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
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)
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 $z$

![Diagram](attachment:image.png)

(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.
A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ combines the outputs of $n$ LFSR [C21]

- Security of the combiner $\iff$ cryptographic properties of $f$
Boolean Functions - Basic Representations

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

\[
\begin{array}{c|cccccccc}
(x_1, x_2, x_3) & 000 & 100 & 010 & 110 & 001 & 101 & 011 & 111 \\
\hline
\Omega_f & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

▶ 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 $a \cdot x$,

\[
W(f, a) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus a \cdot x} \text{ for all } a \in \{0, 1\}^n
\]
Cryptographic Properties: Balancedness

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

<table>
<thead>
<tr>
<th>( (x_1, x_2, x_3) )</th>
<th>000</th>
<th>100</th>
<th>010</th>
<th>110</th>
<th>001</th>
<th>101</th>
<th>011</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Omega_f )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \Downarrow \]

\( f \) 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 $f$

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

  ↓

  $f$ 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:**

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

<table>
<thead>
<tr>
<th>$(x_1, x_2, x_3)$</th>
<th>000</th>
<th>100</th>
<th>010</th>
<th>110</th>
<th>001</th>
<th>101</th>
<th>011</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_f$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W_f$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

$\Downarrow$

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

- **Functions with high nonlinearity resist fast-correlation attacks**
Bent Functions

▶ Parseval’s Relation, valid on any Boolean function:

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

▶ 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**: 
  \[
  \hat{\mathbf{F}}(\omega) = 0 \quad \forall \omega : \mathbf{w}_H(\omega) \leq t
  \]

<table>
<thead>
<tr>
<th>$(x_1, x_2, x_3)$</th>
<th>000</th>
<th>100</th>
<th>010</th>
<th>110</th>
<th>001</th>
<th>101</th>
<th>011</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_f$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{\mathbf{F}}(\omega)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$-4$</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

\[\downarrow\]

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

- Resilient functions of high order $t$ resist to correlation attacks.
S-boxes in SPN Ciphers

(a) Substitution-Permutation Network (SPN)

(b) S-box $S_i$

S-boxes $F : \{0,1\}^n \rightarrow \{0,1\}^n$ are **vectorial** Boolean functions
The output of an \((n, m)\)-function is defined by \(m\) coordinate functions \(f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2\).

Hence, an S-box \(F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m\) 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)

<table>
<thead>
<tr>
<th>((x_1, x_2, x_3))</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{dec}(x_1, x_2, x_3)</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>(F(x_1, x_2, x_3))</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>(f_1(x_1, x_2, x_3))</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(f_2(x_1, x_2, x_3))</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(f_3(x_1, x_2, x_3))</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Component Functions

Given $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ 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:

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

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

\[
\begin{array}{cccccccc}
  x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\end{array}
\]

\[
\downarrow \quad F : \mathbb{F}_2^8 \to \mathbb{F}_2^6
\]

\[
\begin{array}{cccccc}
  f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\
\end{array}
\]

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

Component functions are thus linear combinations of coordinate functions.
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)

<table>
<thead>
<tr>
<th>((x_1, x_2, x_3))</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F(x))</td>
<td>000</td>
<td>101</td>
<td>110</td>
<td>001</td>
<td>011</td>
<td>010</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>(W_F(a, 000))</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(W_F(a, 001))</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>(-4)</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>(W_F(a, 010))</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>(-4)</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>(W_F(a, 011))</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>(-4)</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(W_F(a, 100))</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>(-4)</td>
<td>4</td>
</tr>
<tr>
<td>(W_F(a, 101))</td>
<td>0</td>
<td>(-4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(W_F(a, 110))</td>
<td>0</td>
<td>0</td>
<td>(-4)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(W_F(a, 111))</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>(-4)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>$(x_1, x_2, x_3)$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F(x)$</td>
<td>000</td>
<td>101</td>
<td>110</td>
<td>001</td>
<td>011</td>
<td>010</td>
<td>100</td>
<td>111</td>
</tr>
</tbody>
</table>

$\Downarrow$

$F$ 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:

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

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

$$nl(F) = \min_{v \in \mathbb{F}_2^m \setminus \{0\}} \{nl(v \cdot F)\}$$
Example: $n = m = 3$, nonlinearity of the 3-WAY S-box

<table>
<thead>
<tr>
<th>$(x_1, x_2, x_3)$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
<th>$nl$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F(x)$</td>
<td>000</td>
<td>101</td>
<td>110</td>
<td>001</td>
<td>011</td>
<td>010</td>
<td>100</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>$W_F(a,001)$</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>−4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,010)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>−4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,011)$</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>−4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,100)$</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>−4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,101)$</td>
<td>0</td>
<td>−4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,110)$</td>
<td>0</td>
<td>0</td>
<td>−4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$W_F(a,111)$</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>−4</td>
<td>2</td>
</tr>
</tbody>
</table>

