RECTANGLE: A Lightweight Block Cipher Suitable for Multiple Platforms

Wentao Zhang, Zhenzhen Bao,
Dongdai Lin, Vincent Rijmen,
Bohan Yang, Ingrid Verbauwhede

2015-7-20

Outline

1. Motivation
2. The RECTANGLE Block Cipher
3. Design Rationale
4. Security Analysis
5. Hardware and Software Performance
6. Summary

1. Motivation

- There is a need for new lightweight designs that combine the following properties:
  - good security margin against mathematical attacks
  - easily protectable against side-channel attacks (according to current state of the art)
  - good performance in software AND hardware
  - public design criteria

2. The RECTANGLE Block Cipher

- RECTANGLE
  - Block length: 64 bits
  - Key length: 80 or 128 bits
  - Structure: SP-network with 25 rounds

2. RECTANGLE – Cipher State

$$\begin{bmatrix} w_{15} & \cdots & w_2 & w_1 & w_0 \\ w_{31} & \cdots & w_{18} & w_{17} & w_{16} \\ w_{47} & \cdots & w_{34} & w_{33} & w_{32} \\ w_{63} & \cdots & w_{50} & w_{49} & w_{48} \end{bmatrix}$$

A Cipher State

$$\begin{bmatrix} a_{0,15} & \cdots & a_{0,2} & a_{0,1} & a_{0,0} \\ a_{1,15} & \cdots & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,15} & \cdots & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,15} & \cdots & a_{3,2} & a_{3,1} & a_{3,0} \end{bmatrix}$$

Two-dimensional Way

2. RECTANGLE – Round Function

- The round transformation: 3 steps

  1). AddRoundkey: a simple XOR of the round subkey

  2). SubColumn

  3). ShiftRow

2. RECTANGLE – SubColumn

2). SubColumn: parallel application of S-boxes to the 4 bits in the same column

$$
\begin{pmatrix} a_{0,15} \\ a_{1,15} \\ a_{2,15} \\ a_{3,15} \end{pmatrix} \cdots \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix} \begin{pmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{pmatrix} \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{pmatrix}
$$

$$
\downarrow S \quad \cdots \quad \downarrow S \quad \downarrow S \quad \downarrow S
$$

$$
\begin{pmatrix} b_{0,15} \\ b_{1,15} \\ b_{2,15} \\ b_{3,15} \end{pmatrix} \cdots \begin{pmatrix} b_{0,2} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix} \begin{pmatrix} b_{0,1} \\ b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix} \begin{pmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{pmatrix}
$$

SubColumn Operates on the Columns of the State

2. RECTANGLE – SubColumn (Cont.)

S-box $S : F_2^4 \longrightarrow F_2^4$

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	6	5	C	A	1	E	7	9	B	0	3	D	8	F	4	2

2. RECTANGLE – ShiftRow

3). ShiftRow: a left rotation to each row over different offsets

$$\begin{pmatrix} a_{0,15} & \cdots & a_{0,2} & a_{0,1} & a_{0,0} \end{pmatrix} \xrightarrow{\lll 0} \begin{pmatrix} a_{0,15} & \cdots & a_{0,2} & a_{0,1} & a_{0,0} \end{pmatrix}$$

$$\begin{pmatrix} a_{1,15} & \cdots & a_{1,2} & a_{1,1} & a_{1,0} \end{pmatrix} \xrightarrow{\lll 1} \begin{pmatrix} a_{1,14} & \cdots & a_{1,1} & a_{1,0} & a_{1,15} \end{pmatrix}$$

$$\begin{pmatrix} a_{2,15} & \cdots & a_{2,2} & a_{2,1} & a_{2,0} \end{pmatrix} \xrightarrow{\lll 12} \begin{pmatrix} a_{2,3} & \cdots & a_{2,6} & a_{2,5} & a_{2,4} \end{pmatrix}$$

$$\begin{pmatrix} a_{3,15} & \cdots & a_{3,2} & a_{3,1} & a_{3,0} \end{pmatrix} \xrightarrow{\lll 13} \begin{pmatrix} a_{3,2} & \cdots & a_{3,5} & a_{3,4} & a_{3,3} \end{pmatrix}$$

ShiftRow Operates on the Rows of the State

2. RECTANGLE - Key Schedule

- Two versions: 80-bit and 128-bit key
  - Take 80-bit version for example

2. RECTANGLE - Key Schedule (Cont.)

- At round i (i = 0, 1,…, 24), the 64-bit subkey K_i consists of the first 4 rows of the current contents of the key register.

