RECTANGLE: A Bit-slice Ultra-Lightweight Block Cipher Suitable for Multiple Platforms

Wentao Zhang[1,2], Zhenzhen Bao[1], Dongdai Lin[1], Vincent Rijmen[2], Bohan Yang[2], Ingrid Verbauwhede[2]

1.State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, P. R. China
2.KU Leuven, ESAT/COSIC and iMinds; Kasteelpark Arenberg 10, Leuven, Belgium
{*zhangwentao, baozhenzhen, ddlin*}*@iie.ac.cn*
{*vincent.rijmen, bohan.yang, ingrid.verbauwhede*}*@esat.kuleuven.be*

Abstract. In this paper, we propose a new lightweight block cipher named RECTANGLE. The main idea of the design of RECTANGLE is to allow lightweight and fast implementations using bit-slice techniques. RECTANGLE uses an SP-network. The substitution layer consists of 16 4×4 S-boxes in parallel. The permutation layer is composed of 3 rotations. As shown in this paper, RECTANGLE offers great performance in both hardware and software environment, which proves enough flexibility for different application scenario. The following are 3 main advantages of RECTANGLE. First, RECTANGLE is extremely hardware-friendly. For the 80-bit key version, a one-cycle-per-round parallel implementation only needs 1467 gates for a throughput of 246 Kbits/sec at 100KHz clock and an energy efficiency of 1.11 pJ/bit. Second, RECTANGLE achieves a very competitive software speed among the existing lightweight block ciphers due to its bit-slice style. Using 128-bit SSE instructions, a bit-slice implementation of RECTANGLE reaches an average encryption speed of about 5.38 cycles/byte for messages around 1000 bytes. Last but not least. We propose new design criteria for 4×4 S-boxes. RECTANGLE uses such a new type of S-box. Due to our careful selection of the S-box and the asymmetric design of the permutation layer, RECTANGLE achieves a very good security-performance tradeoff. Our extensive and deep security analysis finds distinguishers for up to 14 rounds only, and the highest number of rounds that we can attack, is 18 (out of 25).

Key words: block cipher, design, lightweight cryptography, bit-slice, software efficiency, RFID

1 Introduction

Small embedded devices (including RFIDs, sensor nodes, smart cards) are now widely used in many applications. They are usually characterized by strong cost constraints, such as area, power, energy consumption for hardware aspect, and low memory, small code size for software aspect. Meanwhile, they also require cryptographic protection. As a result, many new lightweight ciphers have been proposed to provide strong security at a lower cost than standard solutions. Since symmetric-key ciphers, especially block ciphers, play an important role in the security of small embedded devices, the design of lightweight block ciphers has been a very active research topic over the last few years.

In the literature, quite a few lightweight block ciphers with various design strategies have been proposed, such as DESL/DESX/DESXL [33], Hummingbird [21], KATAN/ KTANTAN [19], KLEIN [25], LBlock [48], LED[27], PRESENT [11], TWINE [45] and so on. PRESENT was proposed at CHES'2007, and has attracted a lot of attention from cryptographic researchers due to its simplicity, impressive hardware performance and strong security. The design of PRESENT is extremely hardware-efficient, since it uses a bit permutation as its diffusion layer, which is a simple wiring in hardware implementation. In 2012, PRESENT was adopted as ISO/IEC lightweight cryptography standard. Many lightweight ciphers, such as PRESENT, KATAN/KTANTAN and Hummingbird, succeed in achieving a low area in hardware but the software performance is not good. However, high software performance is also needed from the same algorithm for many classical lightweight applications, as pointed out in [2, 12, 25, 27, 37]. The reason is that many constrained devices need to communicate with a server, hence it is very crucial that the lightweight cipher also has good software performance. LED is proposed at CHES'2011, the designers claim that LED is not only very compact in hardware but also maintains a reasonable performance profile for software implementation.

Among the new proposals, some present weaknesses, including ARMODILLO-2, Hummingbird-1 and KTANTAN [13, 39, 44]. Furthermore, as pointed out in [27], designers of "second generation" lightweight ciphers can learn from the progress and the omissions of the" first generation" proposals. The S-box of PRESENT is mainly selected according to its hardware area instead of security of the underlying cipher. Hence, the S-box of PRESENT is "weak" with respect to cipher security. As pointed out in [31], the PRESENT S-box is among the 8 percent worst S-boxes with respect to clustering of one bit linear trails. Along with the strong symmetry of the PRESENT permutation layer, there are very serious clustering problems both for linear trails and differential trails [10, 14, 31, 40, 46]. We give more details in section 3. As a result, for PRESENT, the best distinguisher so far can reach 24 rounds [14], which can be used to mount a shortcut attack on 26-round PRESENT (out of 31).

The bit-slice technique was introduced for speeding up the software speed of DES [4], and was used in the design of the Serpent block cipher [1]. In a bit-slice implementation, one software logical instruction corresponds to simultaneous execution of n hardware logical gates, where n is the length of a subblock. Take Serpent for example. Serpent is a 128-bit SP-network block cipher. The substitution layer is composed of 32 4×4 S-boxes, thus the subblock length is $n = 128/4 = 32$ for a bit-slice implementation. Noekeon [16], Keccak(SHA-3) [3] and JH [47] are 3 other primitives that can benefit from the bit-slice technique for their software performance. It is worth noticing that Serpent, Noekeon, Keccak and JH not only perform well in hardware but also in software. Furthermore, a bit-slice implementation is safe against implementation attacks such as cache and timing attacks compared with a table-based implementation [38]. However, the main design goal of all the mentioned bit-sliced ciphers is not "lightweight", there is plenty of room for improvement when it comes to a dedicated lightweight block cipher with bit-slice style.

1.1 Contributions

In this paper, we present a new lightweight block cipher RECTANGLE. The design of RECTANGLE makes use of the bit-slice technique in a lightweight manner, hence to achieve not only a very low area in hardware but also a very competitive performance in software. As a result, RECTANGLE adopts the SP-network structure. The substitution layer (S-layer) consists of 16 4×4 S-boxes in parallel. The permutation layer (P-layer) is composed of 3 rotations. The following are 3 main advantages of RECTANGLE:

1. RECTANGLE is extremely hardware-friendly, it has a competitive hardware cost as PRESENT in lightweight applications, while having a higher throughput/area ratio and lower energy per bit than PRESENT. The bit-sliced design principle of RECTANGLE allows for very efficient and flexible hardware implementations. For the 80-bit key version, using UMC $0.13\mu m$ standard cell library at 100 KHz , our round-based implementation could obtain a throughput of 246 Kbits/sec and an energy efficiency of 1.11 pJ/bit with only 1467 gates, our serialized implementation could obtain a throughput of 13.9 Kbits/sec and an energy efficiency of 31.7 pJ/bit with only 1066 gates. Also, the round-based implementation can be easily extended to parallel implementation for different application scenarios. More details are given in section 5.1.
2. Due to its bit-slice style, RECTANGLE achieves a very competitive software speed among the existing lightweight block ciphers. With a parallel mode of operation, a bit-slice implementation of RECTANGLE reaches an average encryption speed of about 5.38 cycles/byte for messages around 1000 bytes, using Intel 128-bit SSE instructions. For comparison, the bit-slice implementation of LED-64 and PRESENT reach an speed of 11.9 cycles/byte and 17.3 cycles/byte respectively [2]. More details are given in section V-B.
3. Last but not least. We propose new design criteria for 4×4 S-boxes. RECTANGLE uses such a new type of S-box. Due to our careful selection of the RECTANGLE S-box, together with the asymmetric design of the permutation layer, RECTANGLE achieves a very good security-performance trade off. After our extensive and deep security analysis, the best distinguisher of RECTANGLE can only reach 14 rounds. Using one 14-round differential distinguisher, we can mount a shortcut attack on 18-round RECTANGLE (out of 25), which is the highest number of rounds that we can attack.

This paper is organized as follows. Section 2 presents a specification of RECTANGLE; Section 3 discusses the security of RECTANGLE against known attacks; Section 4 motivates the design choices of RECTANGLE; Section 5 presents the hardware and software implementation results of the cipher; Section 6 presents the relation of RECTANGLE to several early designs. Section 7 concludes the paper.

2 The RECTANGLE Block Cipher

RECTANGLE is an iterated block cipher. The block length is 64 bits, the key length can be 80 or 128 bits.

2.1 The Cipher State, the Subkey State

A 64-bit plaintext, or a 64-bit intermediate result, or a 64-bit ciphertext is collectively called as a cipher state. A cipher state can be pictured as a 4×16 rectangular array of bits, which is the origin of the cipher name **RECTANGLE**. Let $W = w_{63}||\cdots||w_1||w_0$ denote a 64-bit cipher state, the first 16 bits $w_{15}||\cdots||w_1||w_0$ are arranged in row 0, the next 16 bits $w_{31}||\cdots||w_{17}||w_{16}$ are arranged in row 1, and so on, as illustrated in Figure 1. Similarly, a 64-bit subkey is also pictured as a 4×16 rectangular array. In the following, for convenience of description, a cipher state is described in a two-dimensional way, as illustrated in Figure 2.

$$\begin{bmatrix} w_{15} & w_{14} & \cdots & w_2 & w_1 & w_0 \\ w_{31} & w_{30} & \cdots & w_{18} & w_{17} & w_{16} \\ w_{47} & w_{46} & \cdots & w_{34} & w_{33} & w_{32} \\ w_{63} & w_{62} & \cdots & w_{50} & w_{49} & w_{48} \end{bmatrix}$$

Figure 1. A Cipher State

$$\begin{bmatrix} a_{0,15} & a_{0,14} & \cdots & a_{0,2} & a_{0,1} & a_{0,0} \\ a_{1,15} & a_{1,14} & \cdots & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,15} & a_{2,14} & \cdots & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,15} & a_{3,14} & \cdots & a_{3,2} & a_{3,1} & a_{3,0} \end{bmatrix}$$

Figure 2. Two-dimensional Representation

2.2 The Round Transformation

RECTANGLE is a 25-round SP-network cipher. Each of the 25 rounds consists of the following 3 steps: AddRoundkey, SubColumn, ShiftRow. After the last round, there is a final AddRoundKey.

