Finding Optimal Bitsliced Implementations of 4×4-bit S-boxes

SKEW 2011
February 17, 2011

Markus Ullrich, Christophe De Cannière, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha and Bart Preneel

Contents

Contents

Problem

1. How can we find THE most efficient implementations of s-boxes?
2. Can we find the optimal s-boxes covering all the s-boxes?

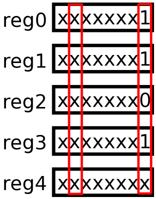
Problem

1. How can we find THE most efficient implementations of s-boxes?
2. Can we find the optimal s-boxes covering all the s-boxes?

- S-boxes limited to
  - 4×4-bit s-boxes
  - Invertible s-boxes

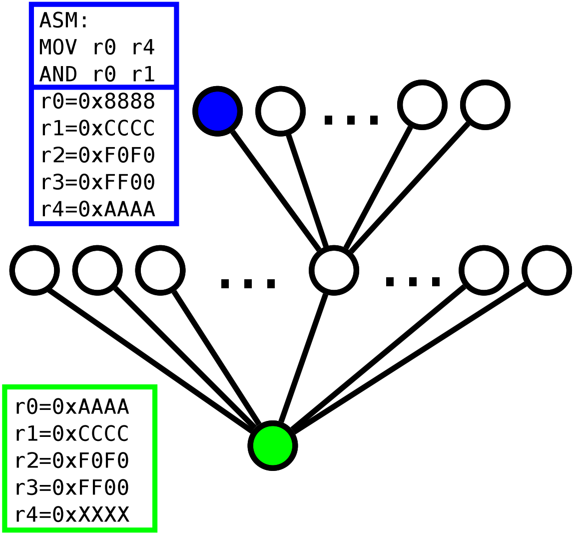
Architecture

- Software implementation using bitslicing
- 4+1 register
- Instruction set
    - AND
    - OR
    - XOR
    - NOT
    - MOV
- No parallelism



Contents

Search

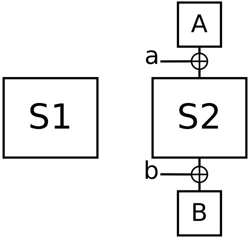

ASM:
MOV r0 r4
AND r0 r1
r0=0x8888
r1=0xCCCC
r2=0xF0F0
r3=0xFF00
r4=0xAAAA
r0=0xAAAA
r1=0xCCCC
r2=0xF0F0
r3=0xFF00
r4=0xXXXX

Search method

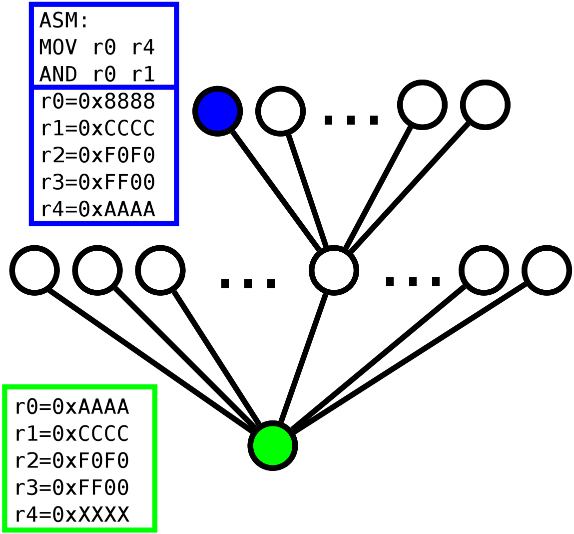
- Enumerating all s-boxes in order of cost function
  - No heuristics
- Limited to applications with monotonously increasing cost functions

Equivalence

- Affine equivalence:
    - Classification according to affine equivalence
    - Definition: $S_1(x) = B(S_2(Ax \oplus a) \oplus b)$
    - Properties regarding linear and differential cryptanalysis invariant



Search


ASM:
MOV r0 r4
AND r0 r1
r0=0x8888
r1=0xCCCC
r2=0xF0F0
r3=0xFF00
r4=0xAAAA
r0=0xAAAA
r1=0xCCCC
r2=0xF0F0
r3=0xFF00
r4=0xXXXX

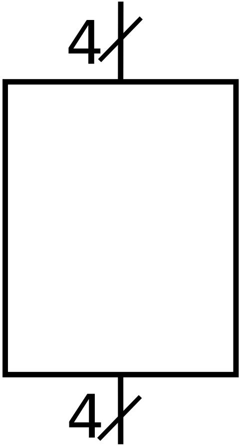
Contents

Reducing the branching factor

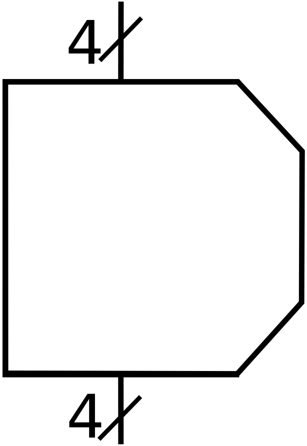
- Rule set from D. A. Osvik[1]
  - S-box invertible
  - No double negation
  - Reading before overwriting
  - Uninitialised values cannot be read
  - Double nodes are dismissed

