Radboud University

Low-Latency Boolean Functions & Bijective S-boxes

Shahram Rasoolzadeh, Radboud University, Nijmegen, The Netherlands

FSE 2023, March 20, 2023

ESCADA

S-boxes

- used for substitutions, to provide confusion and non-linearity
- n- to m-bit vectorial Boolean functions

S-boxes

- used for substitutions, to provide confusion and non-linearity
- n- to m-bit vectorial Boolean functions
- $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ with $F = (f_0, \ldots, f_{m-1})$
- coordinate functions: f_i
- component functions: $F_\alpha := \langle F, \alpha \rangle \bigoplus_{i=0}^{m-1} \alpha_i f_i$ with $\alpha \in \mathbb{F}_2^m \setminus \{0\}$

S-boxes

- used for substitutions, to provide confusion and non-linearity
- n- to m-bit vectorial Boolean functions
- $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ with $F = (f_0, \ldots, f_{m-1})$
- coordinate functions: f_i
- component functions: $F_\alpha := \langle F, \alpha \rangle \bigoplus_{i=0}^{m-1} \alpha_i f_i$ with $\alpha \in \mathbb{F}_2^m \setminus \{0\}$

Balanced S-boxes

- each output value occurs uniformly, i.e., 2^{n-m} times

Introduction

S-boxes

- used for substitutions, to provide confusion and non-linearity
- n- to m-bit vectorial Boolean functions
- $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ with $F = (f_0, \ldots, f_{m-1})$
- coordinate functions: f_i
- component functions: $F_\alpha := \langle F, \alpha \rangle \bigoplus_{i=0}^{m-1} \alpha_i f_i$ with $\alpha \in \mathbb{F}_2^m \setminus \{0\}$

Balanced S-boxes

- each output value occurs uniformly, i.e., 2^{n-m} times
- equivalent to each component function is a balanced Boolean function

Cryptographic Properties

- uniformity: $$\mathrm{uni}(F) = \max_{\alpha \in \mathbb{F}_2^n \setminus \{0\},\, \beta \in \mathbb{F}_2^m} \#\{x \in \mathbb{F}_2^n \mid F(x) \oplus F(x \oplus \alpha) = \beta\}$$

Cryptographic Properties

- uniformity:
- linearity:

$$\mathrm{lin}(F) = \max_{\alpha \in \mathbb{F}_2^n,\, \beta \in \mathbb{F}_2^m \setminus \{0\}} \left| 2 \cdot \#\{x \in \mathbb{F}_2^n \mid \langle \alpha, x \rangle = F_\beta(x)\} - 2^n \right|$$

Cryptographic Properties

- uniformity:
- linearity:
- algebraic degree:

maximum number of input variables in each monomial of the ANF representation of each coordinate function

Properties

Cryptographic Properties

- uniformity:
- linearity:
- algebraic degree:

maximum number of input variables in each monomial of the ANF representation of each coordinate function

Implementation Properties

- area
- gate count
- latency
- gate depth
- power
- ...

Properties

Cryptographic Properties

- uniformity:
- linearity:
- algebraic degree:

maximum number of input variables in each monomial of the ANF representation of each coordinate function

Implementation Properties

- area
- gate count
- latency
- gate depth
- power
- ...

Implementation Complexities

- gate count comp.
- gate depth comp.
- multiplicative comp.
- ...