$$\begin{bmatrix} v_{15} & \cdots & v_2 & v_1 & v_0 \\ v_{31} & \cdots & v_{18} & v_{17} & v_{16} \\ v_{47} & \cdots & v_{34} & v_{33} & v_{32} \\ v_{63} & \cdots & v_{50} & v_{49} & v_{48} \\ v_{79} & \cdots & v_{66} & v_{65} & v_{64} \end{bmatrix} \qquad \begin{bmatrix} \kappa_{0,15} & \cdots & \kappa_{0,2} & \kappa_{0,1} & \kappa_{0,0} \\ \kappa_{1,15} & \cdots & \kappa_{1,2} & \kappa_{1,1} & \kappa_{1,0} \\ \kappa_{2,15} & \cdots & \kappa_{2,2} & \kappa_{2,1} & \kappa_{2,0} \\ \kappa_{3,15} & \cdots & \kappa_{3,2} & \kappa_{3,1} & \kappa_{3,0} \\ \kappa_{4,15} & \cdots & \kappa_{4,2} & \kappa_{4,1} & \kappa_{4,0} \end{bmatrix}$$

A 80-bit Key State and its Two-dimensional Representation

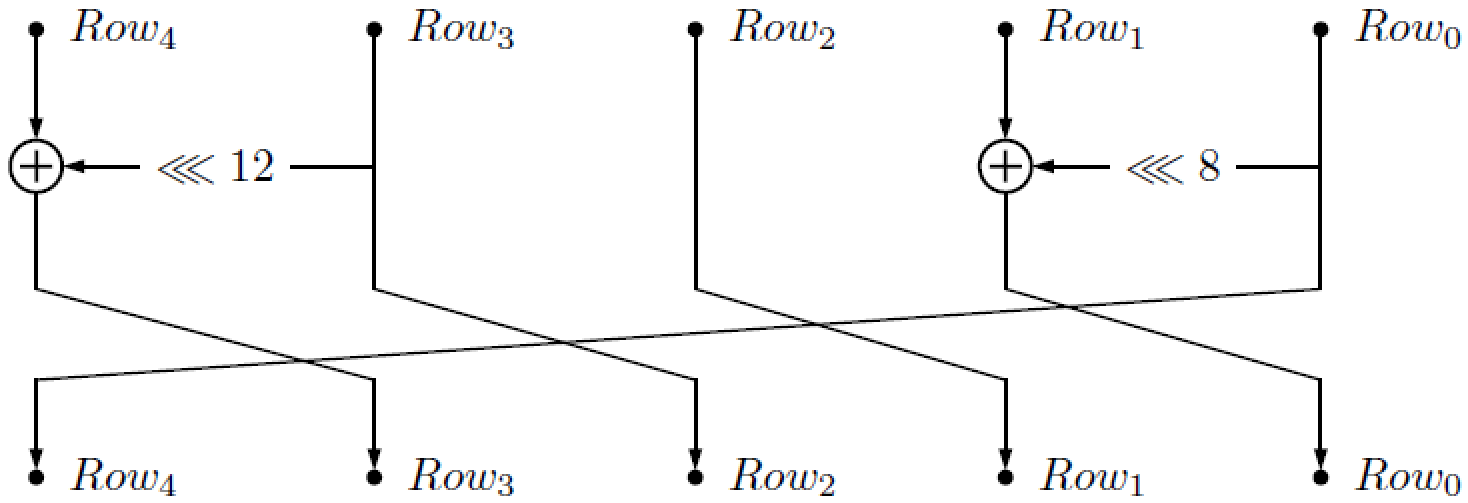
2. RECTANGLE - Key Schedule (Cont.)

- After extracting K_i, the key register is updated as follows:
  1). Applying 4 S-box operations to the upper right part of the key register

$$\begin{bmatrix} \kappa_{0,15} & \cdots & \kappa_{0,3} & \kappa_{0,2} & \kappa_{0,1} & \kappa_{0,0} \\ \kappa_{1,15} & \cdots & \kappa_{1,3} & \kappa_{1,2} & \kappa_{1,1} & \kappa_{1,0} \\ \kappa_{2,15} & \cdots & \kappa_{2,3} & \kappa_{2,2} & \kappa_{2,1} & \kappa_{2,0} \\ \kappa_{3,15} & \cdots & \kappa_{3,3} & \kappa_{3,2} & \kappa_{3,1} & \kappa_{3,0} \\ \kappa_{4,15} & \cdots & \kappa_{4,3} & \kappa_{4,2} & \kappa_{4,1} & \kappa_{4,0} \end{bmatrix}$$