[1] Dag Arne Osvik: Speeding up Serpent. AES Candidate Conference 2000

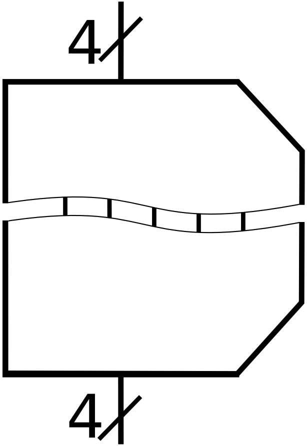
Advanced caching



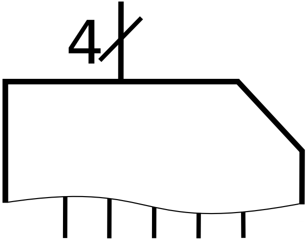
Advanced caching



Advanced caching

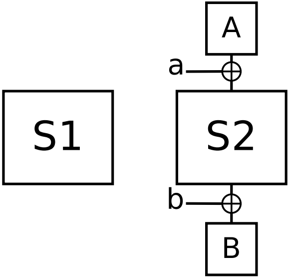


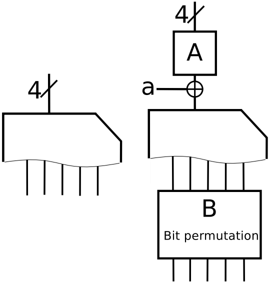
Advanced caching



Advanced caching

- Initial approach: dismissing nodes that are equal
- New approach: using affine equivalences





Contents

Overview

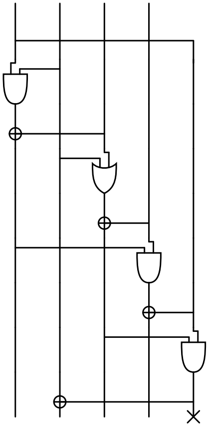
- Searched until cost of 12 instructions
- more than 2 month on 8 Xeon cores with 64GB RAM
- 272 out of 302 classes found
- Cover 90% of all s-boxes
- For each of these classes:
  - Representative
  - Assembly code

Linear and differential properties

MLP $-1/2$	$1/8$	$1/4$	$3/8$	$1/2$
$\lvert c \rvert$	$1/4$	$1/2$	$3/4$	1
min. cost	-	9	9	0

MDP	$1/8$	$1/4$	$3/8$	$1/2$	$5/8$	$3/4$	$7/8$	1
min. cost	-	9	10	6	9	6	-	0

'Smallest s-box ever'



- 9 instructions
- MDP $= 1/4$
- MLP $= 1/2 + 1/4$

ASM code

```
0 MOV r4 r0
1 AND r0 r1
2 XOR r0 r2
3 OR r2 r1
4 XOR r2 r3
5 AND r3 r0
6 XOR r3 r4
7 AND r4 r2
8 XOR r1 r4
r0 r1 r2 r3
```

Compared with literature

Cipher	S-box	Class	cost rep.	cost s-box inst. (cycl.)
Serpent	S_4, S_5	9	11	19 (10)
	S_4^{-1},S_5^{-1}	10	12	19 (10)
	S_0^{-1},S_1	14	10	18 (10)
	S_0,S_1^{-1}	15	10	18 (9)
	S_2,S_2^{-1},S_6,S_6^{-1}	16	11	16 (8)
	S_3,S_3^{-1},S_7,S_7^{-1}	not found	-	18 (10)
Luffa	Q	16	11	16 (6)
Noekeon	$S = S^{-1}$	13	9	16

A new design approach

Old approach

1. Designing the parts other than s-box
    - specifications get refined more and more
2. Finding s-boxes that fulfil the requirements

New approach

1. Choosing an s-box class
2. Selecting the most efficient representative as s-box
3. Designing the other components of the cipher

Contents

Open problems and future research

- Verifying the new design approach
- Affine equivalence and the NOT instruction
- More advanced architectures (SSE, parallelisation)
- Using other classification criteria

Conclusion

- An approach to systematically search efficient implementations of s-boxes has been presented
- Most s-box classes have been found
  - Interesting tradeoffs
  - Compared with literature
- New design approach has been proposed

Questions

Questions?