Optimizing S-box Implementations for Several Criteria using SAT Solvers

Ko Stoffelen

iCIS | Digital Security
Radboud University

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)
- Bitslice gate complexity
  Use only `AND`, `OR`, `XOR`, `NOT` (bitsliced software)

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)
- Bitslice gate complexity
  Use only `AND`, `OR`, `XOR`, `NOT` (bitsliced software)
- Gate complexity
  Also use `NAND`, `NOR`, `XNOR` (hardware, area)

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)
- Bitslice gate complexity
  Use only `AND`, `OR`, `XOR`, `NOT` (bitsliced software)
- Gate complexity
  Also use `NAND`, `NOR`, `XNOR` (hardware, area)
- Circuit depth complexity
  Trade-off #gates and depth (hardware, latency)

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)
- Bitslice gate complexity
  Use only `AND`, `OR`, `XOR`, `NOT` (bitsliced software)
- Gate complexity
  Also use `NAND`, `NOR`, `XNOR` (hardware, area)
- Circuit depth complexity
  Trade-off #gates and depth (hardware, latency)

Goal

Provably minimal implementations of small functions with respect to:

- Multiplicative complexity
  Minimize nonlinear operations (masking, MPC, FHE)
- Bitslice gate complexity
  Use only `AND`, `OR`, `XOR`, `NOT` (bitsliced software)
- Gate complexity
  Also use `NAND`, `NOR`, `XNOR` (hardware, area)
- Circuit depth complexity
  Trade-off #gates and depth (hardware, latency)

Generic solution: encode as SAT instance, solve, retrieve implementation

Encoding the MCDP

Multiplicative Complexity Decision Problem

> *Given a function f and some positive integer k, is there a circuit that implements f and that uses at most k nonlinear operations?*

Encoding the MCDP

Multiplicative Complexity Decision Problem

> *Given a function f and some positive integer k, is there a circuit that implements f and that uses at most k nonlinear operations?*

Encoding by Courtois, Mourouzis, and Hulme [CMH13, Mou15]

- Let x_i be variables representing S-box inputs
- Let y_i be variables representing S-box outputs
- Let q_i be variables representing gate inputs
- Let t_i be variables representing gate outputs
- Let a_i be variables representing wiring between gates

Encoding the MCDP

Multiplicative Complexity Decision Problem

> *Given a function f and some positive integer k, is there a circuit that implements f and that uses at most k nonlinear operations?*

Encoding by Courtois, Mourouzis, and Hulme [CMH13, Mou15]

- Let x_i be variables representing S-box inputs
- Let y_i be variables representing S-box outputs
- Let q_i be variables representing gate inputs
- Let t_i be variables representing gate outputs
- Let a_i be variables representing wiring between gates

For example, lets encode a 4x4 S-box with $k = 3$

Encoding the MCDP (2)

$$
\begin{aligned}
q_0 &= a_0 + a_1 \cdot x_0 + a_2 \cdot x_1 + a_3 \cdot x_2 + a_4 \cdot x_3 \\
q_1 &= a_5 + a_6 \cdot x_0 + a_7 \cdot x_1 + a_8 \cdot x_2 + a_9 \cdot x_3 \\
t_0 &= q_0 \cdot q_1 \\
q_2 &= a_{10} + a_{11} \cdot x_0 + a_{12} \cdot x_1 + a_{13} \cdot x_2 + a_{14} \cdot x_3 + a_{15} \cdot t_0 \\
q_3 &= a_{16} + a_{17} \cdot x_0 + a_{18} \cdot x_1 + a_{19} \cdot x_2 + a_{20} \cdot x_3 + a_{21} \cdot t_0 \\
t_1 &= q_2 \cdot q_3 \\
q_4 &= a_{22} + a_{23} \cdot x_0 + a_{24} \cdot x_1 + a_{25} \cdot x_2 + a_{26} \cdot x_3 + a_{27} \cdot t_0 + a_{28} \cdot t_1 \\
q_5 &= a_{29} + a_{30} \cdot x_0 + a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33} \cdot x_3 + a_{34} \cdot t_0 + a_{35} \cdot t_1 \\
t_2 &= q_4 \cdot q_5 \\
y_0 &= a_{36} \cdot x_0 + a_{37} \cdot x_1 + a_{38} \cdot x_2 + a_{39} \cdot x_3 + a_{40} \cdot t_0 + a_{41} \cdot t_1 + a_{42} \cdot t_2 \\
y_1 &= a_{43} \cdot x_0 + a_{44} \cdot x_1 + a_{45} \cdot x_2 + a_{46} \cdot x_3 + a_{47} \cdot t_0 + a_{48} \cdot t_1 + a_{49} \cdot t_2 \\
y_2 &= a_{50} \cdot x_0 + a_{51} \cdot x_1 + a_{52} \cdot x_2 + a_{53} \cdot x_3 + a_{54} \cdot t_0 + a_{55} \cdot t_1 + a_{56} \cdot t_2 \\
y_3 &= a_{57} \cdot x_0 + a_{58} \cdot x_1 + a_{59} \cdot x_2 + a_{60} \cdot x_3 + a_{61} \cdot t_0 + a_{62} \cdot t_1 + a_{63} \cdot t_2
\end{aligned}
$$