AddRoundkey: A simple bitwise XOR of the round subkey to the intermediate state.

SubColumn: Parallel application of S-boxes to the 4 bits in the same column. The operation of SubColumn is illustrated in Figure 3. The input of a S-box is $Col(j) = a_{3,j}||a_{2,j}||a_{1,j}||a_{0,j}$ for $0 \leq j \leq 15$, and the output is $S(Col(j)) = b_{3,j}||b_{2,j}||b_{1,j}||b_{0,j}$.

$$\begin{pmatrix} a_{0,15} \\ a_{1,15} \\ a_{2,15} \\ a_{3,15} \end{pmatrix} \begin{pmatrix} a_{0,14} \\ a_{1,14} \\ a_{2,14} \\ a_{3,14} \end{pmatrix} \cdots \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix} \begin{pmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{pmatrix} \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{pmatrix}$$

$$\downarrow S \qquad \downarrow S \qquad \cdots \qquad \downarrow S \qquad \downarrow S \qquad \downarrow S$$

$$\begin{pmatrix} b_{0,15} \\ b_{1,15} \\ b_{2,15} \\ b_{3,15} \end{pmatrix} \begin{pmatrix} b_{0,14} \\ b_{1,14} \\ b_{2,14} \\ b_{3,14} \end{pmatrix} \cdots \begin{pmatrix} b_{0,2} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix} \begin{pmatrix} b_{0,1} \\ b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix} \begin{pmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{pmatrix}$$

Figure 3. SubColumn Operates on the Columns of the State

The S-box used in RECTANGLE is a 4-bit to 4-bit S-box $S : F_2^4 \rightarrow F_2^4$. The action of this S-box in hexadecimal notation is given by the following table.

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

ShiftRow: A left rotation to each row over different offsets. Row 0 is not rotated, row 1 is left rotated over 1 bit, row 2 is left rotated over 12 bits, row 3 is left rotated over 13 bits. Let $\lll (x)$ denote left rotation over x bits, the operation ShiftRow is illustrated in Figure 4.

$$\begin{aligned}
&\left(a_{0,15}\ a_{0,14}\ \cdots\ a_{0,2}\ a_{0,1}\ a_{0,0}\right) \xrightarrow{\lll(0)} \left(a_{0,15}\ a_{0,14}\ \cdots\ a_{0,2}\ a_{0,1}\ a_{0,0}\right)\\
&\left(a_{1,15}\ a_{1,14}\ \cdots\ a_{1,2}\ a_{1,1}\ a_{1,0}\right) \xrightarrow{\lll(1)} \left(a_{1,14}\ a_{1,13}\ \cdots\ a_{1,1}\ a_{1,0}\ a_{1,15}\right)\\
&\left(a_{2,15}\ a_{2,14}\ \cdots\ a_{2,2}\ a_{2,1}\ a_{2,0}\right) \xrightarrow{\lll(12)} \left(a_{2,3}\ a_{2,2}\ \cdots\ a_{2,6}\ a_{2,5}\ a_{2,4}\right)\\
&\left(a_{3,15}\ a_{3,14}\ \cdots\ a_{3,2}\ a_{3,1}\ a_{3,0}\right) \xrightarrow{\lll(13)} \left(a_{3,2}\ a_{3,1}\ \cdots\ a_{3,5}\ a_{3,4}\ a_{3,3}\right)
\end{aligned}$$

Figure 4. ShiftRow Operates on the Rows of the State

2.3 Key Schedule

RECTANGLE can accept keys of either 80 or 128 bits.

80-bit key For a 80-bit user-supplied key $V = v_{79}||\cdots||v_1||v_0$, the key is firstly stored in a 80-bit key register, see Figure 5.

$$\begin{bmatrix} v_{19}\ v_{18}\ \cdots\ v_2\ v_1\ v_0 \\ v_{39}\ v_{38}\ \cdots\ v_{22}\ v_{21}\ v_{20} \\ v_{59}\ v_{58}\ \cdots\ v_{42}\ v_{41}\ v_{40} \\ v_{79}\ v_{78}\ \cdots\ v_{62}\ v_{61}\ v_{60} \end{bmatrix} \qquad \begin{bmatrix} \kappa_{0,19} & \kappa_{0,18} & \cdots & \kappa_{0,2} & \kappa_{0,1} & \kappa_{0,0} \\ \kappa_{1,19} & \kappa_{1,18} & \cdots & \kappa_{1,2} & \kappa_{1,1} & \kappa_{1,0} \\ \kappa_{2,19} & \kappa_{2,18} & \cdots & \kappa_{2,2} & \kappa_{2,1} & \kappa_{2,0} \\ \kappa_{3,19} & \kappa_{3,18} & \cdots & \kappa_{3,2} & \kappa_{3,1} & \kappa_{3,0} \end{bmatrix}$$

Figure 5. A 80-bit Key State and its Two-dimensional Representation

At round i ($i = 0, 1, \cdots, 24$), the 64-bit round subkey K_i consists of the 16 rightmost columns of the current contents of the key register, i.e., $K_i = (\kappa_{3,15}||\cdots||\kappa_{3,1}||\kappa_{3,0})||(\kappa_{2,15}||\cdots||\kappa_{2,1}||\kappa_{2,0})||(\kappa_{1,15}||\cdots||\kappa_{1,1}||\kappa_{1,0})||(\kappa_{0,15}||\cdots||\kappa_{0,1}||\kappa_{0,0})$. After extracting K_i, the key register is updated as follows:

1. Applying the S-box S to the 0-th column, i.e.,
$$\kappa_{3,0}||\kappa_{2,0}||\kappa_{1,0}||\kappa_{0,0} := S(\kappa_{3,0}||\kappa_{2,0}||\kappa_{1,0}||\kappa_{0,0})$$
2. A left rotation to each row over different offsets. Row 0 is left rotated over 7 bits, row 1 is left rotated over 9 bits, row 2 is left rotated over 11 bits, row 3 is left rotated over 13 bits.
3. A 5-bit round constant RC[i] is XORed with the 5-bit key state $(\kappa_{0,4}||\kappa_{0,3}||\kappa_{0,2}||\kappa_{0,1}||\kappa_{0,0})$, i.e.,
$$\kappa_{0,4}||\kappa_{0,3}||\kappa_{0,2}||\kappa_{0,1}||\kappa_{0,0} := (\kappa_{0,4}||\kappa_{0,3}||\kappa_{0,2}||\kappa_{0,1}||\kappa_{0,0}) \oplus \text{RC[i]}$$

Finally, K_{25} is extracted from the updated key state. The round constants RC[i] ($i = 0, 1, \cdots, 24$) are generated by a 5-bit LFSR. At each round, the 5 bits $(rc_4, rc_3, rc_2, rc_1, rc_0)$ are left shifted over 1 bit, with the new value to rc_0 being computed as $rc_4 \oplus rc_2$. The initial value is defined as RC[0] $:= 0x1$. We list all the round constants in Appendix A.

128-bit key Due to the limitation of the paper length, the 128-bit key schedule is presented in Appendix B.

2.4 The Cipher

The encryption algorithm of RECTANGLE consists of 25 rounds and a final subkey XOR. The following is a pseudo C code:

$GenerateRoundKeys()$
for i = 0 to 24 do
{ $AddRoundKey(STATE, K_i)$
$SubColumn(STATE)$
$ShiftRow(STATE)$
}
$AddRoundKey(STATE, K_{25})$

3 Security Analysis

In this section, we present the results of our security analysis of RECTANGLE.

3.1 Differential Cryptanalysis

Differential [7] and linear [35] cryptanalysis are among the most powerful techniques available for block ciphers.

To attack an n-bit block cipher using differential cryptanalysis (DC), there must be a predictable difference propagation over all but a few rounds with a probability significantly larger than 2^{1-n}. A difference propagation is composed of a set of differential trails, where its probability is the sum of the probabilities of all differential trails that have the specified input difference and output difference [17]. For RECTANGLE, to be resistant against DC, it is a necessary condition that there is no difference propagation with a probability higher than 2^{-63}.

M.Matsui has presented a search algorithm for the best differential/linear trail of DES in [36], which uses branch-and-bound methods and very effective for DES. Based on this algorithm, we have written a program to search for the best differential trails of RECTANGLE from 1 round to 15 rounds , the results are presented in Table 1. The probability of the best 15-round differential trail is 2^{-66}.

