On the S-boxes Generated via Cellular Automata Rules

Stjepan Picek[1], Luca Mariot[2], Domagoj Jakobovic[3], Alberto Leporati[2]

[1] CSAIL, MIT, USA and Cyber Security Research Group, TU Delft, The Netherlands

[2] DISCo, Università degli Studi Milano - Bicocca, Italy

[3] University of Zagreb, Croatia

July 4, 2017

Outline

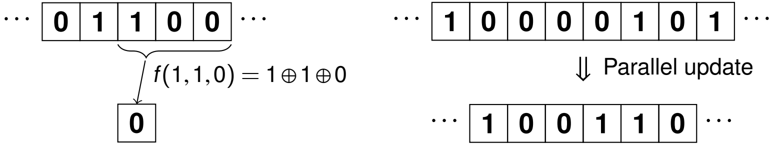
Cellular Automata (CA)

Definition

One-dimensional cellular automaton: triple $\langle n, d, f \rangle$ where $n \in \mathbb{N}$ is the number of cells arranged on a one-dimensional array, $d \in \mathbb{N}$ is the neighborhood size and $f : \mathbb{F}_2^d \rightarrow \mathbb{F}_2$ is the local rule

- Each cell synchronously updates its state $s \in \mathbb{F}_2$ by applying f to itself and the $d-1$ cells to its right

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



CA Global Rule and Boundary Conditions

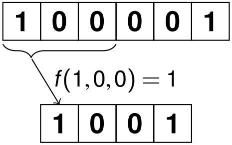
- Global rule of $\langle n, d, f \rangle$: vectorial Boolean function induced by f
- No Boundary Conditions: $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ is defined as

$$F(x_0, \cdots, x_{n-1}) = (f(x_0, \cdots, x_{d-1}), f(x_1, \cdots, x_d), \cdots, f(x_{n-d}, \cdots, x_{n-1}))$$

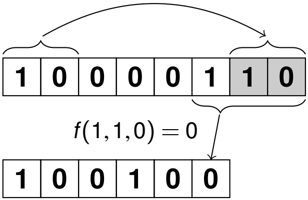
- Periodic Boundary Conditions: $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is defined as

$$F(x_0, \cdots, x_{n-1}) = (f(x_0, \cdots, x_{d-1}), f(x_1, \cdots, x_d), \cdots, f(x_{n-1}, \cdots, x_{d-2}))$$

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



No Boundary CA – NBCA



Periodic Boundary CA – PBCA

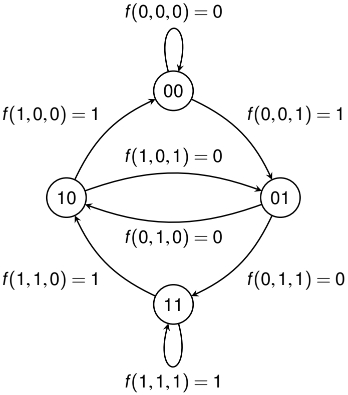
CA Local Rule Representations

- Wolfram code of f: Decimal encoding of the truth table of f

x	000	001	010	011	100	101	110	111		Code
$f(x)$	0	1	0	0	1	0	1	1	$\Rightarrow$	210

Example: $d = 3$, $f(x) = x_0 \oplus x_1 x_2 \oplus x_2$ (Keccak χ function, rule 210)

- De Bruijn graph of f: directed graph $G(V, E)$ with $V = \mathbb{F}_2^{d-1}$ and $(v_1, v_2) \in E \Leftrightarrow v_1$ and v_2 overlap on $d-2$ coordinates
- f is represented as a labeling over E



Walsh Spectrum of Permutive NBCA (1/4)

- $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is called left permutive if there is $g : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ s.t.