⇓

Nonlinearity of $F$: $nl = 2$
Differential Uniformity

Given $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$, 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:

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

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

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

S-boxes should have low differential uniformity to resist differential cryptanalysis attacks.
Example: \( n = m = 3 \), differential uniformity of the 3-WAY S-box

<table>
<thead>
<tr>
<th>((x_1, x_2, x_3))</th>
<th>000 001 010 011 100 101 110 111</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F(x))</td>
<td>000 101 110 001 011 010 100 111</td>
</tr>
</tbody>
</table>

\[\delta_f(a, b)\]

<table>
<thead>
<tr>
<th>(\delta_f(a, b))</th>
<th>000 001 010 011 100 101 110 111</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>0 2 0 2 0 2 0 2</td>
</tr>
<tr>
<td>010</td>
<td>0 0 0 0 2 2 2 2</td>
</tr>
<tr>
<td>011</td>
<td>0 2 0 2 2 0 2 0</td>
</tr>
<tr>
<td>100</td>
<td>0 0 2 2 0 0 2 2</td>
</tr>
<tr>
<td>101</td>
<td>0 2 2 0 0 2 2 0</td>
</tr>
<tr>
<td>110</td>
<td>0 0 2 2 2 2 0 0</td>
</tr>
<tr>
<td>111</td>
<td>0 2 2 0 2 0 0 2</td>
</tr>
</tbody>
</table>

\[\Rightarrow \text{differential uniformity of } F: \delta_f = 2 \text{ (APN function)}\]
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: $2^{2^n}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$3$</th>
<th>$4$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{2^n}$</td>
<td>256</td>
<td>65536</td>
<td>$4.3 \cdot 10^9$</td>
<td>$1.8 \cdot 10^{19}$</td>
<td>$3.4 \cdot 10^{38}$</td>
<td>$1.2 \cdot 10^{77}$</td>
</tr>
</tbody>
</table>

⇒ too huge for exhaustive search when $n > 5$!

**Number** of $(n, m)$-functions: $m2^{2^n}$
Boolean Functions and S-boxes

Evolutionary Algorithms

Evolutionary Design of Boolean Functions and S-boxes

Other Representations: orthogonal arrays

Evolving Secondary Constructions
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...)

\[
\chi \text{ point }
\]

\[
\chi \text{ point }
\]

\[
F : \{0,1\}^n \rightarrow \{0,1\}^m
\]

1 0 0 0 0 1 0 1

↓

1 0 0 1 1 0
Combinatorial Optimization

- Combinatorial Optimization Problem: map $\mathcal{P} : \mathcal{I} \rightarrow S$ from a set $\mathcal{I}$ of problem instances to a family $S$ of solution spaces
- $S = \mathcal{P}(\mathcal{I})$ is a finite set equipped with a fitness function $\text{fit} : S \rightarrow \mathbb{R}$, giving a score to candidate solutions $x \in S$
- Optimization goal: find $x^* \in S$ such that:
  - Minimization: $x^* = \arg\min_{x \in S} \{ \text{fit}(x) \}$
  - Maximization: $x^* = \arg\max_{x \in S} \{ \text{fit}(x) \}$
- Heuristic optimization algorithm: iteratively tweaks a set of candidate solutions using $\text{fit}$ to drive the search
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 $\Rightarrow$ fixed-length bitstring

```
0 1 1 1 1 0 0 0
```

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

\[ f(x_1, x_2, x_3, x_4) = (x_1 \text{ AND } x_2) \text{ OR } (x_3 \text{ XOR } x_4) \]
The EA Loop

Initialize Population

Selection

Crossover

Mutation

Fitness Evaluation

Replace

Terminate?

Yes

No

Output Best Solution

L. Mariot

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

**GA Example:** One-Point Crossover

\[ p_1: \begin{array}{ccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array} \quad c_1: \begin{array}{ccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \]

\[ p_2: \begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \quad c_2: \begin{array}{ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array} \]

**GP Example:** Subtree Crossover

Swap subtrees
**Mutation**

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

**GA Example:** Bit-flip mutation

\[
downarrow r < p_{\mu} \\
1\ 0\ 0\ 0\ 1\ \ 0\ 1\ 1\ 1
\]

\[
downarrow \mu \\
1\ 0\ 1\ 0\ 1\ 0\ 1\ 1
\]

**GP Example:** Subtree mutation

![Diagram of GP Example: Subtree mutation](image)