Because of the simplicity of the ShiftRow transformation, we also need to consider the security of RECTANGLE against multiple differential cryptanalysis [10] and

Table 1. Probabilities of the Best Differential Trails of RECTANGLE

♯ Rounds	Probability	♯ Rounds	Probability	♯ Rounds	Probability
1	2^{-2}	6	2^{-18}	11	2^{-46}
2	2^{-4}	7	2^{-25}	12	2^{-51}
3	2^{-7}	8	2^{-31}	13	2^{-56}
4	2^{-10}	9	2^{-36}	14	2^{-61}
5	2^{-14}	10	2^{-41}	15	2^{-66}

the structure attack [46]. From a differential point of view, since all the operations in RECTANGLE have rotational symmetry, every trail has up to 16 rotation equivalent variants. For 15-round RECTANGLE, based on the branch-and-bound algorithm, we have searched for all the differential trails with probability between 2^{-66} and 2^{-76} (up to a rotation equivalence) and examined all the difference propagations made up of the investigated trails, the following are the experimental results:

1. There are 32 best difference propagations with probability $81 \times 2^{-72} \approx 2^{-65.66}$ each. Each is composed of 4 differential trails. Among the 4 trails, one with probability 2^{-66}, two with probability 2^{-69} each, and one with probability 2^{-72}.
2. Among all the difference propagations, the maximum number of trails of a difference propagation is 108, i.e., a difference propagation is composed of at most 108 different differential trails. For such a difference propagation, the probability is $281 \times 2^{-76} \approx 2^{-67.87}$.

From result 1, the probability of the best difference propagation is lower than 2^{-63}. From the two results, it can be seen that the clustering of differential trails of RECTANGLE is very limited, which can not be used to construct an effective difference propagation with more than 14 rounds.

In the case of PRESENT, there exists a large number of trails which can result in difference propagations with much higher probability [10, 46]. For comparison, we give some statistical data concerning the clustering of differential trails of 16-round PRESENT from [46]. For 16-round PRESENT, the probability of the best differential trail is 2^{-70}. There exists a 16-round difference propagation satisfying the following properties:

1. It includes 31996 trails when restricting the probability of differential trails between 2^{-70} and 2^{-80}. The probability is $2^{-62.175}$ when only considering these 31996 trails;
2. It includes 83720 trails when restricting the probability of differential trails between 2^{-70} and 2^{-92}. The probability is $2^{-62.133}$ when considering all the 83720 trails.

Therefore, we believe that it is impossible to construct an effective 15-round (multiple) differential distinguisher for RECTANGLE. Full dependency is reached already after 4 rounds, hence we believe 25-round RECTANGLE is enough to resist against (multiple) differential cryptanalysis.

Using one 14-round differential distinguisher, we can mount an attack on 18-round RECTANGLE, which is the best short-cut attack on RECTANGLE we found. Appendix E presents the distinguisher.

3.2 Linear Cryptanalysis

Assume a linear trail hold with probability p , define the bias ε as $(p-\frac{1}{2})$, the correlation coefficient C as 2ε. To attack an n-bit block cipher using linear cryptanalysis (LC), there must be a predictable linear propagation over all but a few rounds with an amplitude significantly larger than $2^{-\frac{n}{2}}$. A linear propagation is composed of a set of linear trails, where its amplitude is the sum of the correlation coefficients of all linear trails that have the specified input and output selection patterns [17]. The correlation coefficients of

Table 2. Correlation Coefficients of the Best Linear Trails of RECTANGLE

♯ Rounds	Cor. Coeffi.	♯ Rounds	Cor. Coeffi.	♯ Rounds	Cor. Coeffi.
1	$\pm 2^{-1}$	6	$\pm 2^{-10}$	11	$\pm 2^{-25}$
2	$\pm 2^{-2}$	7	$\pm 2^{-13}$	12	$\pm 2^{-28}$
3	$\pm 2^{-4}$	8	$\pm 2^{-16}$	13	$\pm 2^{-31}$
4	$\pm 2^{-6}$	9	$\pm 2^{-19}$	14	$\pm 2^{-34}$
5	$\pm 2^{-8}$	10	$\pm 2^{-22}$	15	$\pm 2^{-37}$

the linear trails are signed and their sign depends on the value of the round keys. For RECTANGLE, to be resistant against LC, it is a necessary condition that there is no linear propagation with an amplitude higher than 2^{-32}.

Since the strong round key dependence of interference makes locating the input and output selection patterns for which high correlations occur practically infeasible [17], we have to use the following theorem for an estimation.

Theorem 1 ([17]). *The square of the correlation is called correlation potential. The average correlation potential between an input and an output selection pattern is the sum of the correlation potentials of all linear trails between the input and output selection patterns:*

$$E(C_t^2) = \sum_i (C_i)^2$$

where C_t is the overall correlation, and C_i the correlation coefficient of a linear trail.

We have modified the search program used in the differential case to search for the best linear trails of RECTANGLE from 1 round to 15 rounds, the results are presented in Table 2. All of the best linear trails avoid the weakness that there is only one active S-box in each round, which occurs in PRESENT.

Similarly, we also need to consider the security of RECTANGLE against multiple linear cryptanalysis [9] and multidimensional linear cryptanalysis [24]. For 15-round RECTANGLE, the amplitude of the correlation coefficient of the best linear trail is 2^{-37}. Also based on the branch-and-bound algorithm, we have searched for all the linear trails with an amplitude of the correlation coefficient between 2^{-37} and 2^{-40} (up to a rotation equivalence) for 15-round RECTANGLE and examined all the linear propagations made up of the investigated trails, the following are the experimental results:

1. There are 128 best linear propagations with an amplitude of the average correlation $(1846 \times 2^{-80})^{0.5} \approx 2^{-34.58}$ each, which is lower than 2^{-32}. Each is composed of 883 linear trails. Among the 883 trails, 2 with correlation coefficient $\pm 2^{-37}$ each, 26 with correlation coefficient $\pm 2^{-38}$ each, 149 with correlation coefficient $\pm 2^{-39}$ each, and 706 with correlation coefficient $\pm 2^{-40}$ each.
2. Among all the linear propagations, the maximum number of trails of a linear propagation is 883. Actually, the best linear propagations have the maximum number of trails.

For comparison with PRESENT, notice the following two facts :

1. There exists a 16-round linear propagation of PRESENT, which is composed of $435,600$ linear trails with an amplitude of the correlation 2^{-32} each [40]. Thus, the amplitude of the average correlation is $(435600 \times 2^{-64})^{0.5} \approx 2^{-22.63}$.
2. There exists a 23-round linear propagation of PRESENT, which is composed of $367,261,713$ linear trails with an amplitude of the correlation 2^{-46} each [40]. Thus, the amplitude of the average correlation is $(367261713 \times 2^{-92})^{0.5} \approx 2^{-31.77}$.

From the above results and a comparison with PRESENT, it can be seen that the clustering of linear trails of RECTANGLE is limited, which can not be used to construct an effective linear propagation with more than 14 rounds. Therefore, we believe that it is impossible to construct an effective 15-round (multiple, multidimensional) linear distinguisher for RECTANGLE. Full dependency is reached already after 4 rounds, hence we believe 25-round RECTANGLE is enough to resist against linear cryptanalysis and its extension attacks.

3.3 Statistical Saturation Attack

The statistical saturation attack [15] is specially designed for PRESENT. It uses the weak diffusion of the PRESENT permutation. More specifically, for 4 selected S-box positions, 8 out of 16 input bits are directed to the same 4 S-box positions after the permutation. Using this property, there exists a theoretical attack against 24-round PRESENT.

Due to the weak diffusion of the permutation layer of RECTANGLE, we must consider the security of RECTANGLE against statistical saturation attack. Consider the following 4 properties:

1. Consider the 3 columns with an index set $\{0,1,13\}$, then 6 out of 12 bits are still directed to the same 3 column positions after ShiftRow.
2. Consider the 4 columns with an index set $\{0,1,12,13\}$, then 9 out of 16 bits are still directed to the same 4 column positions after ShiftRow.
3. Consider the 5 columns with an index set $\{0,1,4,12,13\}$, then 12 out of 20 bits are still directed to the same 5 column positions after ShiftRow.
4. Consider the 6 columns with an index set $\{0,1,3,4,12,13\}$, then 15 out of 24 bits are still directed to the same 6 column positions after ShiftRow.

Figure 6 illustrates the 2nd property, and Algorithm 1 presents the procedure of our experiment using this property. Assuming the data complexity required to distinguish two distributions is proportional to the inverse of the squared Euclidean distance *Dis*, then the threshold of *Dis* is 2^{-32}. Our experimental results show that the distinguisher can reach 7 rounds at most using either the 1st or the 4th property , and the distinguisher can reach 8 rounds at most using either the 2nd or the 3rd property. Table 3 gives a part of the results using the 2nd property. Considering the full rounds is 25, we believe there is enough security margin for RECTANGLE against the statistical saturation attack.

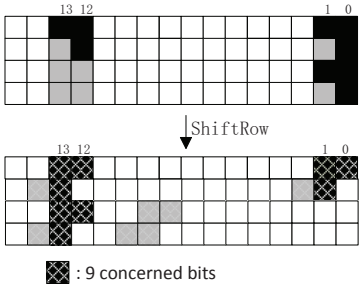

Figure 6. A Weak Property of ShiftRow

Algorithm 1

Set the subkey in each round to a random value.
for $r = 1$ to 10 do
 for $m = 1$ to 37 do

{ 1. Choose a set of 2^m plaintexts which have zero values in the 4 columns with an index set $\{0, 1, 12, 13\}$, while having random values in the other $64 - 16 = 48$ bits.

2. Calculate the distribution of the outputs in the concerned 9 bit positions after r-round encryption, and compute the squared Euclidian distance between this distribution and uniform distribution. Let O denote the output after r-round encryption, j denote the value of the 9-bit string $O_{3,13}||O_{2,13}||O_{2,12}||O_{1,13}||O_{1,1}||O_{0,13}||O_{0,12}||O_{0,1}||O_{0,0}$, the distance is defined as:

$$Dis = \sum_{j=0}^{2^9-1} (\frac{counter[j]}{2^m} - \frac{1}{2^9})^2$$

where $counter_j$ denotes the times of occurence of j among all the 2^m values.