Encoding the MCDP (3)

Not bound to specific S-box yet

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially
- Rename x_i, y_i, q_i, t_i, but not a_i

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially
- Rename x_i, y_i, q_i, t_i, but not a_i
- Concatenate

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially
- Rename x_i, y_i, q_i, t_i, but not a_i
- Concatenate
- Add lookup table (constant equation per bit)

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially
- Rename x_i, y_i, q_i, t_i, but not a_i
- Concatenate
- Add lookup table (constant equation per bit)

Encoding the MCDP (3)

Not bound to specific S-box yet

- Consider S-box as lookup table with 2^n entries (x, y)
- A correct circuit works for all possible values
- Create 2^n copies of equations
  Note: unfortunately this encoding grows exponentially
- Rename x_i, y_i, q_i, t_i, but not a_i
- Concatenate
- Add lookup table (constant equation per bit)

These equations are in ANF, but SAT solvers require CNF

Use method by Bard et al. for converting sparse systems of low-degree multivariate polynomials [BCJ07]

Multiplicative complexity results

S-box	Size *nxm*	Multiplicative complexity
Ascon	5x5	5
ICEPOLE	5x5	6
Keccak/Ketje/Keyak	5x5	5
PRIMATEs	5x5	$\in \{6, 7\}$
PRIMATEs^{-1}	5x5	$\in \{6, 7, 8, 9, 10\}$
Joltik/Piccolo	4x4	4
Joltik^{-1}/Piccolo^{-1}	4x4	4
LAC	4x4	4
Minalpher	4x4	5
Prøst	4x4	4
RECTANGLE	4x4	4
RECTANGLE^{-1}	4x4	4

Bitslice gate complexity

Given a function f and some positive integer k, is there a circuit with only gates $\in \{\mathtt{AND}, \mathtt{OR}, \mathtt{XOR}, \mathtt{NOT}\}$ that implements f and that uses at most k gates?

Bitslice gate complexity

Given a function f and some positive integer k, is there a circuit with only gates $\in \{\mathtt{AND}, \mathtt{OR}, \mathtt{XOR}, \mathtt{NOT}\}$ that implements f and that uses at most k gates?

Idea

Hard-code k gates of unknown type:

- Let b_i be variables representing wiring *inside* gates

Bitslice gate complexity

> *Given a function f and some positive integer k, is there a circuit with only gates $\in \{\mathit{AND}, \mathit{OR}, \mathit{XOR}, \mathit{NOT}\}$ that implements f and that uses at most k gates?*

Idea

Hard-code k gates of unknown type:

- Let b_i be variables representing wiring *inside* gates
- $t_0 = b_0 \cdot q_0 \cdot q_1 + b_1 \cdot q_0 + b_1 \cdot q_1 + b_2 + b_2 \cdot q_0$
  $0 = b_0 \cdot b_2$
  $0 = b_1 \cdot b_2$

Bitslice gate complexity

Given a function f and some positive integer k, is there a circuit with only gates $\in \{\mathtt{AND}, \mathtt{OR}, \mathtt{XOR}, \mathtt{NOT}\}$ that implements f and that uses at most k gates?

Idea

Hard-code k gates of unknown type:

- Let b_i be variables representing wiring *inside* gates
- $t_0 = b_0 \cdot q_0 \cdot q_1 + b_1 \cdot q_0 + b_1 \cdot q_1 + b_2 + b_2 \cdot q_0$
  $0 = b_0 \cdot b_2$
  $0 = b_1 \cdot b_2$

Bitslice gate complexity

Given a function f and some positive integer k, is there a circuit with only gates $\in \{\mathtt{AND}, \mathtt{OR}, \mathtt{XOR}, \mathtt{NOT}\}$ that implements f and that uses at most k gates?