Equivalency

$$F = P_{out} \circ G \circ P_{in}$$

- Bit-Permutation: with P_{in} and P_{out} being bijective bit-permutation functions

$$\# \text{ Eq. Fun. } \leq n! \cdot m!$$

Equivalency

- Bit-Permutation:

$$F = P_{out} \circ G \circ P_{in}(\cdot + \alpha) + \beta$$

with P_{in} and P_{out} being bijective bit-permutation functions

- Ext. Bit-Perm.:

$$\#\ \text{Eq. Fun.} \leq n! \cdot m! \cdot 2^{n+m}$$

Equivalency

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:

$$F = L_{out} \circ G \circ L_{in}$$

with L_{in} and L_{out} being bijective linear functions

$\#$ Eq. Fun. $\leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i)$

Equivalency

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:
- Affine:

$$F = A_{out} \circ G \circ A_{in}$$

with A_{in} and A_{out} being bijective affine functions

$$\#\ \text{Eq. Fun.} \leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i) \cdot 2^{n+m}$$

Equivalency

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:
- Affine:
- Ext. Affine:

$$F = A_{out} \circ G \circ A_{in} + L$$

with A_{in} and A_{out} being bijective affine functions and L being a linear function

$$\#\text{ Eq. Fun.} \leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i) \cdot 2^{nm+n+m}$$

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:
- Affine:
- Ext. Affine:

$$F = A_{out} \circ G \circ A_{in} + L$$

with A_{in} and A_{out} being bijective affine functions and L being a linear function

$$\# \text{ Eq. Fun.} \leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i) \cdot 2^{nm+n+m}$$

Extended Affine Equivalent Examples

- linearity / uniformity
- algebraic degree (of non-linear functions)
- multiplicative count / depth complexities

Equivalency

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:
- Affine:
- Ext. Affine:

$$F = A_{out} \circ G \circ A_{in} + L$$

with A_{in} and A_{out} being bijective affine functions and L being a linear function

$$\#\text{ Eq. Fun.} \leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i) \cdot 2^{nm+n+m}$$

Bit-Permutation Equivalent Examples

- circuit implementation costs: area / latency / power
- gate depth / count complexities

- Bit-Permutation:
- Ext. Bit-Perm.:
- Linear:
- Affine:
- Ext. Affine:

$$F = A_{out} \circ G \circ A_{in} + L$$

with A_{in} and A_{out} being bijective affine functions and L being a linear function

$$\# \text{ Eq. Fun.} \leq \prod_{i=0}^{n-1}(2^n - 2^i) \cdot \prod_{i=0}^{m-1}(2^m - 2^i) \cdot 2^{nm+n+m}$$

Extended Bit-Permutation Semi-Equivalent Examples

- By accepting small tolerances,
- and due to combing small circuits/functions to build larger circuits/functions,

bit-perm. equivalent properties are extended bit-perm. semi-equivalent properties.

Latency

the time required to compute *all* the outputs of a circuit

Latency Complexity

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property
- technology-specific property

Latency Complexity

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property
- technology-specific property

Gate Depth Complexity

the minimum possible value for the longest path (concerning the number of gates used in the path) from any input to any output for implementing the function

Latency Complexity

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property
- technology-specific property

Gate Depth Complexity

the minimum possible value for the longest path (concerning the number of gates used in the path) from any input to any output for implementing the function

in the basis of all gates with fan-in number 1 or 2

Latency Complexity

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property
- technology-specific property

Gate Depth Complexity

the minimum possible value for the longest path (concerning the number of gates used in the path) from any input to any output for implementing the function

in the basis of all gates with fan-in number 1 or 2

Gate	NAND2	NOR2	AND2	OR2	XOR2	XNOR2
15 nm	2.031	2.554	3.580	3.644	5.268	6.788
45 nm	27.886	40.650	40.171	56.414	73.019	57.604

Latency Complexity

Latency

the time required to compute *all* the outputs of a circuit

- circuit-specific property
- technology-specific property

Gate Depth Complexity

the minimum possible value for the longest path (concerning the number of gates used in the path) from any input to any output for implementing the function

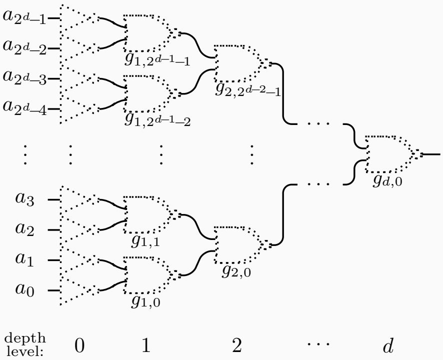
in the basis of all gates with fan-in number 1 or 2