}

Table 3. A Part of Experimental Results using the 2nd Property

m	*Dis* for 7 rounds	*Dis* for 8 rounds	*Dis* for 9 rounds
33	1.37089×2^{-28}	1.81821×2^{-31}	1.392×2^{-33}
34	1.3564×2^{-28}	1.67983×2^{-31}	1.74206×2^{-34}
35	1.34684×2^{-28}	1.62867×2^{-31}	1.24087×2^{-34}
36	1.34594×2^{-28}	1.58832×2^{-31}	1.00152×2^{-34}
37	1.34584×2^{-28}	1.56878×2^{-31}	1.78638×2^{-35}

3.4 Impossible Differential Cryptanalysis

Impossible differential cryptanalysis [6] exploits differential trails with probability 0. Impossible differential distinguishers are usually constructed by meet-in-the-middle approach, that is to say, one differential trail with probability one along the encryption direction and one differential trail with probability one along the decryption direction, whose conditions cannot be met in the middle.

We found some 8-round impossible differential distinguishers for RECTANGLE. Here is one. Firstly, a 4-round differential trail with probability 1 along the encryption direction, then a 4-round differential trail with probability 1 along the decryption direction. More exactly, let (i, j) denote the bit position in the i-th row and j-th column of a cipher state, as Figure 2 shows. Given a pair of round inputs in round 0 which are equal in all bits except one bit position $(2,0)$, then the round outputs in round 3 can not be equal in one bit position $(2,0)$. In the backward direction, given a pair of round outputs in round 7 which are equal in all bits except two bit positions $(2,12)$ and $(3,13)$, then the round inputs in round 4 must be equal in one bit position $(2,0)$. It is obvious that the output in round 3 equals to the input in round 4, thus we have a contradiction.

Notice that the following properties of the RECTANGLE S-box are used to construct the above distinguisher. Let $x = x_3||x_2||x_1||x_0$, where x_i is the i-th bit of x, $i = 0,1,2,3$. Let $(\Delta x \rightarrow \Delta y)$ denote a differential with input difference Δx and output difference Δy. For the S-box of RECTANGLE, $(0100 \rightarrow *1**)$ holds with probability 1, where "$*$" denotes an unknown bit; For the inverse S-box of RECTANGLE, $(1100 \rightarrow ***0)$ holds with probability 1. Since 4-round RECTANGLE reach the full dependency, it is expected that full-round RECTANGLE has enough security against impossible differential cryptanalysis.

3.5 Integral Cryptanalysis

Integral cryptanalysis [30] considers the propagation of sums of many values. An integral distinguisher holds with probability 1.

We found some 7-round higher-order integral distinguishers. Here is one. First is a 4-round integral distinguisher. Choose a set of two plaintexts P and P^*, set $P_{0,1} = P_{0,2} = 0$ and $P^*_{0,1} = P^*_{0,2} = 1$, while they have a fixed random value in the other 62 bit positions. Note that P and P^* only have a non-zero difference in column 0, with $P_{Col(0)} \oplus P^*_{Col(0)} = 0110$. The difference $(0110 \rightarrow *0**)$ holds with probability 1 for the S-box. Then, it can be easily checked that after 4-round encryption the round outputs in round 3 have a zero difference in the following 4 bit positions: $(0,4),(1,5),(2,0),(3,1)$. In other words, the XOR sum in any of the 4 bit positions equals to 0. Next, consider the decryption direction, we can extend the above 4-round distinguisher to a 7-round higher-order integral distinguisher. Choose a set of 2^{56} plaintexts which have certain fixed values in column 0 and column 9, while having all the 2^{56} possible values in the other 14 columns. Then, after 3-round encryption, the 2^{56} intermediate values can be divided into 2^{55} subsets, the two values in each subset satisfy the data condition of the above 4-round distinguisher. If the sum of the elements in each subset can be determined, then the sum of all the elements in the set can also be determined. Hence, after 7-round encryption, the XOR sum of all the 2^{56} outputs in any of the 4 bit positions equals to 0, which is a 7-round integral distinguisher. Similarly, it is expected that full-round RECTANGLE has enough security against integral cryptanalysis.

3.6 Key Schedule Attacks

Among key schedule attacks, the most effective ones are slide attack [8] and related-key cryptanalysis [5]. For RECTANGLE, the adding of different round constants in the key

schedule will prevent slide attacks. For 80-bit user-supplied keys, the union of subkey bits of any consecutive two rounds depends on each of the 80 bits of the user-supplied key. For 128-bit user-supplied keys, the union of subkey bits of any consecutive three rounds depends on each of the 128 bits of the user-supplied key. The choice of the rotation offsets of ShiftRow operation in the round function is different from that in the key schedule. We believe that the above properties are sufficient for RECTANGLE to resist against key schedule attacks.

4 Motivation for Design Choices of RECTANGLE

In this section, we justify the choices we took during design of RECTANGLE.

4.1 Bit-Slice Technique and Lightweight Block Cipher

Consider a 64-bit SP-network block cipher, the S-layer consists of 16 4×4 S-boxes in parallel, thus the subblock length is 16 for a bit-slice implementation. Let a 64-bit state be 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, aiming to provide good diffusion. In such a situation, 16-bit rotations are probably the best choice: they are simple wirings in hardware implementation, they can achieve the goal of mixing up different columns, they can be easily implemented in software using bit-slice technique. So far, we got the framework of RECTANGLE.

4.2 The ShiftRow Transformation

Let c_i ($i = 0, 1, 2, 3$) denote the left rotation offset of the i-th row. The choice criteria of c_i are as follows:

1. The four offsets are different;
2. $c_0 < c_1 < c_2 < c_3$, and $c_0 = 0$;
3. Full dependency after a minimal number of rounds.

Our experimental result shows that there are 16 candidates satisfying the above criteria. For each of the 16 candidates, after 4 rounds each of the 64 input bits influence each of the 64 output bits. From them, we choose $(c_1, c_2, c_3) = (1, 12, 13)$ as the rotation offsets of ShiftRow transformation for RECTANGLE.

4.3 Design Criteria of the S-box

A 4×4 S-box is typically much more compact in hardware than a 8×8 S-box. Serpent uses 8 different S-boxes, however. The use of different S-boxes for different rounds does not result in a plausible improvement of the resistance against known attacks (a variant view from [17]). Moreover, the hardware area can be reduced using only one S-box. Hence, we decide to use only one 4×4 S-box for RECTANGLE.

Let S denote a 4×4 S-box. Let $\triangle I, \triangle O \in F_2^4$, define $ND_S(\triangle I, \triangle O)$ as:

$$ND_S(\triangle I, \triangle O) = \sharp\{x \in F_2^4 | S(x) \oplus S(x \oplus \triangle I) = \triangle O\}.$$

Let $\Gamma I, \Gamma O \in F_2^4$, define the imbalance $Imb_S(\Gamma I, \Gamma O)$ as:

$$Imb_S(\Gamma I, \Gamma O) = |\sharp\{x \in F_2^4 | \Gamma I \bullet x = \Gamma O \bullet S(x)\} - 8|.$$

where $\bullet$ denotes the inner product on F_2^4.

The design criteria of the S-box of RECTANGLE are as follows:

1. Bijective, i.e., $S(x) \neq S(x')$ for any $x \neq x'$.
2. For any non-zero input difference $\triangle I \in F_2^4$ and any non-zero output difference $\triangle O \in F_2^4$, we require:
$$ND_S(\triangle I, \triangle O) \leq 4.$$
3. Let $\triangle I \in F_2^4$ be a non-zero input difference and $\triangle O \in F_2^4$ a non-zero output difference. Let $wt(x)$ denote the Hamming weight of x. Define $SetD1_S$ as:
$$SetD1_S = \{(\triangle I, \triangle O) \in F_2^4 \times F_2^4 | wt(\triangle I) = wt(\triangle O) = 1 \text{ and } ND_S(\triangle I, \triangle O) \neq 0\}.$$
Let $CarD1_S$ denote the cardinality of $SetD1_S$, we require $CarD1_S = 2$.
4. For any non-zero input selection pattern $\Gamma I \in F_2^4$ and any non-zero output selection pattern $\Gamma O \in F_2^4$, we require:
$$Imb_S(\Gamma I, \Gamma O) \leq 4.$$
5. Let $\Gamma I \in F_2^4$ be a non-zero input selection pattern and $\Gamma O \in F_2^4$ a non-zero output selection pattern, define $SetL1_S$ as:
$$SetL1_S = \{(\Gamma I, \Gamma O) \in F_2^4 \times F_2^4 | wt(\Gamma I) = wt(\Gamma O) = 1 \text{ and } Imb_S(\Gamma I, \Gamma O) \neq 0\}.$$
Let $CarL1_S$ denote the cardinality of $SetL1_S$, we require $CarL1_S = 2$.
6. No fixed point, i.e., $S(x) \neq x$ for any $x \in F_2^4$.

Note that $CarD1_S = 0$ and $CarL1_S = 8$ for the S-box of PRESENT, the best shortcut attack on PRESENT [14] uses the fact that $CarL1_S = 8$.

4.4 Selection of the S-box of RECTANGLE

In the following, a S-box means a 4×4 S-box.