Idea

Hard-code k gates of unknown type:

- Let b_i be variables representing wiring *inside* gates
- $t_0 = b_0 \cdot q_0 \cdot q_1 + b_1 \cdot q_0 + b_1 \cdot q_1 + b_2 + b_2 \cdot q_0$
  $0 = b_0 \cdot b_2$
  $0 = b_1 \cdot b_2$

Gate input q_i can be precisely one:

- S-box input bit
- Previous gate output bit

Linear combination with additional at-most-1 constraints

Bitslice gate complexity results

S-box	Bitslice gate complexity	Implementation
Keccak/Ketje/Keyak	≤ 13	3 AND, 2 OR, 5 XOR, 3 NOT
Joltik/Piccolo	10	1 AND, 3 OR, 4 XOR, 2 NOT
Joltik^{-1}/Piccolo^{-1}	10	1 AND, 3 OR, 4 XOR, 2 NOT
LAC	11	2 AND, 2 OR, 6 XOR, 1 NOT
Minalpher	≥ 11	
Prøst	8	4 AND, 4 XOR
RECTANGLE	$\in \{11, 12\}$	1 AND, 3 OR, 7 XOR, 1 NOT
RECTANGLE^{-1}	$\in \{10, 11, 12\}$	4 OR, 7 XOR, 1 NOT

Gate complexity

Only difference:
$t_0 = b_0 \cdot q_0 \cdot q_1 + b_1 \cdot q_0 + b_1 \cdot q_1 + b_2$

$b_{3i} b_{3i+1} b_{3i+2}$	Gate t_i function
000	0
001	1
010	$q_{2i} \oplus q_{2i+1}$
011	$q_{2i} \leftrightarrow q_{2i+1}$
100	$q_{2i} \wedge q_{2i+1}$
101	$q_{2i} \uparrow q_{2i+1}$
110	$q_{2i} \vee q_{2i+1}$
111	$q_{2i} \downarrow q_{2i+1}$

Gate complexity results

S-box	Gate complexity	Implementation
Joltik/Piccolo	8	2 OR, 1 XOR, 2 NOR, 3 XNOR
$\text{Joltik}^{-1}/\text{Piccolo}^{-1}$	8	2 OR, 1 XOR, 2 NOR, 3 XNOR
LAC	10	1 AND, 3 OR, 2 XOR, 4 XNOR
Prøst	8	4 AND, 4 XOR
RECTANGLE	$\in \{10, 11\}$	1 AND, 1 OR, 2 XOR, 1 NAND, 1 NOR, 5 XNOR
RECTANGLE^{-1}	$\in \{10, 11\}$	1 AND, 1 OR, 6 XOR, 1 NAND, 1 NOR, 1 XNOR

Circuit depth complexity

Decreasing the depth of a circuit allows for increasing the clock frequency

Circuit depth complexity

Decreasing the depth of a circuit allows for increasing the clock frequency

Every function can be implemented in depth 2 (normal forms)
However, at the cost of more gates (width)

Circuit depth complexity

Decreasing the depth of a circuit allows for increasing the clock frequency

Every function can be implemented in depth 2 (normal forms)
However, at the cost of more gates (width)

Idea
Introduce maximum width w

> *Given a function f and some positive integer k, is there a circuit of depth at most k and width at most w that implements f?*

Encoding like gate complexity, but gate input q_i is now either S-box input or gate output *on previous depth layer*

Circuit depth complexity results

S-box	k	w	Implementation	UNSAT bounds
Joltik/Piccolo	4	2	2 OR, 1 XOR, 2 NOR, 3 XNOR	$k=4, w=1$ $k=3, w=10$
Joltik^{-1}/Piccolo^{-1}	4	3	3 OR, 5 XOR, 1 NOR, 3 XNOR	$k=4, w=2$ $k=3, w=10$
LAC	3	6	3 OR, 4 XOR, 4 NAND, 4 XNOR	$k=3, w=4$ $k=2, w=10$
Prøst	4	3	4 AND, 1 OR, 4 XOR, 1 NAND, 1 XNOR	$k=4, w=2$ $k=3, w=10$
RECTANGLE	3	6	2 AND, 3 OR, 5 XOR, 1 NAND, 1 NOR, 3 XNOR	$k=3, w=4$ $k=2, w=10$
RECTANGLE^{-1}	3	6	1 OR, 8 XOR, 3 NAND, 2 NOR, 2 XNOR	$k=3, w=4$ $k=2, w=10$

Combining criteria

What about combinations?