Latency Complexity

the gate depth complexity in the basis of $\{\mathtt{NAND2}, \mathtt{NOR2}, \mathtt{INV}\}$ without counting $\mathtt{INV}$s

Proposition 1

Any Boolean function $f(x_0, \ldots, x_{n-1})$ with latency complexity d can be implemented by a circuit of the following structure:



Building Representative Functions of $\mathcal{F}_{n,d}$

Data: $\mathcal{F}_{n',d'}$ for all $n' \leq n$ and $d' < d$ // the sets of all full-dependent n'-bit Boolean functions with latency complexity d'

Result: $\mathcal{F}_{n,d}$

$\mathcal{F} \leftarrow \emptyset$ and $\mathcal{F}_{n,d} \leftarrow \emptyset$

for $n_0 \leftarrow 1$ **to** n **do**

  for $n_1 \leftarrow \max(1, n - n_0)$ **to** n_0 **do**

    foreach $d_0, d_1 \in \mathbb{Z}_d$ **do**

      if $\mathcal{F}_{n_0,d_0} \neq \emptyset$ **and** $\mathcal{F}_{n_1,d_1} \neq \emptyset$ **and** $(d_0 = d - 1$ **or** $d_1 = d - 1)$ **then**

        foreach $\pi \in \mathbb{Z}_n^{n_1}$ **if** π *follows the restrictions* **do** // for $i < j$, $\pi[i] \neq \pi[j]$, // and if $n_0 \leq \pi_1[i]$ and $n_0 \leq \pi_1[j]$, then $\pi_1[i] < \pi_1[j]$.

          Compute the corresponding bit-permutation function P.

          foreach $\alpha \in \mathbb{F}_2^{n_1}$ **if** α *follows the restrictions* **do** // for each i such that $n_0 \leq \pi_1[i]$, $\alpha_1[i]$ must be 0.

            foreach $f_0^* \in \mathcal{F}_{n_0,d_0}$ **and** $f_1^* \in \mathcal{F}_{n_1,d_1}$ **do**

              $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_0^* \barwedge f_1^*(P(\cdot) \oplus \alpha)\}$

foreach $f \in \mathcal{F}$ **do**

  $\mathcal{F}_{n,d} \leftarrow \mathcal{F}_{n,d} \cup \{\textsc{ComputeRepresentative}(f)\}$

for $n' \leftarrow 1$ **to** n **do**

  for $d' \leftarrow 0$ **to** d **do**

    if $(n', d') \neq (n, d)$ **then**

      $\mathcal{F}_{n,d} \leftarrow \mathcal{F}_{n,d} - \mathcal{F}_{n',d'}$

Number of Representative Functions in $\mathcal{F}_{n,d}$

d/n	2	3	4	5	6	7	8
1	1						
2	1	3	3				
3		5	54	159	170	64	20
4		2	149	109 674	20 658 457	227 737 882	?
5			2	506 005	?	?	?
6				66	?	?	?

Balanced

d/n	2	3	4	5	6	7	8
2	1	1					
3		2	6	8	3		
4		1	45	12 128	931 780	4 436 770	4 489 235
5			1	74 389	?	?	?
6				30	?	?	?