Definition 1 ([32]). *Two S-boxes S and S' are called **affine equivalent** if there exist bijective linear mappings A, B and constants $a, b \in F_2^4$ such that $S'(x) = B(S(A(x) + a)) + b$. The equivalence is called **affine equivalence**.*

If a S-box satisfies criteria 1, 2 and 4 (see section 4.3), then any of its affine equivalent S-boxes also satisfies criteria 1, 2 and 4.

Definition 2 ([32]). *Two S-boxes S and S' are called **permutation-then-XOR equivalent** if there exist 4×4 permutation matrices P_0, P_1 and constants $a, b \in F_2^4$ such that $S'(x) = P_1(S(P_0(x) + a)) + b$. The equivalence is called **PE equivalence** for short.*

Table 4. Representatives for all the 4 PE classes of all 4×4 S-boxes fulfilling criteria 1-5

PE_0	6,0,8,15,12,3,7,13,11,14,1,4,5,9,10,2
PE_1	3,2,8,13,15,5,6,10,9,14,4,7,0,12,11,1
PE_2	6,8,15,4,12,7,9,3,11,1,0,14,5,10,2,13
PE_3	8,1,6,12,5,15,10,3,7,11,13,2,0,14,9,4

If a S-box satisfies criteria 1-5, then any of its PE equivalent S-boxes also satisfies criteria 1-5.

It is a surprising fact that all 4×4 S-boxes satisfying criteria 1, 2 and 4 can be classified into only 16 affine equivalence classes [32]. By using the 16 representatives of the 16 affine equivalence classes presented in [32] (Using another set of the 16 representatives in [18], the same result is derived), we have designed an efficient algorithm to classify all 4×4 S-boxes fulfilling criteria 1-5 into PE classes. Our program outputs the result within 3 minutes, it is also a surprising fact that there are only 4 different PE classes . We list a representative for each PE class in Table 4. In each row of Table 4, the first integer represents the image of 0, the second the image of 1, and so on.

Up to adding constants before and after a S-box, which does not change any of the criteria 1-5 and furthermore does not change the probability of the best differential/linear trail for a specific number of rounds, there are $4 \times 4! \times 4! = 2304$ S-boxes that can be generated from the 4 representatives in Table 4. The 2304 S-boxes are denoted as $\{S_i\}, i = 0, 1, \cdots 2303$.

In [14], a multidimensional linear attack is successfully applied on 26-round PRESENT, which mainly uses the fact that there are relatively high number of linear trails with a single active S-box in each round. To avoid such a weakness, we have designed Algorithm 2.

Consider 2-round RECTANGLE, it can be easily seen that the i-th ($i = 0, 1, 2, 3$) bit of a S-box output in the first round will be the i-th bit of a S-box input in the second round. Hence, if the condition in Step 2 holds, it can be easily verified that there exists a differential trail with a single active S-box in each round, for any number of rounds. Similarly, if the condition in Step 4 holds, then there exists a linear trail with a single active S-box in each round, for any number of rounds.

One may say that the permutation layer lets the positions inside the S-box invariant may introduce some weaknesses. However, we believe that it will not be a problem for the security of the cipher, since the S-box has the property that each of the 4 output bits depends on all of the 4 input bits, for 4-round RECTANGLE each of the 64 output bits depends on all of the 64 input bits.

We have implemented Algorithm 2. The result shows that 1776 S-boxes are discarded and only 528 S-boxes are remained. Next, we create a further filtering by considering the security of the underlying cipher against differential and linear cryptanalysis.

Let $Prob\_D15$ denote the probability of the best 15-round differential trail, and Cor_L15 the absolute value of the correlation coefficient of the best 15-round linear trail. Fixing the ShiftRow transformation, for each choice of the remained 528 S-boxes,

Algorithm 2

INPUT: 2304 S-boxes $\{S_i\}, i = 0, 1, \cdots 2303$.
OUTPUT: Discard a part of the S-boxes which can result in a differential (or linear) trail with a single active S-box in each round.

for i = 0 to 2303 do

{ 1. For the i-th S-box S_i, calculate the two $(\triangle I, \triangle O)$ pairs which belong to the set $SetD1_S$. Let $(\triangle I_1, \triangle O_1)$ and $(\triangle I_2, \triangle O_2)$ denote the two pairs, i.e., $wt(\triangle I_i) = wt(\triangle O_i) = 1$ and $ND_S(\triangle I_i, \triangle O_i) \neq 0$, for $i = 1, 2$.
2. If $(\triangle I_1 = \triangle O_1)$ or $(\triangle I_2 = \triangle O_2)$ or $(\triangle I_2 = \triangle O_1$ *and* $\triangle I_1 = \triangle O_2)$, then discard the S-box and $i \leftarrow i + 1$; else go to the following Step 3.
3. For the i-th S-box S_i, calculate the two $(\Gamma I, \Gamma O)$ pairs which belong to the set $SetL1_S$. Let $(\Gamma I_1, \Gamma O_1)$ and $(\Gamma I_2, \Gamma O_2)$ denote the two pairs, i.e., $wt(\Gamma I_i) = wt(\Gamma O_i) = 1$ and $Imb_S(\Gamma I_i, \Gamma O_i) \neq 0$, for $i = 1, 2$.
4. If $(\Gamma I_1 = \Gamma O_1)$ or $(\Gamma I_2 = \Gamma O_2)$ or $(\Gamma I_2 = \Gamma O_1$ *and* $\Gamma I_1 = \Gamma O_2)$, then discard the S-box and $i \leftarrow i + 1$.

}

Table 5. 5 groups of the 32 S-boxes

Group Number	1	2	3	4	5
$Prob\_D15$	66	66	66	66	67
Cor_L15	37	36	35	33	33
Number of S-boxes	4	8	4	8	8

check whether the two inequalities hold for the underlying cipher : $Prob\_D15 < 2^{-64}$ and $Cor\_L15 < 2^{-32}$. Our experimental result shows that only 32 S-boxes satisfy the two inequalities. According to the values of $Prob\_D15$ and Cor_L15, the 32 S-boxes can be divided into 5 groups, we illustrate the result in Table 5.

For the 5 groups, by checking the probability of the best differential trail and the correlation coefficient of the best linear trail up to 15 rounds, we finally choose group 1. There are 4 S-boxes in group 1, which belong to 2 different PE classes. Thus, we only need to consider 2 out of the 4 S-boxes. By adding constants before and after the S-box, we can get $2 \times 16 \times 16 = 512$ different S-boxes. Among the 512 S-boxes, we choose one with no fixed point and low area requirement as the S-box for RECTANGLE.

4.5 The Key Schedule

80-bit key The design criteria of 80-bit key schedule are as follows:

1. Similarity with the design of the round function;
2. The union of subkey bits of any 2 consecutive rounds depends on each of the 80 bits of the user-supplied key;
3. Each bit in the 80-bit key register is a non-linear function of the 80-bit user-supplied key after 21 rounds;
4. Use round constants to eliminate symmetries.

128-bit key See Appendix B.

4.6 The Number of Rounds

The number of rounds is mainly determined by investigating the highest number of rounds which can be distinguished from a random function. Our analysis shows that there exist 14-round (multiple) differential distinguishers and 14-round (multiple or multidimensional) linear distinguishers of RECTANGLE, which are the longest distinguishers known to us. There is no 15-round distinguisher of RECTANGLE, and 4-round RECTANGLE reaches the full dependency, hence it is impossible to attack $15 + 2 \times 4 = 23$ rounds according to the state-of-art security analysis of block ciphers. By adding 2 more redundant rounds, we take 25 as the round number of RECTANGLE.

5 Performance in Various Environments

5.1 Hardware Implementation

We implemented RECTANGLE in Verilog HDL and used Mentor Graphics Modelsim SE PLUS 6.6d for functional simulation. All proposed hardware designs in this paper were synthesized with Synopsys Design Compiler D-2010.03-SP4 to the UMC's $0.13\mu m$.1P8M Low Leakage Standard cell Library with typical values (voltage of $1.2V$ and temperature of $25°C$). A round-based architecture is a direct mapping of the algorithm, which is the most often used for evaluation. Because there is no overhead on multiplexers and flip-flops, this architecture typically has the lowest energy/bit, which can show a good trade off among area, time and throughput. Moreover, this architecture can be straightforwardly reduced to a serial architecture or unfolded into a parallel architecture to meet specific demands for different application scenarios. Due to the limitation of the paper length, we only present the detail of a round-based architecture.

Round-based Architecture Round-based RECTANGLE-80 uses 64/80-bit datapaths for state and key respectively. It performs one round in one clock cycle. The state datapath consists of the 64-bit register for storing, one S-layer, one P-layer and 64-bit XOR of key addition. Except the 80-bit register for key storing, the S-box, P-layer and 5-bit XOR are utilized to update the subkey. A Finite State Machine is used to generate control logic. The plaintext and the key are loaded into each register via multiplexers. Then on each of the following 25 clock cycles, data is read out from the registers, passed through the state and key datapaths and stored back to register respectively. Finally, we can obtain the ciphertext at the output of the 64-bit XOR. Figure 7 illustrates the design diagram of RECTANGLE-80. For the 128-bit version, the state datapath is the same as the 80-bit version, and the key datapath has one more S-box and a different P-layer.