2. RECTANGLE - Key Schedule (Cont.)

2). Applying a 1-round generalized Feistel transformation to the key register:



2. RECTANGLE - Key Schedule (Cont.)

3). A 5-bit round constant RC[i] is XORed with the 5-bit key state $(\kappa_{0,4}||\kappa_{0,3}||\kappa_{0,2}||\kappa_{0,1}||\kappa_{0,0})$

Note: the round constants are generated using a LFSR

2. RECTANGLE - Key Schedule (Cont.)

- Finally, K_{25} is extracted from the updated key state

3. Design Rationale

3.1 Bit-slice technique

3.2 Shift-Row transformation

3.3 Choice Criteria of S-box

3.4 Key Schedule

3.1 Bit-slice technique

- Bit-slice technique:
  - In a bit-slice implementation, one software logical instruction corresponds to simultaneous execution of n hardware logical gates, n is the length of a subblock.

3.1 Bit-slice technique (Cont.)

- Examples:
  - Serpent, Noekeon, Keccak and JH are 4 cryptographic algorithms that can benefit from the bit-slice technique for their software performance.
  - It is worth noticing that the 4 ciphers not only perform well in hardware but also in software.

3.1 Bit-slice technique (Cont.)

- The main idea of the design of RECTANGLE is to make use of bit-slice technique in a lightweight manner.

3.2 Shift-Row transformation

- Consider a 64-bit SP-network block cipher
  - A 64-bit state is arranged as a 4×16 array
  - First applying the same S-box to each column independently
  - Then, the P-layer should make each column dependent on some other columns

3.2 Shift-Row transformation (Cont.)

- In such a situation, 16-bit rotations are probably the best choice:
  - achieve the goal of mixing up different columns
  - simple wirings in hardware implementation
  - easily implemented in software using bit-slice technique

3.2 Shift-Row transformation (Cont.)

- Parameter selection: full dependency after a minimal number of rounds
  - 16 candidates
  - For each candidate, after 4 rounds, each of the 64 input bits influence each of the 64 output bits.
  - From them, we choose one: (0, 1, 12, 13)

3.3 Choice Criteria of S-box

- For a better security against DC and LC, a new idea, restrict the two values of an S-box:
  - the number of differentials (a, b) with wt(a)=wt(b)=1
  - the number of linear approximations (a, b) with wt(a)=wt(b)=1

- For more details: *A New Classification of 4-bit Optimal S-boxes and its Application to PRESENT, RECTANGLE and SPONGENT*, FSE'2015

3.3 Choice Criteria of S-box (Cont.)

- efficient implementation
  - the RECTANGLE S-box: a sequence of 12 basic logical instructions:

1. $t_1 = \sim a_1;$ 2. $t_2 = a_0 \,\&\, t_1;$ 3. $t_3 = a_2 \oplus a_3;$
4. $b_0 = t_2 \oplus t_3;$ 5. $t_5 = a_3 | t_1;$ 6. $t_6 = a_0 \oplus t_5;$
7. $b_1 = a_2 \oplus t_6;$ 8. $t_8 = a_1 \oplus a_2;$ 9. $t_9 = t_3 \,\&\, t_6;$
10. $b_3 = t_8 \oplus t_9;$ 11. $t_{11} = b_0 | t_8;$ 12. $b_2 = t_6 \oplus t_{11}$

$a_3||a_2||a_1||a_0$ is the input, and $b_3||b_2||b_1||b_0$ is the output.
"$\sim$" is NOT, "&" is AND, "$|$" is OR, "$\oplus$" is XOR.

3.4 Key Schedule

- **Take 80-bit key for example:**
  - Usage of 4 S-boxes to provide appropriate confusion;
  - The 1-round generalized Feistel transformation is used to provide appropriate diffusion;
  - Usage of round constants to eliminate symmetries.

4. Security Analysis

4.1 Security against Mathematical Attacks

4.2 Protection against Side Channel Attacks

4.1 Security against Mathematical Attacks

- We evaluated in detail the security of RECTANGLE against differential, linear, impossible differential, integral and key schedule attacks.
  - we can mount a differential attack on 18-round RECTANGLE, which is the highest number of rounds that we can attack.
- We believe that 25-round RECTANGLE has an enough and comfortable security margin.