Finding All Possible Lowest-Depth Implementations of a Function

Data: $\mathcal{F}^*_{d'}$ for all d' `// all sets of` n`-bit representative Boolean functions (not necessarily full-dependent) with latency complexity` d'

Result: $\mathcal{I}$ `// all possible implementations of the given Boolean function`

Function FindAllCircuits(f):
- **if** $f = x_i$ *or* $f = \neg x_i$ **then** `// the case for latency complexity of` $d' = 0$
  - **return** the corresponding x_i or $\neg x_i$
- $\mathcal{I} \leftarrow \emptyset$, $\mathcal{A}_{\overline{\wedge}} \leftarrow \emptyset$ and $\mathcal{A}_{\overline{\vee}} \leftarrow \emptyset$
- **for** $d' \leftarrow 0$ **to** $d - 1$ **do**
  - **foreach** $g \in \{g \,|\, g \text{ is equivalent to } g^* \in \mathcal{F}^*_{d'}\}$ **do**
    - **if** $\neg f \wedge g = \neg f$ **then**
      - $\mathcal{A}_{\overline{\wedge}} \leftarrow \mathcal{A}_{\overline{\wedge}} \cup \{g\}$
    - **if** $\neg f \vee g = \neg f$ **then**
      - $\mathcal{A}_{\overline{\vee}} \leftarrow \mathcal{A}_{\overline{\vee}} \cup \{g\}$
  - **foreach** $g, h \in \mathcal{A}_{\overline{\wedge}}$ **do**
    - **if** $g \wedge h = \neg f$ **then**
      - $\mathcal{I} \leftarrow \mathcal{I} \cup \{(\text{FindAllCircuits}(g), \text{FindAllCircuits}(h), \texttt{NAND})\}$
  - **foreach** $g, h \in \mathcal{A}_{\overline{\vee}}$ **do**
    - **if** $g \vee h = \neg f$ **then**
      - $\mathcal{I} \leftarrow \mathcal{I} \cup \{(\text{FindAllCircuits}(g), \text{FindAllCircuits}(h), \texttt{NOR})\}$
  - **if** $\mathcal{I} \neq \emptyset$ **then** `// this means that the latency complexity of` f `is` $d' + 1$
    - **return** $\mathcal{I}$

Latency Complexity of Previously Known S-boxes

- 3-bit: $d = 3$ for all of them.

Latency Complexity of Previously Known S-boxes

- 3-bit: $d = 3$ for all of them.
- 4-bit:

  except for `Midori`-s_0 and χ_4 with $d = 3$,

  others have $d = 4$ or $d = 5$.

Latency Complexity of Previously Known S-boxes

- 3-bit: $d = 3$ for all of them.
- 4-bit:

  except for $\texttt{Midori-}s_0$ and χ_4 with $d = 3$,

  others have $d = 4$ or $d = 5$.
- 5-bit:

  χ_5 has $d = 3$ with $\ell = 16$ and $u = 8$,

  while APN S-boxes such as `Fides-5` have $d = 5$.

Latency Complexity of Previously Known S-boxes

- 3-bit: $d = 3$ for all of them.
- 4-bit:

  except for $\texttt{Midori-}s_0$ and χ_4 with $d = 3$,

  others have $d = 4$ or $d = 5$.
- 5-bit:

  χ_5 has $d = 3$ with $\ell = 16$ and $u = 8$,

  while APN S-boxes such as $\texttt{Fides-5}$ have $d = 5$.
- 6-bit:

  χ_5 has $d = 3$ with $\ell = 32$ and $u = 16$,

  $\texttt{Speedy}$ S-box has $d = 4$ with $\ell = 24$ and $u = 8$.

Latency Complexity of Previously Known S-boxes

- 3-bit: $d = 3$ for all of them.
- 4-bit:

  except for $\texttt{Midori-}s_0$ and χ_4 with $d = 3$,

  others have $d = 4$ or $d = 5$.
- 5-bit:

  χ_5 has $d = 3$ with $\ell = 16$ and $u = 8$,

  while APN S-boxes such as `Fides-5` have $d = 5$.
- 6-bit:

  χ_5 has $d = 3$ with $\ell = 32$ and $u = 16$,

  `Speedy` S-box has $d = 4$ with $\ell = 24$ and $u = 8$.
- 7- and 8-bit:

  except χ_7 and χ_8 there is no S-box with $d \leq 5$.