As indicated in Table 6, the area consumption of a round-based RECTANGLE-80 is 1466.25 GE (Gate Equivalent: The size of one NAND gate under specified technology). The most area consuming parts are the flip-flops for the state and key storing, the 17 S-boxes and the 64-bit XOR array. Based on this specified CELL Library, our S-box

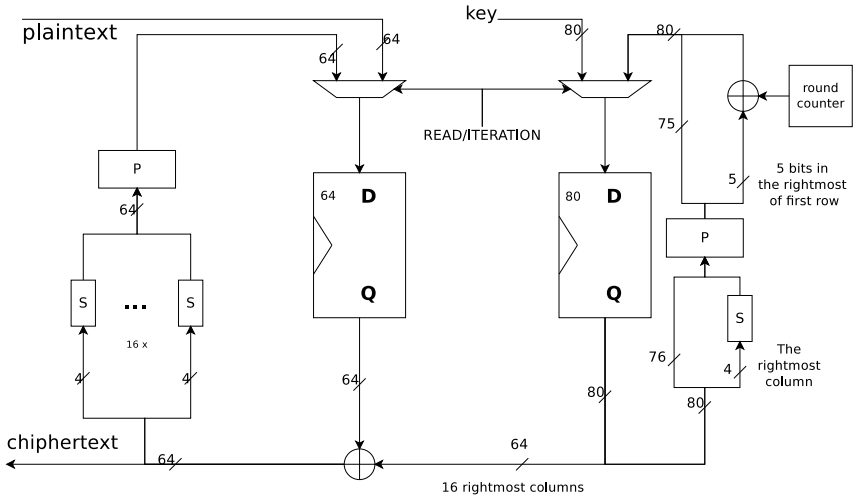

Figure 7. The datapath of the round-based RECTANGLE-80

Table 6. Implementation results of round-based RECTANGLE-80&128

	Key:80-bit		Key:128-bit	
module	Area(GE)	%	Area(GE)	%
data state	400	27.28	400	22.38
s-layer	324	22.10	324	18.13
p-layer	0	0	0	0
key XOR	176	12.00	176	9.85
FSM	3	0.20	3	0.17
KS:key state	500	34.1	800	44.77
KS:S-box	20.75	1.42	41.5	2.32
KS:p-layer	0	0	0	0
KS:counter-XOR	42.5	2.90	42.5	2.38
sum	1466.25	100	1787	100

Table 7. Comparison of light weight cipher implementations (Area vs. Throughput)

	Key size	Block size	Cycles per Block	Tput.At 100KHz(Kbps)	Tech. μm	Area (GE)
Block Ciphers						
RECTANGLE-80	80	64	26	246	0.13	1467
RECTANGLE-128	128	64	26	246	0.13	1787
PRESENT80[43]	80	64	32	200	0.18	1570
AES-128[22]	128	128	1032	12.4	0.35	3400
DESXL[42]	184	64	144	44.4	0.18	2168
Stream Ciphers						
Trivium[26]	80	1	1	100	0.13	2599
Grain[26]	80	1	1	100	0.13	1294

consumes only 20.75 GE. The P-layer is only wiring. The round-based RECTANGLE-80 has a simulated power consumption of 27.3μw at 10MHz. For the round-based RECTANGLE-128 implementation, the area consumption is 1787 GE and the simulated power consumption is 33.0μw at 10MHz.

Table 8. Comparison of 3 different architectures of implementations

	Tech. (μm)	Datapath (Bit)	Freq. (MHz)	Area (GE)	Tput	Energy/Bit (pJ/bit)
Round-based						
RECTANGLE-80	0.13	64	10	1467	24.6$Mbps$	1.11
PRESENT80[43]	0.18	64	10	1570	20.6$Mbps$	3.74
ICEBERG[34]	0.13	64	250	7732	1000.0$Mbps$	9.6
HWang AES[28]	0.18	128	50	79K	582$Mbps$	93
Parallel						
RECTANGLE-80	0.13	64	200	21101	12.8$Gbps$	0.40
PRESENT80[43]	0.18	64	200	27027	10.22$Gbps$	0.67
Serial						
RECTANGLE-80	0.13	4	0.1	1066	13.9$Kbps$	31.7
PRESENT80[43]	0.18	4	0.1	1075	11.4$Kbps$	221.1
Feldhofer AES[23]	0.35	8	0.1	3400	12.4$Kbps$	362.9

Results and Comparisons A comparison of round-based implementations of RECTANGLE and other ciphers follows in Table 7. The throughput is calculated in bits per second. The result in Table 7 illustrates that RECTANGLE has a rather high throughput with a compact area consumption.

Table 8 gives a comparison of the 3 architectures of RECTANGLE-80 and other ciphers. The power consumption is estimated on the gate level by PowerCompiler, based on the switching activates generated by a real testbench. The power strongly depends on the clock frequency and technology. To draw a fair comparison, we use energy per bit to represent the energy efficiency. The results show that RECTANGLE meets the needs under different scenarios and has a rather low energy consumption. The round-based architecture has a good tradeoff between the area and the throughput. The parallel implementation achieve a high throughput rate but consumes the most area and power consumption. The serial design has more ideal compact structure. However, the cost of this area saving is the increasing processing time of 461 cycles.

5.2 Software Implementation

We have implemented RECTANGLE on a 2.70GHz Intel(R) Core i7-2620M CPU running a 64-bit operating system with an Intel C++ compiler.

For one block data, our bit-slice implementation gives a speed at about 36.5 cycles/byte for encryption and 33.5 cycles/byte for decryption. The S-box S can be implemented using a sequence of 12 logical instructions, the P-layer only needs 3 rotations, and the subkey addition only needs 4 XORs. The above 3 functions can be compiled under a low register pressure by the Intel C++ compiler. The inverse S-box can be also implemented using 12 logical instructions. In addition, the memory footprint of the encryption/decryption routines is also very low.

In the case of a parallel mode of operation such as CTR, using Intel 128-bit SSE instructions can give RECTANGLE a very impressive performance. First, consider 8 64-bit blocks, put the first 16 bits of each of the 8 blocks to the first 128-bit vector register, the second 16 bits of each of the 8 blocks to the second 128-bit vector register, and so on. Next, consider messages with x blocks ($1 \leq x \leq 128$). If x is not a multiple of 8, then the corresponding part of the 128-bit registers is set to all zero. Since

RECTANGLE is designed as a bit-sliced cipher, the cost of data load and data format conversion is very low, which takes less than 0.2 cycles/byte when $x \geq 7$. When $40 \leq x \leq 128$ and x is a multiple of 8, it always gives an encryption speed of 5.2 cycles/byte; When $75 \leq x \leq 128$, the encryption speed is lower or equal to 5.6 cycles/byte; When $121 \leq x \leq 128$, it gives an average encryption speed of about 5.38 cycles/byte. Figure 8 illustrates our experimental result.

Our bit-slice implementation of RECTANGLE reaches an average speed of about 5.38 cycles/byte for messages with a length from 120 blocks to 128 blocks (around 1000 bytes). Refer to the latest software performance results of LED and PRESENT presented in [2], the best reported speed is 11.9 cycles/byte and 17.3 cycles/byte respectively for LED and PRESENT, we can see that the software performance of RECTANGLE is quite impressive.

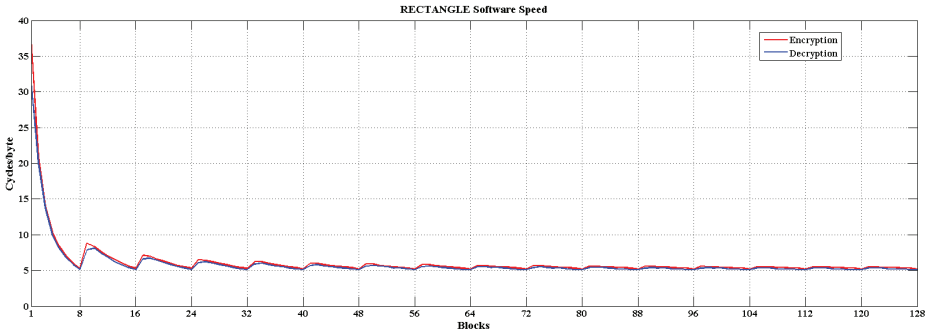

Figure 8. Performance of RECTANGLE with SSE Instruction Set

6 Relation to Early Designs

The main idea of the design of RECTANGLE is to allow lightweight and fast implementations using bit-slice techniques. Serpent and Noekeon are two early bit-slice based block ciphers. However, the design goal of the two ciphers is general-purpose instead of lightweight, almost all aspects need to be reconsidered when it comes to a dedicated lightweight block cipher, including the block length, the key length, the selection of the S-box, the design of the P-layer and the design of the key schedule.

Many block ciphers use parallel 4×4 S-boxes to provide confusion such as Serpent, Noekeon, PRESENT, LED, KLEIN, LBlock and TWINE. In this paper, we proposed new design criteria for 4×4 S-boxes, i.e. $CarD1_S = CarL1_S = 2$. RECTANGLE uses such a new type of S-box. The new criteria are mainly motivated by the existing security analysis of PRESENT, specifically (multiple) differential/linear cryptanalysis on reduced-round PRESENT [14, 40, 46]. Moreover, one can get more confidence in the security of RECTANGLE, by comparing the security of PRESENT and RECTANGLE against (multiple) differential/linear cryptanalysis, which was shown in section 3.1 and 3.2.