Combining criteria

What about combinations?

Lets optimize PRIMATEs 5x5 S-box

- First for multiplicative complexity
- Then reduce linear gates, i.e. `XOR`, `NOT`

Combining criteria

What about combinations?

Lets optimize PRIMATEs 5x5 S-box

- First for multiplicative complexity
- Then reduce linear gates, i.e. `XOR`, `NOT`

Apply existing methods for solving the Shortest Linear Straight-Line Program (SLP) problem

- Exact, using SAT solvers (Fuhs–Schneider-Kamp [FSK10])
- Heuristics (Boyar–Peralta [BP10])

SLP

Given $\mathbb{F}$ and constants $a_{i,j} \in \mathbb{F}$, compute linear forms

$$
\begin{gathered}
a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n \\
a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n \\
\ldots \\
a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n
\end{gathered}
$$

in the shortest number of program lines of the form

$$u := \lambda v + \mu w$$

where $\lambda, \mu \in \mathbb{F}$

Optimizing PRIMATEs S-box (1)

$$
\begin{aligned}
q_0 &= x_0 \oplus x_3 \\
q_1 &= x_1 \\
t_0 &= q_0 \vee q_1 \\
q_2 &= \neg(x_1 \oplus x_3) \\
q_3 &= x_0 \oplus x_2 \\
t_1 &= q_2 \wedge q_3 \\
q_4 &= x_0 \oplus x_1 \oplus x_4 \\
q_5 &= x_0 \oplus x_2 \oplus x_3 \\
t_2 &= q_4 \wedge q_5 \\
q_6 &= \neg(x_0 \oplus x_2 \oplus x_3 \oplus x_4) \\
q_7 &= x_1 \oplus x_2 \oplus x_4 \\
t_3 &= q_6 \vee q_7 \\
q_8 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4
\end{aligned}
$$

$$
\begin{aligned}
q_9 &= x_2 \oplus t_0 \oplus t_3 \\
t_4 &= q_8 \wedge q_9 \\
q_{10} &= x_0 \oplus x_3 \oplus x_4 \\
q_{11} &= \neg(x_0 \oplus x_4) \\
t_5 &= q_{10} \vee q_{11} \\
q_{12} &= \neg(x_1 \oplus x_2 \oplus t_0 \oplus t_2 \oplus t_3 \oplus t_4) \\
q_{13} &= x_2 \oplus x_3 \\
t_6 &= q_{12} \wedge q_{13} \\
y_0 &= x_1 \oplus x_3 \oplus t_2 \oplus t_3 \oplus t_5 \oplus t_6 \\
y_1 &= x_0 \oplus x_4 \oplus t_1 \oplus t_2 \oplus t_3 \oplus t_4 \oplus t_5 \oplus t_6 \\
y_2 &= x_1 \oplus x_2 \oplus x_4 \oplus t_1 \oplus t_3 \oplus t_4 \oplus t_5 \\
y_3 &= x_0 \oplus x_2 \oplus x_3 \oplus x_4 \oplus t_3 \oplus t_4 \oplus t_5 \oplus t_6 \\
y_4 &= \neg(x_2 \oplus t_0 \oplus t_2 \oplus t_3 \oplus t_4 \oplus t_5 \oplus t_6)
\end{aligned}
$$

Optimizing PRIMATEs S-box (2)

- Treat linear operations before and after nonlinear operations as two separate SLP instances

Optimizing PRIMATEs S-box (2)

- Treat linear operations before and after nonlinear operations as two separate SLP instances
- Try exact method

Optimizing PRIMATEs S-box (2)

- Treat linear operations before and after nonlinear operations as two separate SLP instances
- Try exact method
- If infeasible, try heuristic method

Optimizing PRIMATEs S-box (2)

- Treat linear operations before and after nonlinear operations as two separate SLP instances
- Try exact method
- If infeasible, try heuristic method

Optimizing PRIMATEs S-box (2)

- Treat linear operations before and after nonlinear operations as two separate SLP instances
- Try exact method
- If infeasible, try heuristic method