Building S-boxes with Low-Latency Complexity

Criteria

1. latency complexity
2. linearity
3. uniformity
4. algebraic degree (quadratic or maximum)

Building S-boxes with Low-Latency Complexity

Criteria

1. latency complexity
2. linearity
3. uniformity
4. algebraic degree (quadratic or maximum)

Method of Building S-boxes:

- Stepping over the coordinates of S-box

Building S-boxes with Low-Latency Complexity

Criteria

1. latency complexity
2. linearity
3. uniformity
4. algebraic degree (quadratic or maximum)

Method of Building S-boxes:

- Stepping over the coordinates of S-box
- $S = (f_0, f_1, \ldots, f_{n-1})$

Criteria

1. latency complexity
2. linearity
3. uniformity
4. algebraic degree (quadratic or maximum)

Method of Building S-boxes:

- Stepping over the coordinates of S-box
- $S = (f_0, f_1, \ldots, f_{n-1})$
- $\mathcal{F}$: the set of all Boolean functions satisfying the criteria 1, 2 (and 4)
- $\mathcal{R}_1$: the set of all representatives from $\mathcal{F}$

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \,\|\, f$.

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \parallel f$.
   - 2a. If F fulfills the criteria, add it to $\mathcal{S}$.

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \| f$.
   - 2a. If F fulfills the criteria, add it to $\mathcal{S}$.
3. For each $F_1 \in \mathcal{S}$, set $\texttt{new} = 1$

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \,\|\, f$.
   - 2a. If F fulfills the criteria, add it to $\mathcal{S}$.
3. For each $F_1 \in \mathcal{S}$, set $\texttt{new} = 1$
   - 3a. For each $F_0 \in \mathcal{R}_i$, check if F_1 is equivalent to F_0.

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \| f$.
   - 2a. If F fulfills the criteria, add it to $\mathcal{S}$.
3. For each $F_1 \in \mathcal{S}$, set $\texttt{new} = 1$
   - 3a. For each $F_0 \in \mathcal{R}_i$, check if F_1 is equivalent to F_0.
     - If they are equivalent, set $\texttt{new} = 0$ and break the loop.

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \,\|\, f$.
   - 2a. If F fulfills the criteria, add it to $\mathcal{S}$.
3. For each $F_1 \in \mathcal{S}$, set $\texttt{new} = 1$
   - 3a. For each $F_0 \in \mathcal{R}_i$, check if F_1 is equivalent to F_0.
     - If they are equivalent, set $\texttt{new} = 0$ and break the loop.
   - 3b. If $\texttt{new} = 1$, add F_1 to $\mathcal{R}_i$.

Building S-boxes with Steps on Coordinates [Can07]

1. Set $i = 2$, and $\mathcal{S} = \mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \| f$.
   - 2a If F fulfills the criteria, add it to $\mathcal{S}$.
3. For each $F_1 \in \mathcal{S}$, set $\texttt{new} = 1$
   - 3a For each $F_0 \in \mathcal{R}_i$, check if F_1 is equivalent to F_0.
     - If they are equivalent, set $\texttt{new} = 0$ and break the loop.
   - 3b If $\texttt{new} = 1$, add F_1 to $\mathcal{R}_i$.
4. Increase i, if $i \leq m$. Set $\mathcal{S} = \emptyset$ and go to step 4.

Building S-boxes with Steps on Coordinates (Improved 1)

1. Set $i = 2$, and $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \parallel f$.

Building S-boxes with Steps on Coordinates (Improved 1)

1. Set $i = 2$, and $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \parallel f$.
   - If F fulfills the criteria, compute its representative and add it to $\mathcal{R}_i$.

Building S-boxes with Steps on Coordinates (Improved 1)

1. Set $i = 2$, and $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \parallel f$.
   - If F fulfills the criteria, compute its representative and add it to $\mathcal{R}_i$.
3. Increase i, if $i \leq m$ go to step 4.