Table 9. A summary of Serpent, Noekeon, PRESENT and RECTANGLE

	Serpent	Noekeon	PRESENT	RECTANGLE
block length	128	128	64	64
key length	128, 192, 256	128	80, 128	80, 128
S-box (4×4)	Serpent-type: $CarD1_S = 0$	a new type: $CarD1_S = 7$ $CarL1_S = 6$	Serpent-type: $CarD1_S = 0$	a new type: $CarD1_S = 2$ $CarL1_S = 2$
P-layer	6 32-bit rotations 2 32-bit shifts 8 32-bit XORs	4 32-bit rotations 10 32-bit XORs	a 64-bit permutation	3 16-bit rotations
♯ rounds attacked / ♯ rounds in cipher	12 / 32 [20] (for Serpent-256)	12 / 16 [16]	26 / 31 [14]	18 / 25
hardware implementation	0.35 μm 504000 GE 931.58 Mbps [29] (for Serpent-128)	0.13 μm, 0.1 MHz 4981 GE, 0.178 Mbps 5 pJ/bit [41]	0.18 μm, 10 MHz 1570 GE, 20.6 Mbps, 3.74 pJ/bit [11] (for PRESENT-80)	0.13 μm, 10 MHz 1467 GE, 24.6 Mbps, 1.11 pJ/bit (for RECTANGLE-80)
software implementation	15.2 cycles/byte [38] Intel Core2, E6400@2.13GHz	29.69 cycles/byte [16]	17.3 cycles/byte [2] Intel Core,i3-2367M @ 1.4GHz	5.38 cycles/byte Intel Core, i7-2620M @ 2.7GHz

The design of the P-layer of RECTANGLE depends largely on the bit-slice technique, which is determined by 3 rotation offsets. Compared with the P-layers of Serpent and Noekeon, the P-layer of RECTANGLE is much more friendly for hardware implementation. Furthermore, the P-layer of RECTANGLE is not symmetric, which reduces the differential/linear trail clustering greatly.

We summarize the design, security, hardware implementation and software implementation of the four ciphers in Table 9. Note that the software speed of Noekeon could be improved by a factor of 3 if using 128-bit SSE instructions, i.e., to about 9.9 cycles/byte.

7 Conclusion

We have proposed RECTANGLE, a new lightweight block cipher based on the bit-slice technique. RECTANGLE is a simple design. Largely due to our careful selection of the S-box, RECTANGLE achieves a very good security-performance tradeoff. The highest number of attacked rounds is 18 (out of 25) after we have done an intensive and careful security analysis on RECTANGLE. The bit-sliced design principle allows for very efficient hardware and software implementations. A one-cycle-per-round parallel implementation of RECTANGLE-80 only needs 1467 gates for a throughput of 246 Kbits/sec at 100 KHz clock. It also allows for the lowest energy per bit implementation with only 1.11 pJ/bit. Using Intel 128-bit SSE instructions, a bit-slice implementation of RECTANGLE reaches an average encryption speed of 5.38 cycles/byte for messages around 1000 bytes on a 64-bit operating system.

References

1. Anderson, R., Biham, E., Knudsen, L.R.: Serpent: A Proposal for the Advanced Encryption Standard. NIST AES proposal (1998)
2. Benadjila1, R., Guo, J., Lomn, V., Peyrin, T.: Implementing Lightweight Block Ciphers on x86 Architectures, Cryptology ePrint Archive: Report 2013/445, http://eprint.iacr.org/2013/445.
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak Specifications. NIST SHA-3 Submission (2008), http:// keccak.noekeon.org/.
4. Biham, E.: A Fast New DES Implementation in Software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997)
5. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology 7(4), 229–246 (1994)
6. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
8. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
9. Biryukov, A., De Canni‘ere, C., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
10. Blondeau, C., Gerard, B.: Multiple Differential Cryptanalysis: Theory and Practice. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 35–54. Springer, Heidelberg (2011)
11. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS4727, pp. 450–466. Springer, Heidelberg (2007)
12. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., and Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption, FSE 2013. LNCS. Springer, Heidelberg, to appear.
13. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
14. Cho, J.: Linear Cryptanalysis of Reduced-Round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
15. Collard, B., Standaert, F.X.: A Statistical Saturation Attack against the Block Cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
16. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: Nessie Proposal: the Block Cipher Noekeon, Nessie submission (2000), http://gro.noekeon.org/.
17. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
18. De Cannière, C.: Analysis and Design of Symmetric Encryption Algorithms, Doctoral Dissertaion, KULeuven (2007)
19. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES'09. LNCS, vol. 5747, pp. 272–288. Springer (2009)
20. Dunkelman, O., Indesteege, S., Keller, N.: A Differential-Linear Attack on 12-Round Serpent, In: Chowdhury, D.R., Rijmen, V., and Das, A. (eds.) INDOCRYPT'2008, LNCS, vol. 5365, pp. 308C321. Springer (2008).

21. Engels, D., Saarinen, M.-J.O., Schweitzer, P., Smith, E.M.: The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 19–31. Springer, Heidelberg (2012)
22. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong Authentication for RFID Systems Using the AES Algorithm. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 85–140. Springer, Heidelberg (2004)
23. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on A Grain of Sand. IEE Proceedings on Information Security 152(1), 13–20 (2005)
24. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional Extension of Matsui's Algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS5665, pp. 209–227. Springer, Heidelberg (2009)
25. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS7055, pp. 1–18. Springer, Heidelberg (2012)
26. Good, T., Benaissa, M.: Hardware Results for Selected Stream Cipher Candidates. In: Preproceedings of SASC 2007, pp. 191–204 (2007)
27. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
28. Hwang, D., Tiri, K., Hodjat, A., Lai, B.-C., Yang, S., Schaumont, P., Verbauwhede, I.: AES-Based Security Coprocessor IC in 0.18-μm CMOS with Resistance to Differential Power Analysis Side-Channel Attacks. In: IEEE Transactions on Solid-State Circuits, pp. 781–792 (2006)
29. Ichikawa, T., Kasuya, T., Matsui, M.: Hardware evaluation of the AES finalist. In: The Third Advanced Encryption Standard Candidate Conference, pp. 279–285, (2000)
30. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
31. Leander, G.: On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
32. Leander, G., Poschmann, A.: On the Classification of 4 bit S-boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)
33. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
34. Mace, F., Standaert, F.-X., Quisquater, J.-J.: ASIC Implementations of the Block Cipher SEA for Constrained Applications. In: RFID Security - RFIDsec 2007, Workshop Record, Malaga, Spain, pp. 103–114 (2007)
35. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
36. Matsui, M.: On Correlation between the Order of S-Boxes and the Strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
37. Matsuda, S., Moriai S.: Lightweight Cryptography for the Cloud: Exploit the Power of Bitslice Implementation. In: Prouff E., Schaumont P. (eds.), CHES 2012, LNCS, vol. 7428, 408–425. Springer (2012)
38. Matsui, M., Nakajima, J.: On the Power of Bitslice Implementation on Intel Core2 Processor. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 121–134. Springer, Heidelberg (2007)
39. Naya-Plasencia, M., Peyrin, T.: Practical Cryptanalysis of ARMADILLO2. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 146–162. Springer, Heidelberg (2012)
40. Ohkuma, K.: Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)

41. Plos, T., Dobraunig, C., Hofinger, M., Oprisnik, A., Wiesmeier, C., Wiesmeier, J.: Compact Hardware Implementations of the Block Ciphers mCrypton, NOEKEON, and SEA. In Galbraith, S., Nandi, M. (eds.) Progress in Cryptology-INDOCRYPT 2012. LNCS, vol. 7668, pp. 358–377. Springer, Heidelberg (2012)
42. Poschmann, A., Leander, G., Schramm, K., Paar, C.: A Family of Light-Weight Block Ciphers Based on DES Suited for RFID Applications. In: Workshop on RFID Security 2006 (RFIDSec 2006), Graz, Austria, July 12–14 (2006)
43. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-Lightweight Implementations for Smart Devices - Security for 1000 Gate Equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)
44. Saarinen, M.-J.O.: Cryptanalysis of Hummingbird-1. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 328–341. Springer, Heidelberg (2011)
45. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A Lightweight, Versatile Blockcipher. In: ECRYPT Workshop on Lightweight Cryptography (2011), http://www.uclouvain.be/crypto/
46. Wang, M., Sun, Y., Tischhauser, E., Preneel, B.: A Model for Structure Attacks, with Applications to PRESENT and Serpent. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 49–68. Springer, Heidelberg (2012)
47. Wu, H.: The Hash Function JH. Submission to NIST (2008), http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/jh.pdf
48. Wu, W., Zhang, L.: LBlock: A Lightweight Block Cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)

A The round constants

$\mathrm{RC}[0] = 0X01$, $\mathrm{RC}[1] = 0X02$, $\mathrm{RC}[2] = 0X04$, $\mathrm{RC}[3] = 0X09$,
$\mathrm{RC}[4] = 0X12$, $\mathrm{RC}[5] = 0X05$, $\mathrm{RC}[6] = 0X0B$, $\mathrm{RC}[7] = 0X16$,
$\mathrm{RC}[8] = 0X0C$, $\mathrm{RC}[9] = 0X19$, $\mathrm{RC}[10] = 0X13$, $\mathrm{RC}[11] = 0X07$,
$\mathrm{RC}[12] = 0X0F$, $\mathrm{RC}[13] = 0X1F$, $\mathrm{RC}[14] = 0X1E$, $\mathrm{RC}[15] = 0X1C$,
$\mathrm{RC}[16] = 0X18$, $\mathrm{RC}[17] = 0X11$, $\mathrm{RC}[18] = 0X03$, $\mathrm{RC}[19] = 0X06$,
$\mathrm{RC}[20] = 0X0D$, $\mathrm{RC}[21] = 0X1B$, $\mathrm{RC}[22] = 0X17$, $\mathrm{RC}[23] = 0X0E$,
$\mathrm{RC}[24] = 0X1D$.