Managed to reduce 58 `XOR` gates to 31 `XOR` gates

Optimizing PRIMATEs S-box (3)

$$
\begin{aligned}
z_0 &= x_0 \oplus x_4 \\
z_1 &= x_1 \oplus x_2 \\
z_2 &= x_2 \oplus x_3 \\
q_0 &= x_0 \oplus x_3 \\
t_0 &= q_0 \vee x_1 \\
q_2 &= x_1 \oplus x_3 \\
q_3 &= \neg(x_0 \oplus x_2) \\
t_1 &= q_2 \vee q_3 \\
q_4 &= x_1 \oplus z_0 \\
q_5 &= x_0 \oplus z_2 \\
t_2 &= q_4 \wedge q_5 \\
q_6 &= \neg(x_4 \oplus q_5)
\end{aligned}
$$

$$
\begin{aligned}
q_7 &= x_4 \oplus z_1 \\
t_3 &= q_6 \vee q_7 \\
q_8 &= q_4 \oplus z_2 \\
z_9 &= t_0 \oplus t_3 \\
q_9 &= x_2 \oplus z_9 \\
t_4 &= q_8 \wedge q_9 \\
q_{10} &= \neg(x_3 \oplus z_0) \\
t_5 &= q_{10} \wedge z_0 \\
q_{12} &= \neg(z_1 \oplus z_9 \oplus t_2 \oplus t_4) \\
t_6 &= q_{12} \wedge z_2 \\
z_3 &= t_5 \oplus t_6 \\
z_4 &= t_3 \oplus z_3
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= t_2 \oplus z_4 \\
z_6 &= t_1 \oplus t_6 \\
z_7 &= t_4 \oplus z_5 \\
z_8 &= t_1 \oplus z_7 \\
z_{10} &= t_0 \oplus z_7 \\
z_{11} &= t_4 \oplus z_4 \\
z_{12} &= z_6 \oplus z_{11} \\
y_0 &= \neg(q_2 \oplus z_5) \\
y_1 &= z_0 \oplus z_8 \\
y_2 &= q_7 \oplus z_{12} \\
y_3 &= q_6 \oplus z_{11} \\
y_4 &= x_2 \oplus z_{10}
\end{aligned}
$$

Wrapping up. . .

- The paper includes all optimized implementations in the appendix

Wrapping up...

- The paper includes all optimized implementations in the appendix
- Tools to automate this (generate equations, convert to CNF, solve, retrieve result and corresponding implementation) are available online and in the public domain
  `https://ko.stoffelen.nl/`

Wrapping up. . .

- The paper includes all optimized implementations in the appendix
- Tools to automate this (generate equations, convert to CNF, solve, retrieve result and corresponding implementation) are available online and in the public domain

  `https://ko.stoffelen.nl/`

- For small functions, our method works quite nicely

Wrapping up. . .

- The paper includes all optimized implementations in the appendix
- Tools to automate this (generate equations, convert to CNF, solve, retrieve result and corresponding implementation) are available online and in the public domain

  `https://ko.stoffelen.nl/`

- For small functions, our method works quite nicely

- Not feasible for, say, 8-bit S-boxes because of exponential encoding E.g., RECTANGLE with $k = 5, w = 4$ already has 21372 variables and 106151 clauses in CNF

Wrapping up. . .

- The paper includes all optimized implementations in the appendix
- Tools to automate this (generate equations, convert to CNF, solve, retrieve result and corresponding implementation) are available online and in the public domain

  `https://ko.stoffelen.nl/`

- For small functions, our method works quite nicely

- Not feasible for, say, 8-bit S-boxes because of exponential encoding
  E.g., RECTANGLE with $k = 5, w = 4$ already has 21372 variables and 106151 clauses in CNF

- Thanks for your attention

References I

Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson.
Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers.
Cryptology ePrint Archive, Report 2007/024, 2007.
http://eprint.iacr.org/.

Joan Boyar and René Peralta.
A new combinational logic minimization technique with applications to cryptology.
In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer Berlin Heidelberg, 2010.

Nicolas Courtois, Theodosis Mourouzis, and Daniel Hulme.
Exact logic minimization and multiplicative complexity of concrete algebraic and cryptographic circuits.
International Journal On Advances in Intelligent Systems, 6(3 and 4):165–176, 2013.

Carsten Fuhs and Peter Schneider-Kamp.
Synthesizing shortest linear straight-line programs over GF(2) using SAT.
In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing – SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin Heidelberg, 2010.

References II

Theodosis Mourouzis.
Optimizations in Algebraic and Differential Cryptanalysis.
PhD thesis, UCL (University College London), 2015.