Building S-boxes with Steps on Coordinates (Improved 1)

1. Set $i = 2$, and $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $F' \in \mathcal{R}_{i-1}$ and each $f \in \mathcal{F}$, compute $F = F' \,\|\, f$.
   - If F fulfills the criteria, compute its representative and add it to $\mathcal{R}_i$.
3. Increase i, if $i \leq m$ go to step 4.

Improved Complexity

$$Time = \sum_{i=1}^{m-1} |\mathcal{F}| \cdot |\mathcal{R}_i| \cdot (t_{criteria\ check} + p_i \cdot t_{representative\ computation})$$

$$Memory = |\mathcal{F}| + \max_i |\mathcal{R}_i|$$

Building S-boxes with Steps on Coordinates (Improved 2)

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \parallel f_1$.

Building S-boxes with Steps on Coordinates (Improved 2)

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \,\|\, f_1$.
   - 2a If F_2 fulfills the criteria, compute its representative and add it to $\mathcal{R}_2$, otherwise, choose another f_1.

Building S-boxes with Steps on Coordinates (Improved 2)

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \,||\, f_1$.
   - 2a. If F_2 fulfills the criteria, compute its representative and add it to $\mathcal{R}_2$, otherwise, choose another f_1.
   - 2b. For each $f_2 \in \mathcal{F}$, compute $F_3 = F_2 \,||\, f_2$.

Building S-boxes with Steps on Coordinates (Improved 2)

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \,\|\, f_1$.
   - 2a. If F_2 fulfills the criteria, compute its representative and add it to $\mathcal{R}_2$, otherwise, choose another f_1.
   - 2b. For each $f_2 \in \mathcal{F}$, compute $F_3 = F_2 \,\|\, f_2$.
     - If F_3 fulfills the criteria, compute its representative and add it to $\mathcal{R}_3$, otherwise, choose another f_2.

Building S-boxes with Steps on Coordinates (Improved 2)

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \,\|\, f_1$.
   - 2a. If F_2 fulfills the criteria, compute its representative and add it to $\mathcal{R}_2$, otherwise, choose another f_1.
   - 2b. For each $f_2 \in \mathcal{F}$, compute $F_3 = F_2 \,\|\, f_2$.
     - If F_3 fulfills the criteria, compute its representative and add it to $\mathcal{R}_3$, otherwise, choose another f_2.
     - For each $f_3 \in \mathcal{F}$, compute $F_4 = F_3 \,\|\, f_3 \quad \ldots$

1. Set $\mathcal{R}_2 = \cdots = \mathcal{R}_m = \emptyset$
2. For each $f_0 \in \mathcal{R}_1$ and each $f_1 \in \mathcal{F}$, compute $F_2 = f_0 \,\|\, f_1$.
   - 2a If F_2 fulfills the criteria, compute its representative and add it to $\mathcal{R}_2$, otherwise, choose another f_1.
   - 2b For each $f_2 \in \mathcal{F}$, compute $F_3 = F_2 \,\|\, f_2$.
     - If F_3 fulfills the criteria, compute its representative and add it to $\mathcal{R}_3$, otherwise, choose another f_2.
     - For each $f_3 \in \mathcal{F}$, compute $F_4 = F_3 \,\|\, f_3 \quad \ldots$

$$\begin{aligned} Time &= |\mathcal{R}_1| \cdot |\mathcal{F}| \cdot (t_0 + p_1 \cdot t_1 + p_1 \cdot |\mathcal{F}| \cdot (t_0 + p_2 \cdot t_1 + p_2 \cdot |\mathcal{F}| \cdot (\ldots))) \\ &= |\mathcal{R}_1| \cdot |\mathcal{F}| \cdot \big(t_0 \cdot (1 + p_1|\mathcal{F}| + p_1 p_2 |\mathcal{F}|^2 + \ldots) + t_1 p_1 \cdot (1 + p_2|\mathcal{F}| + \ldots)\big) \end{aligned}$$

$$Memory = |\mathcal{F}| + \sum_{i=1}^{m} |\mathcal{R}_i|$$

4. For each $f_0 \in \mathcal{R}_1$
   - 4a. Compute $\mathcal{F}_1^\dagger = \{f_1 \in \mathcal{F} \mid (f_0 \parallel f_1) \text{ fulfills the criteria}\}$