$$f(x_0, x_1, \cdots, x_{n-1}) = x_0 \oplus g(x_1, \cdots, x_{n-1})$$

- Example: KECCAK χ rule, $\chi(x_0, x_1, x_2) = x_0 \oplus x_1 x_2 \oplus x_3$

Theorem

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ be the global rule of a NBCA with left permutive local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$, and let $W_{v \cdot F}(\omega)$ be a Walsh coefficient of $v \cdot F$. Then, the coefficient $W_{v' \cdot F'}(\omega')$ of $v' \cdot F'$ obtained by appending a cell to the left of F is one of the following:

- $W_{v' \cdot F'}(\omega') = 0$
- $W_{v' \cdot F'}(\omega') = 2 \cdot W_{v \cdot F}(\omega)$

Walsh Spectrum of Permutive NBCA (2/4)

Proof (Idea): by induction on the number of output cells

- Base: $n = d + 1$ (2 output cells). Only three components must be checked, namely $(1,0)$, $(0,1)$ and $(1,1)$:
  - For $(1,0)$ and $(0,1)$, it suffices to split the sum of the Walsh coefficient with respect to the value of x_0:

$$W_{(0,1)\cdot F}(\omega) = \sum_{x \in \mathbb{F}_2^{n+1} : x_0 = 0} (-1)^{f(x_1,\cdots,x_n)\omega_1 x_1 \oplus \cdots \oplus \omega_n x_n} + (-1)^{\omega_0} \sum_{x \in \mathbb{F}_2^{n+1} : x_0 = 1} (-1)^{f(x_1,\cdots,x_n)\omega_1 x_1 \oplus \cdots \oplus \omega_n x_n}$$

- for $\omega_0 = 0 \Rightarrow W_{(0,1)\cdot F}(\omega) = 2 \cdot W_f(\omega_1, \cdots, \omega_n)$
- for $\omega_0 = 1 \Rightarrow W_{(0,1)\cdot F}(\omega) = 0$

Walsh Spectrum of Permutive NBCA (3/4)

Proof (Idea): by induction on the number of output cells

- Base: $n = d + 1$ (2 output cells). Only three components must be checked, namely $(1,0)$, $(0,1)$ and $(1,1)$:
  - For $(1,1)$: use left permutivity $\Rightarrow f(0, x_1, \cdots x_n) \neq f(1, x_1, \cdots, x_n)$ and again split with respect to x_0:

$$W_{(1,1)\cdot F}(\omega) = \sum_{x \in \mathbb{F}_2^{n+1} : x_0 = 0} (-1)^{f(0,x_1,\cdots,x_{n-1}) \oplus f(x_1,\cdots,x_n) \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n}$$

$$+(-1)^{\omega_0} \sum_{x \in \mathbb{F}_2^{n+1} : x_0 = 1} (-1)^{f(1,x_1,\cdots,x_{n-1}) \oplus f(x_1,\cdots,x_n) \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n}$$

- for $\omega_0 = 0 \Rightarrow W_{(0,1)\cdot F}(\omega) = 0$,
- for $\omega_0 = 1 \Rightarrow W_{(0,1)\cdot F}(\omega) = 2 \cdot W_f(\omega_1, \cdots, \omega_n)$

Walsh Spectrum of Permutive NBCA (4/4)

Proof (Idea): by induction on the number of output cells

- Induction: $F' : \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{n-d+2}$ obtained by appending a cell to the left of $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$
- The number of component functions doubles: for $v \in \mathbb{F}_2^n \{\underline{0}\}$,
  - Case $(0, v)$: Similar to the base case $(0, 1)$
    - $\omega_0 = 0 \Rightarrow W_{(0,v)\cdot F'}(\omega) = 2 \cdot W_{v \cdot F}(\omega_1, \cdots, \omega_{n+1})$
    - $\omega_0 = 1 \Rightarrow W_{(0,v)\cdot F'}(\omega) = 0$
  - Case $(1, v)$: Use again left permutivity, as in base case $(1, 1)$
    - $\omega_0 = 0 \Rightarrow W_{(1,v)\cdot F'}(\omega) = 0$
    - $\omega_0 = 1 \Rightarrow W_{(1,v)\cdot F'}(\omega) = 2 \cdot W_{v \cdot F}(\omega_1, \cdots, \omega_{n+1})$

