and S-boxes

Evolutionary Algorithms

Evolutionary Design of Boolean Functions and S-boxes

Other Representations: orthogonal arrays

Evolving Secondary Constructions

# Evolving Secondary Constructions

Example of secondary construction: *Rothaus's* construction [**?**]

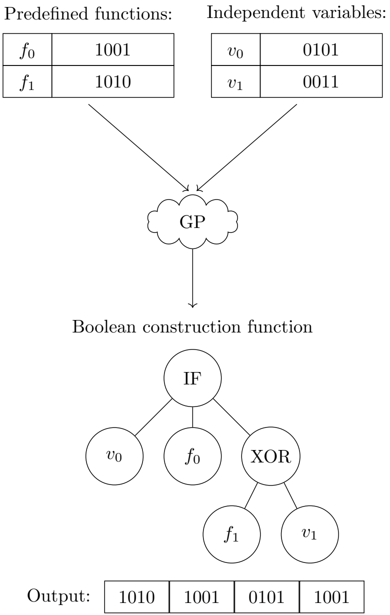
- If $g, h, k$ and $g \oplus h \oplus k$ are bent (maximally nonlinear) on $\mathbb{F}_2^n$, then the following function is bent:

$$f(x_1, x_2, x) = g(x)h(x) \oplus g(x)k(x) \oplus h(x)k(x) \oplus \\ \oplus [g(x) \oplus h(x)]x_1 \oplus [g(x) \oplus k(x)]x_2 \oplus x_1 x_2$$

  where $(x_1, x_2, x) \in \mathbb{F}_2^{n+2}$ with $x_1, x_2 \in \mathbb{F}_2$, $x \in \mathbb{F}_2^n$

**Goal:** Evolve secondary constructions using GP

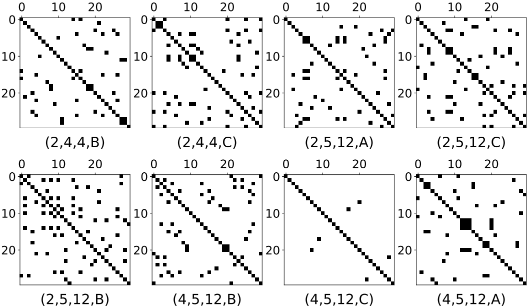
# GP Representation




- **Idea:** represent a secondary construction as a GP tree
- $f_0$, $f_1$: *seed* functions
- $v_0$ $v_1$: additional *independent variables*
- The GP tree yields a new function of $n+2$ variables
- Seed functions are obtained through direct GP search

# Simplification of GP Solutions

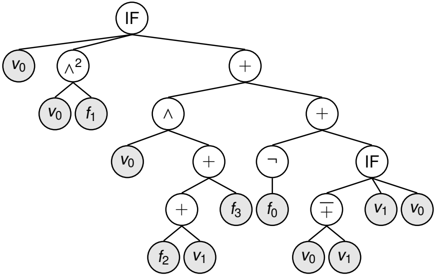
- ESPRESSO tool to *minimize* the best GP trees
- **Equivalence check** among the best solutions



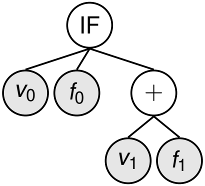

- **Result**: many solutions turn out to be the same construction, especially when 2 seeds are used

# Interpretation of Simplest Solutions

Example of bloated GP construction:




**Main Remark**: many constructions are equivalent to the well-known *indirect sum construction* [C21]




$$F(v_0, v_1, v) = \begin{cases} f_0(v) \ , & \text{if } v_0 = 1 \ , \\ f_1(v) \oplus v_1 \ , & \text{if } v_0 = 0 \ . \end{cases}$$

# Conclusions and Perspectives

**Summing up:**

- Up to now, AI-based methods and models can help in solving certain specific design problems for symmetric ciphers.
- Many more open directions remain!

**Open questions:**

- take into account other primitives (e.g. *permutation layers*)
- perform *fitness landscape analsysis* on these search spaces
- Develop new algebraic constructions with evolutionary algorithms

# References


- [BDPV11] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche: The Keccak reference. (January 2011). `http://keccak.noekeon.org/`
- [C21] C. Carlet: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)
- [DJMP23] M. Djurasevic, D. Jakobovic, L. Mariot, S. Picek: A Survey of Metaheuristic Algorithms for the Design of Cryptographic Boolean Functions. CoRR abs/2301.08012 (2023)
- [LM14] A. Leporati and L. Mariot: Cryptographic properties of bipermutive cellular automata rules. J. Cell. Autom. 9(5-6):437–475 (2014)
- [MMT20] L. Manzoni, L. Mariot, E. Tuba: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)
- [MJBC22] L. Mariot, D. Jakobovic, T. Bäck, J. Hernandez-Castro: Artificial Intelligence for the Design of Symmetric Cryptographic Primitives. Security and Artificial Intelligence 2022: 3-24 (2022)
- [MPLJ19] L. Mariot, S. Picek, A. Leporati, and D. Jakobovic. Cellular automata based S-boxes. Cryptography and Communications 11(1):41–62 (2019)
- [MCD98] W. Millan, J. Clark, E. Dawson: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. Proceedings of EUROCRYPT 1998, pp. 489-499 (1998)
- [PJMBC16] S. Picek, D. Jakobovic, J.F. Miller, L. Batina, M. Cupic: Cryptographic Boolean functions: One output, many design criteria. Appl. Soft Comput. 40: 635-653 (2016)
- [PMYJM17] S. Picek, L. Mariot, B. Yang, D. Jakobovic, N. Mentens: Design of S-boxes defined with cellular automata rules. Conf. Computing Frontiers 2017: 409-414 (2017)
- [W86] S. Wolfram. Cryptography with cellular automata. In CRYPTO ’85, pp. 429–432 (1986)