Building S-boxes with Steps on Coordinates (Improved 3)

4. For each $f_0 \in \mathcal{R}_1$
   - 4a. Compute $\mathcal{F}_1^\dagger = \{f_1 \in \mathcal{F} \mid (f_0 \parallel f_1) \text{ fulfills the criteria}\}$
   - 4b. For each $f_1 \in \mathcal{F}_1^\dagger$
     1. Compute representative of $(f_0 \parallel f_1)$ and add it to $\mathcal{R}_2$

4. For each $f_0 \in \mathcal{R}_1$
   - 4a. Compute $\mathcal{F}_1^\dagger = \{f_1 \in \mathcal{F} \mid (f_0 \parallel f_1) \text{ fulfills the criteria}\}$
   - 4b. For each $f_1 \in \mathcal{F}_1^\dagger$
     1. Compute representative of $(f_0 \parallel f_1)$ and add it to $\mathcal{R}_2$
     2. Compute $\mathcal{F}_2^\dagger = \{f_2 \in \mathcal{F}_1^\dagger \mid (f_0 \parallel f_1 \parallel f_2) \text{ fulfills the criteria}\}$

Building S-boxes with Steps on Coordinates (Improved 3)

4. For each $f_0 \in \mathcal{R}_1$
   - 4a. Compute $\mathcal{F}_1^\dagger = \{f_1 \in \mathcal{F} \mid (f_0 \parallel f_1) \text{ fulfills the criteria}\}$
   - 4b. For each $f_1 \in \mathcal{F}_1^\dagger$
     1. Compute representative of $(f_0 \parallel f_1)$ and add it to $\mathcal{R}_2$
     2. Compute $\mathcal{F}_2^\dagger = \{f_2 \in \mathcal{F}_1^\dagger \mid (f_0 \parallel f_1 \parallel f_2) \text{ fulfills the criteria}\}$
     3. For each $f_2 \in \mathcal{F}_2^\dagger$

        $\ldots$

Building S-boxes with Steps on Coordinates (Improved 3)

4. For each $f_0 \in \mathcal{R}_1$
   - 4a. Compute $\mathcal{F}_1^\dagger = \{f_1 \in \mathcal{F} \mid (f_0 \parallel f_1) \text{ fulfills the criteria}\}$
   - 4b. For each $f_1 \in \mathcal{F}_1^\dagger$
     1. Compute representative of $(f_0 \parallel f_1)$ and add it to $\mathcal{R}_2$
     2. Compute $\mathcal{F}_2^\dagger = \{f_2 \in \mathcal{F}_1^\dagger \mid (f_0 \parallel f_1 \parallel f_2) \text{ fulfills the criteria}\}$
     3. For each $f_2 \in \mathcal{F}_2^\dagger$

        ...

$$\begin{aligned} Time &= |\mathcal{R}_1| \cdot |\mathcal{F}| \cdot (t_0 + p_1 \cdot t_1 + p_1 \cdot |\mathcal{F}_1^\dagger| \cdot (t_0 + p_2 \cdot t_1 + p_2 \cdot |\mathcal{F}_2^\dagger| \cdot (\ldots))) \\ &= |\mathcal{R}_1| \cdot |\mathcal{F}| \cdot \left(t_0 \cdot (1 + p_1^2|\mathcal{F}| + p_1^2 p_2^2 |\mathcal{F}|^2 + \ldots) + t_1 p_1 \cdot (1 + p_1 p_2 |\mathcal{F}| + \ldots)\right) \end{aligned}$$

More Speed-Up Techniques

Ordering Coordinates

- We are interested to find the functions up-to the selected equivalency.
- We can order the coordinates due to the output mapping in equivalency equation.