- Generate random subtree
- \(\mu\) point
- \(\mu\) point
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]

\[ p_1 = 010101110 \]
\[ p_2 = 100010111 \]
\[ \chi \Rightarrow \begin{array}{cccccccc}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
count[1] = 4 & & & & & & & fill with 0
\end{array} \]
Evolving Boolean Functions with GP

- The truth table is synthesized from a GP tree:

  \[
  f(x) = \neg (x_1 \lor (x_2 \land x_3))
  \]

- 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)

```
  1 0 0 0 0 1 1 0
```

  \( f(1,0,0) = 1 \)

```
  1 0 0 1
```

  No Boundary CA – NBCA

```
  1 0 0 0 0 1 1 0
```

  \( f(1,1,0) = 0 \)

```
  1 0 0 1 0 0
```

  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

\[ f(s_i, s_{i+1}, \ldots, s_{i+d-1}) = \begin{cases} 1 & \text{if } s_i \oplus \cdots \oplus s_{i+d-1} = 1 \\ 0 & \text{otherwise} \end{cases} \]
Motivations

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

```
Seed K ────> K
    |      |     |
    |      V     |
CA ───> Z ───> K
    |      |     |
    |      V     |
PT ───> CT  Encryption
    |      |
    |      V
CT ───> PT  Decryption
```

- 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]
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.

<table>
<thead>
<tr>
<th>S-box size</th>
<th>$T_{max}$</th>
<th>GP</th>
<th>$N_F$</th>
<th>$\delta_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Avg</td>
<td>Std dev</td>
<td></td>
</tr>
<tr>
<td>$4 \times 4$</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>42</td>
<td>42</td>
<td>41.73</td>
<td>1.01</td>
</tr>
<tr>
<td>$6 \times 6$</td>
<td>86</td>
<td>84</td>
<td>80.47</td>
<td>4.72</td>
</tr>
<tr>
<td>$7 \times 7$</td>
<td>182</td>
<td>182</td>
<td>155.07</td>
<td>8.86</td>
</tr>
<tr>
<td>$8 \times 8$</td>
<td>364</td>
<td>318</td>
<td>281.87</td>
<td>13.86</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Size</th>
<th>Rule</th>
<th>Keccak</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5 \times 5$</td>
<td>$321.684 \text{ LPow: } 299.725 \text{ Area: } 17 \text{ Latency:0.14}$</td>
<td></td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>$324.849 \text{ LPow: } 308.418 \text{ Area: } 17 \text{ Latency:0.14}$</td>
<td></td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>$446.782 \text{ LPow: } 479.33 \text{ Area: } 24.06 \text{ Latency:0.2}$</td>
<td></td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>$534.015 \text{ LPow: } 493.528 \text{ Area: } 26.67 \text{ Latency:0.17}$</td>
<td></td>
</tr>
</tbody>
</table>

- Results on par with the Keccak $\chi$ S-box
Example of Optimal CA S-box found by GP

L. Mariot
AI and Cryptography
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 $a$

\[
\begin{array}{c|cccccccc}
(x_1, x_2, x_3) & 000 & 100 & 010 & 110 & 001 & 101 & 011 & 111 \\
\hline
\Omega_f & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
\hat{F}(\omega) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\
\end{array}
\]

$\Downarrow$

- $f$ is 2-order correlation immune

- $t$-order CI functions $\Rightarrow$ Masking countermeasures of order $t$ for **Side-Channel Analysis**
Orthogonal Arrays (OA)

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

Example: OA \((8, 4, 2, 3)\)

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 $f$

<table>
<thead>
<tr>
<th>Truth table</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Theorem

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

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

\[
p_1 \quad \begin{array}{cccccc}
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
\]

\[
p_2 \quad \begin{array}{cccccc}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
\]

\[
\chi \quad \Rightarrow \quad \begin{array}{cccccc}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0
\end{array}
\]

\[
\text{count}[1] = 4 \quad \text{fill with } 0
\]

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

Swap subtrees

\[
\chi \text{ point}
\]

\[
\chi \text{ point}
\]
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
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:

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

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**

- **Predefined functions:**
  - $f_0$: 1001
  - $f_1$: 1010

- **Independent variables:**
  - $v_0$: 0101
  - $v_1$: 0011

**Boolean construction function**

- **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

Output: 1010 1001 0101 1001
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]

\[ F(v_0, v_1, v) = \begin{cases} 
  f_0(v) & , \quad \text{if } v_0 = 1 , \\
  f_1(v) \oplus v_1 & , \quad \text{if } v_0 = 0 . 
\end{cases} \]
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 analysis on these search spaces
▶ Develop new algebraic constructions with evolutionary algorithms
References