4.2 Protection against Side Channel Attacks

- Threshold implementation and masking of RECTANGLE are feasible and not harder than other lightweight block ciphers:
  - the RECTANGLE S-box: the only nontrivial part in threshold implementation and masking
  - According to the work of Begul Bilgin et al (CHES’2012), the RECTANGLE S-box belongs to class 266, only 3 shares and within each share only one pair of G and F are required for protection against 1st order attack.

5. Hardware and Software Performance

5.1 Hardware performance

5.2 Software performance on 64-bit CPU

5.3 Software performance on 8-bit AVR

5.1 Hardware Performance

- Verilog HDL, Mentor Graphics Modelsim SE PLUS 6.6d
- Synopsys Design Compiler D-2010.03-SP4, UMC’s 0.13μm.1P8M Low Leakage Library

A comparison (Area vs. Throughput)

	Key size	Block size	Cycles per Block	Tech. μm	Area (GE)	Tput.At 100KHz(Kbps)
			Block Ciphers			
AES-128[44]	128	128	226	0.13	2400	56.6
LED-64[32]	64	64	1248	0.18	966	5.1
PICCOLO-80[50]	80	64	27	0.13	1496	237
PRESENT-80[48]	80	64	32	0.18	1570	200
RECTANGLE-80	80	64	26	0.13	1599.5	246
RECTANGLE-128	128	64	26	0.13	2063.5	246
			Stream Ciphers			
Grain[31]	80	1	1	0.13	1294	100
Trivium[31]	80	1	1	0.13	2599	100

5.2 Software Performance on 64-bit CPU

- 2.5GHz Intel(R) Core i5-2520M CPU, Intel C++ compiler

A Comparison with several other ciphers:

	LED	Piccolo	PRESENT	RECTANGLE
block length	64	64	64	64
key length	64	80	80	80
one block enc.	65	67.1 [4]	62	30.5
SSE enc. (cycles/byte)	-	4.57 [39] 16 para. blocks	4.73 [39] 32 para. blocks	3.9 8 para. blocks

"one block enc." is for a single block encryption.
"SSE enc." is for multiple parallel encryptions using 128-bit SSE instructions.
"para." means parallel. "-" means the value is unavailable at the time of writing.

5.3 Software Performance on 8-bit AVR

- Atmel ATtiny45, AVR studio 6.0
- Static implementation: the seed key is expanded and the round keys are loaded into SRAM

Key size [bits]	Code size [bytes] (enc. + k.s.)	RAM [bytes]	Enc. [Cycles]	Key Schedule [Cycles]
80	636	226	1920	1878
128	614	232	1920	1462

3 different implementations:

Method	Key size [bits]	Code size [bytes]		RAM [bytes]	Cycles		
		enc.+k.s.	dec.+k.s.	enc./dec.	enc.	dec.	k.s.
Static	80	636	638	226	1920	1945	1878
	128	614	616	232	1920	1945	1462

Method	Key size [bits]	Code size [bytes]		RAM [bytes]	Cycles		
		enc.+e.k.	dec.+e.k.	enc./dec.	enc.	dec.	k.s.
Fixed	80/128	574	576	8	2129	2154	-

Method	Key size [bits]	Code size [bytes]		RAM [bytes]	Cycles		Unrolled
		enc.+k.s.	dec.+i.k.s.	enc./dec.	enc.+ k.s.	dec.+i.k.s.	
On-the-fly	80	500	504	18	2801	2851	1-round
	128	488	492	24	2438	2488	
	80	1284	1304	18	2617	2667	5-round
	128	1092	1112	24	2112	2162	4-round

“enc.”, “dec.”,“k.s. ”, “i.k.s. ” and “e.k. ”means encryption, decryption, key schedule, inverse key schedule and expanded key respectively.

6. Summary

- RECTANGLE satisfies all the requirements that we listed in the motivation:

6. Summary

- RECTANGLE satisfies all the requirements that we listed in the motivation:
  - good security margin against mathematical attacks
  - easily protectable against side-channel attacks
  - good performance in software AND hardware
  - public design criteria

6. Summary

- Furthermore:
  - Largely due to our careful selection of the S-box, RECTANGLE achieves a very good security-performance tradeoff.
  - The selection of the P-layer is also important:
    - 3 rotations: extremely low-cost in hardware, but also very efficient in software.
  - The combination of the S-box and the P-layer brings the cipher a very limited clustering of differential/linear trails.

6. Summary

- In the end, we strongly encourage further security analysis of RECTANGLE.

Thanks for your attention !

Questions?