Ordering Coordinates

- We are interested to find the functions up-to the selected equivalency.
- We can order the coordinates due to the output mapping in equivalency equation.

Case of Extended Bit-Permutation Equivalence

$$f_0 < f_1 < \ldots < f_{m-1}$$

More Speed-Up Techniques

Ordering Coordinates

- We are interested to find the functions up-to the selected equivalency.
- We can order the coordinates due to the output mapping in equivalency equation.

Case of Extended Bit-Permutation Equivalence

$$f_0 < f_1 < \ldots < f_{m-1}$$

Criteria Check

- Linearity: we only need to check linearity of 2^{x-2} Boolean functions.

More Speed-Up Techniques

Ordering Coordinates

- We are interested to find the functions up-to the selected equivalency.
- We can order the coordinates due to the output mapping in equivalency equation.

Case of Extended Bit-Permutation Equivalence

$$f_0 < f_1 < \ldots < f_{m-1}$$

Criteria Check

- Linearity: we only need to check linearity of 2^{x-2} Boolean functions.
- Uniformity: $\quad \mathrm{uni}(S_x) \leq 2^{m-x} \cdot \mathrm{uni}(S_x \| S_{m-x})$

Results

Latency Complexity $d = 2$

- 3-bit: 2 S-boxes with $\ell = 8$ and $u = 4$
- 4-bit: 1 S-box with $\ell = u = 16$
- 5-bit: 13 S-boxes with $\ell = u = 32$
- 6-bit: 19 S-boxes with $\ell = u = 64$
- 7-bit: 125 S-boxes with $\ell = u = 128$
- 8-bit: 181 S-boxes with $\ell = u = 256$

Latency Complexity $d = 3$

- 3-bit: all the S-boxes
- 4-bit: 281 S-boxes with $\ell = 8$ and $u = 4$
- 5-bit: 13 S-boxes with $\ell = 16$ and $u = 6$
- 6-bit: 49 quadratic S-boxes with $\ell = 32$ and $u = 16$
- 7-bit: 10 quadratic S-boxes with $\ell = 64$ and $u = 32$
- 8-bit: 84 quadratic S-boxes with $\ell = 128$ and $u = 64$

Latency Complexity $d = 4$

- 4-bit: all the Golden S-boxes: $\ell = 8$ and $u = 4$
- 5-bit: 2510 APN S-boxes: $\ell = 8$ and $u = 2$
- 6-bit: 908 quadratic S-boxes with $\ell = 16$ and $u = 4$
  together with one cubic S-box with $\ell = 16$ and $u = 4$ used in BipBip TBC
- 7-bit: 134 quadratic S-boxes with $\ell = 32$ and $u = 16$

Summary

- an algorithm to find all the Boolean functions with low-latency complexity
- an algorithm to find all the possible circuit corresponding to the latency complexity
- an improved algorithm for building S-boxes up-to the given criteria
- several low-latency S-boxes with good cryptographic properties
- an algorithm to optimize the theoretical circuit for real ASIC implementation

Summary

- an algorithm to find all the Boolean functions with low-latency complexity
- an algorithm to find all the possible circuit corresponding to the latency complexity
- an improved algorithm for building S-boxes up-to the given criteria
- several low-latency S-boxes with good cryptographic properties
- an algorithm to optimize the theoretical circuit for real ASIC implementation

`https://gitlab.science.ru.nl/shahramr/LowLatencySBoxes.git`

Summary

- an algorithm to find all the Boolean functions with low-latency complexity
- an algorithm to find all the possible circuit corresponding to the latency complexity
- an improved algorithm for building S-boxes up-to the given criteria
- several low-latency S-boxes with good cryptographic properties
- an algorithm to optimize the theoretical circuit for real ASIC implementation

`https://gitlab.science.ru.nl/shahramr/LowLatencySBoxes.git`

Thank you for your attention!