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

\[
\begin{pmatrix}
a_{0,15} \\
a_{1,15} \\
a_{2,15} \\
a_{3,15}
\end{pmatrix}
\ldots
\begin{pmatrix}
a_{0,2} \\
a_{1,2} \\
a_{2,2} \\
a_{3,2}
\end{pmatrix}
\begin{pmatrix}
a_{0,1} \\
a_{1,1} \\
a_{2,1} \\
a_{3,1}
\end{pmatrix}
\begin{pmatrix}
a_{0,0} \\
a_{1,0} \\
a_{2,0} \\
a_{3,0}
\end{pmatrix}
\]

\[
\begin{pmatrix}
b_{0,15} \\
b_{1,15} \\
b_{2,15} \\
b_{3,15}
\end{pmatrix}
\ldots
\begin{pmatrix}
b_{0,2} \\
b_{1,2} \\
b_{2,2} \\
b_{3,2}
\end{pmatrix}
\begin{pmatrix}
b_{0,1} \\
b_{1,1} \\
b_{2,1} \\
b_{3,1}
\end{pmatrix}
\begin{pmatrix}
b_{0,0} \\
b_{1,0} \\
b_{2,0} \\
b_{3,0}
\end{pmatrix}
\]

SubColumn Operates on the Columns of the State
2. RECTANGLE – SubColumn (Cont.)

S-box $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(x)$</td>
<td>6</td>
<td>5</td>
<td>C</td>
<td>A</td>
<td>1</td>
<td>E</td>
<td>7</td>
<td>9</td>
<td>B</td>
<td>0</td>
<td>3</td>
<td>D</td>
<td>8</td>
<td>F</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
2. RECTANGLE – ShiftRow

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

\[
\begin{align*}
(a_{0,15} \cdots a_{0,2} \ a_{0,1} \ a_{0,0}) & \xrightarrow{0} (a_{0,15} \cdots a_{0,2} \ a_{0,1} \ a_{0,0}) \\
(a_{1,15} \cdots a_{1,2} \ a_{1,1} \ a_{1,0}) & \xrightarrow{1} (a_{1,14} \cdots a_{1,1} \ a_{1,0} \ a_{1,15}) \\
(a_{2,15} \cdots a_{2,2} \ a_{2,1} \ a_{2,0}) & \xrightarrow{12} (a_{2,3} \cdots a_{2,6} \ a_{2,5} \ a_{2,4}) \\
(a_{3,15} \cdots a_{3,2} \ a_{3,1} \ a_{3,0}) & \xrightarrow{13} (a_{3,2} \cdots a_{3,5} \ a_{3,4} \ a_{3,3})
\end{align*}
\]

**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
At round $i$ ($i = 0, 1, \ldots, 24$), the 64-bit subkey $K_i$ consists of the first 4 rows of the current contents of the key register.
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
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
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 $4 \times 16$ 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 \(\text{wt}(a)=\text{wt}(b)=1\)
- the number of linear approximations \((a, b)\) with \(\text{wt}(a)=\text{wt}(b)=1\)

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 0.13\(\mu\)m 1P8M Low Leakage Library
## A comparison (Area vs. Throughput)

<table>
<thead>
<tr>
<th>Block Ciphers</th>
<th>Key size</th>
<th>Block size</th>
<th>Cycles per Block</th>
<th>Tech. $\mu$m</th>
<th>Area (GE)</th>
<th>Tput. At 100KHz (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128[44]</td>
<td>128</td>
<td>128</td>
<td>226</td>
<td>0.13</td>
<td>2400</td>
<td>56.6</td>
</tr>
<tr>
<td>LED-64[32]</td>
<td>64</td>
<td>64</td>
<td>1248</td>
<td>0.18</td>
<td>966</td>
<td>5.1</td>
</tr>
<tr>
<td>PICCOLO-80[50]</td>
<td>80</td>
<td>64</td>
<td>27</td>
<td>0.13</td>
<td>1496</td>
<td>237</td>
</tr>
<tr>
<td>PRESENT-80[48]</td>
<td>80</td>
<td>64</td>
<td>32</td>
<td>0.18</td>
<td>1570</td>
<td>200</td>
</tr>
<tr>
<td>RECTANGLE-80</td>
<td>80</td>
<td>64</td>
<td>26</td>
<td>0.13</td>
<td>1599.5</td>
<td>246</td>
</tr>
<tr>
<td>RECTANGLE-128</td>
<td>128</td>
<td>64</td>
<td>26</td>
<td>0.13</td>
<td>2063.5</td>
<td>246</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stream Ciphers</th>
<th>Key size</th>
<th>Block size</th>
<th>Cycles per Block</th>
<th>Tech. $\mu$m</th>
<th>Area (GE)</th>
<th>Tput. At 100KHz (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grain[31]</td>
<td>80</td>
<td>1</td>
<td>1</td>
<td>0.13</td>
<td>1294</td>
<td>100</td>
</tr>
<tr>
<td>Trivium[31]</td>
<td>80</td>
<td>1</td>
<td>1</td>
<td>0.13</td>
<td>2599</td>
<td>100</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th></th>
<th>LED</th>
<th>Piccolo</th>
<th>PRESENT</th>
<th>RECTANGLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>block length</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>key length</td>
<td>64</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>one block enc.</td>
<td>65</td>
<td>67.1 [4]</td>
<td>62</td>
<td>30.5</td>
</tr>
<tr>
<td>(cycles/byte)</td>
<td>-</td>
<td>16 para. blocks</td>
<td>32 para. blocks</td>
<td>8 para. blocks</td>
</tr>
</tbody>
</table>

“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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>636</td>
<td>226</td>
<td>1920</td>
<td>1878</td>
</tr>
<tr>
<td>128</td>
<td>614</td>
<td>232</td>
<td>1920</td>
<td>1462</td>
</tr>
</tbody>
</table>
3 different implementations:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>enc.+k.s.</td>
<td>dec.+k.s.</td>
<td>enc./dec.</td>
</tr>
<tr>
<td>Static</td>
<td>80</td>
<td>636</td>
<td>638</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>614</td>
<td>616</td>
<td>232</td>
</tr>
<tr>
<td>Fixed</td>
<td>80/128</td>
<td>574</td>
<td>576</td>
<td>8</td>
</tr>
<tr>
<td>On-the-fly</td>
<td>80</td>
<td>500</td>
<td>504</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>488</td>
<td>492</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1284</td>
<td>1304</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>1092</td>
<td>1112</td>
<td>24</td>
</tr>
</tbody>
</table>

“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?