Nonlinearity of Permutive NBCA

Corollary

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, with $m = n - d + 1$ be the global rule of a CA with left permutive local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. Then,

$$NL(F) = 2^{m-1} \cdot NL(f)$$

- Example: KECCAK χ rule: $NL(\chi) = 2$

n	4	5	6	7
$NL(F)$	4	8	16	32

- By experimental observations, the same formula seems to hold also for permutive PBCA

Outline

Cellular Automata

Experimental Results

Conclusions

Construction of S-boxes using CA Rules

- What do those results mean from the practical (cryptographic) perspective?
- How to use CA rules to construct optimal (with respect to the nonlinearity and differential uniformity property) S-boxes?
- For smaller sizes (i.e., up to 5×5) it is easy to conduct exhaustive search

Construction of S-boxes using CA Rules

Table: Results for exhaustive search

n	Number of (CA) S-boxes	Number of bijective S-boxes	Number of optimal S-boxes
3	256	36	12
4	65 536	1 536	512
5	4 294 967 296	22 500 002	2 880

Construction of S-boxes using CA Rules

- For 4×4 size, there are 512 optimal S-boxes
- However, all of them belong to only 4 optimal classes - G_3, G_4, G_5, G_6
- In each class, there are 128 S-boxes

Construction of S-boxes using CA Rules

- If exhaustive search is not possible, we can use heuristics
- Genetic programming (GP) seems to be a rather natural choice for this task
- Genetic programming is an evolutionary algorithm in which the data structures that undergo optimization are computer programs

Construction of S-boxes using CA Rules

- Since the aim of GP is to automatically generate new programs, each individual represents a computer program, where the most common are symbolic expressions representing parse trees
- A tree can represent a mathematical expression, a rule set or a decision tree
- The building elements in a tree-based GP are functions (inner nodes) and terminals (leaves, problem variables)
- Additional benefits are that we can limit the size of a tree (consequently, the size of a rule) and influence the maximal latency of the underlying S-box

Construction of S-boxes using CA Rules

Algorithm 1 Pseudocode for GP.

$Input: \ Parameters\ of\ the\ algorithm$
$Output: \ Optimal\ solution\ set$
$t \leftarrow 0$
$P(0) \leftarrow CreateInitialPopulation$
while $TerminationCriterion$ **do**
    $t \leftarrow t + 1$
    $P'(t) \leftarrow SelectMechanism\,(P(t-1))$
    $P(t) \leftarrow VariationOperators(P'(t))$
end while
$Return\ OptimalSolutionSet(P)$

CA Local Rule Optimization with Genetic Programming

- Construct a CA rule in symbolic form
- Genetic programming (GP) optimizes symbolic representation of Boolean functions
- Potential solutions represented as a graph:
  - terminal nodes (leaves) represent current state bits (s_i)
  - functional nodes are Boolean functions (AND, OR, NOT, ...)
- Indirectly search the space of S-boxes
- With GP, we are able to find optimal S-boxes for dimension 7×7 and S-boxes with differential uniformity equal to 4 for 6×6 size

Search for Reusable CA Rules

- Secondary goal: find a CA rule applicable for construction of S-boxes of varying sizes
- Assume base search dimension is given (n)
- Procedure:
  - generate candidate CA rule for size n
  - apply rule to generate S-boxes of sizes n, $n+2$, $n+4$, ...
  - assign quality measure based on properties for all considered sizes

Outline

Conclusions

- CA rules represent interesting option to build S-boxes
- We can use either CA rules that result in bijective S-boxes for a number of sizes but then cryptographic properties degrade or a CA rules resulting in optimal S-boxes for only one size
- We can conduct exhaustive search for up to 5×5 size with CA rules, which is not possible for general 5×5 S-boxes
- For larger sizes we can easily use heuristics

Questions?

Thanks for your attention!

Q?