B Key schedule for 128-bit version of RECTANGLE and its design criteria

Key Schedule for 128-bit Version For a 128-bit user-supplied key, the key is firstly stored in a 128-bit key register and arranged as a 4×32 array of bits.

The corresponding two-dimensional representation of the 128-bit key state is as follows:

$$\begin{bmatrix} \kappa_{0,31} & \kappa_{0,30} & \cdots & \kappa_{0,2} & \kappa_{0,1} & \kappa_{0,0} \\ \kappa_{1,31} & \kappa_{1,30} & \cdots & \kappa_{1,2} & \kappa_{1,1} & \kappa_{1,0} \\ \kappa_{2,31} & \kappa_{2,30} & \cdots & \kappa_{2,2} & \kappa_{2,1} & \kappa_{2,0} \\ \kappa_{3,31} & \kappa_{3,30} & \cdots & \kappa_{3,2} & \kappa_{3,1} & \kappa_{3,0} \end{bmatrix}$$

At round $i (i = 0, 1, \cdots, 24)$, the 64-bit round subkey K_i consists of the 16 rightmost columns of the current contents of the key. After extracting the round subkey K_i, the key register is updated as follows:

1. Applying the S-box S to column 0 and column 1, i.e.,
$$\kappa_{3,0}||\kappa_{2,0}||\kappa_{1,0}||\kappa_{0,0} := S(\kappa_{3,0}||\kappa_{2,0}||\kappa_{1,0}||\kappa_{0,0})$$
$$\kappa_{3,1}||\kappa_{2,1}||\kappa_{1,1}||\kappa_{0,1} := S(\kappa_{3,1}||\kappa_{2,1}||\kappa_{1,1}||\kappa_{0,1})$$
2. A left rotation to each row over different offsets. Row 0 is left rotated over 10 bits, row 1 is left rotated over 14 bits, row 2 is left rotated over 18 bits, row 3 is left rotated over 22 bits.
3. A 5-bit round constant RC[i] is XORed with the 5-bit key state $(\kappa_{0,4}||\kappa_{0,3}||\kappa_{0,2}||\kappa_{0,1}||\kappa_{0,0})$, where RC[i] $(i = 0, 1, \cdots, 24)$ are the same as those used in the 80-bit key schedule.

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

Design Criteria The design criteria of the 128-bit key schedule are as follows:

1. Same with criterion 1 of the 80-bit version;
2. Each bit in the 128-bit key register is a non-linear function of the 128-bit user-supplied key after 17 rounds;
3. The union of subkey bits of any 3 consecutive rounds depends on each of the 128 bits of the user-supplied key;
4. Same with criterion 4 of the 80-bit version.

C Test Vectors

	Plaintext	Key	Ciphertext
REC-80	0000000000000000	00000000000000000000	0111100100011110
	0000000000000000	00000000000000000000	1100111100101100
	0000000000000000	00000000000000000000	1110111100001001
	0000000000000000	00000000000000000000	1010011101100011
REC-80	1111111111111111	11111111111111111111	1010011011010110
	1111111111111111	11111111111111111111	0110010110010110
	1111111111111111	11111111111111111111	1110101100001111
	1111111111111111	11111111111111111111	0101110000001010
REC-128	0000000000000000	00000000000000000000000000000000	0111000110110011
	0000000000000000	00000000000000000000000000000000	1110101110110101
	0000000000000000	00000000000000000000000000000000	1000011100100011
	0000000000000000	00000000000000000000000000000000	0100000011010000
REC-128	1111111111111111	11111111111111111111111111111111	1101100010111110
	1111111111111111	11111111111111111111111111111111	0001110010111111
	1111111111111111	11111111111111111111111111111111	1011100010110101
	1111111111111111	11111111111111111111111111111111	1010101010111101

D A Bit-slice Description of RECTANGLE

Our essential thoughts for the design of RECTANGLE will become more clear as we consider the bit-slice description of RECTANGLE. In the following, we present an equivalent description of SubColumn and ShiftRow transformations. Based on them, one can easily write a code for a software implementation of RECTANGLE, i.e., a bit-slice implementation. Our software implementation of RECTANGLE is just based on these results. Although the representation of *SubColumn* given here is also suitable for a hardware implementation, we point out that it is not the best one. Indeed, we have found a better representation of *SubColumn* when it comes to area-first hardware implementation.

D.1 SubColumn

As shown in Figure 2, a 64-bit state is described as a 4×16 array. Let $A_i = a_{i,15}||a_{i,14}||\cdots||a_{i,2}||a_{i,1}||a_{i,0}$ denote the i-th row, $i = 0,1,2,3$. A_i can be regarded as a 16-bit word.

Let A_0, A_1, A_2, A_3 be 4 16-bit inputs of SubColumn, B_0, B_1, B_2, B_3 the 4 16-bit outputs, where A_i and B_i denote the i-th row of the cipher state. Let T_i denote 16-bit temporary variables, $i = 1,2,3,5,6,8,9,11,12$. The SubColumn transformation can be computed in the following 12 steps:

1. $T_1 = A_0 \oplus A_1$;
2. $T_2 = A_0 | A_3$;
3. $T_3 = A_2 \oplus T_2$;
4. $B_2 = A_1 \oplus T_3$;
5. $T_5 = A_0 \,\&\, T_3$;
6. $T_6 = A_3 \oplus B_2$;
7. $B_1 = T_5 \oplus T_6$;
8. $T_8 = \neg B_1$;
9. $T_9 = T_3 | T_8$;
10. $B_3 = T_1 \oplus T_9$;
11. $T_{11} = T_8 | B_3$;
12. $B_0 = T_3 \oplus T_{11}$;

D.2 ShiftRow

Let B_0, B_1, B_2, B_3 be 4 16-bit inputs of ShiftRow transformation, C_0, C_1, C_2, C_3 the 4 16-bit outputs. Then:

$C_0 = B_0$; $C_1 = B_1 \lll 1$; $C_2 = B_2 \lll 12$; $C_3 = B_3 \lll 13$.

where "$A \lll (x)$" denotes a left rotation over x bits within a 16-bit word A.

E A Short-cut Attack on 18-round RECTANGLE

E.1 14-round Differential Distinguishers

For 14-round RECTANGLE, the probability of the best differential trail is 2^{-61}. We have searched for all 14-round differential trails with probability between 2^{-61} and 2^{-71} (up to a rotation equivalence), and examined all the difference propagations made up of these investigated trails, the following are the experimental results:

1. There are 32 best difference propagations with probability $81 \times 2^{-67} \approx 2^{-60.66}$ each. Each is composed of 4 differential trails. Among the 4 trails, one with probability 2^{-61}, two with 2^{-64} each, and one with 2^{-67}.
2. Among all the difference propagations, the maximum number of trails of a difference propagation is 114, i.e., a difference propagation is composed of at most 114 different trails. For such a difference propagation, the probability is $500 \times 2^{-71} \approx 2^{-62.03}$.
3. There are 10872 difference propagations with a probability larger than 2^{-64}.

There are 10872 effective 14-round difference propagations, the question is, which one should we choose? Our goal is to investigate what is the highest number of attacked rounds of RECTANGLE. Using one (or some) 14-round difference propagation(s), adding r_b rounds at the beginning and r_e rounds at the end, an attacker means to mount an attack on $14 + r_b + r_e$ rounds. After comparing our attack results by choosing several different 14-round difference propagations, we found that the time complexity depends largely on the number of guessed subkey bits for attacking 17 or 18 rounds. The number of guessed subkey bits mainly depends on two values, one is the number of active bits of the input difference of a difference propagation, the other is the number of active S-boxes of the output difference of the difference propagation. Generally, the smaller the two values, the lower the time complexity. Let *sum* denote the sum of the two values, our experiment shows that the minimum value of *sum* is 4. Among all the effective 14-round difference propagations with $sum = 4$, we choose one with the highest probability. In the following, we present this difference propagation.

A 14-round Difference Propagation of RECTANGLE The difference propagation we have chosen is composed of 2 differential trails. Among the 2 trails, one with probability 2^{-63}, the other with probability 2^{-66}, thus the total probability of this difference propagation is $2^{-63} + 2^{-66} \approx 2^{-62.83}$. The following table gives the input difference and the output difference:

Input Difference of Round 0	Output Difference of Round 13
0000000000000000	0000000000000000
0000000000000001	0000001000010000
0000000000000001	0000000000010000
0000000000000000	0000000000000000

The first differential trail with probability 2^{-63}:

Round Index	Round Prob. ($-log_2$)	Input Difference of the Round	Output Difference of the S-box
0	2	0000000000000000 0000000000000001 0000000000000001 0000000000000000	0000000000000001 0000000000000000 0000000000000000 0000000000000000
1	3	0000000000000001 0000000000000000 0000000000000000 0000000000000000	0000000000000000 0000000000000000 0000000000000000 0000000000000001
2	3	0000000000000000 0000000000000000 0000000000000000 0010000000000000	0010000000000000 0010000000000000 0000000000000000 0000000000000000
3	5	0010000000000000 0100000000000000 0000000000000000 0000000000000000	0000000000000000 0100000000000000 0100000000000000 0010000000000000
4	5	0000000000000000 1000000000000000 0000010000000000 0000010000000000	0000000000000000 1000010000000000 1000000000000000 0000000000000000
5	5	0000000000000000 0000100000000001 0000100000000000 0000000000000000	0000000000000000 0000100000000001 0000000000000001 0000000000000000
6	5	0000000000000000 0001000000000010 0001000000000000 0000000000000000	0000000000000000 0001000000000010 0000000000000010 0000000000000000
7	5	0000000000000000 0010000000000100 0010000000000000 0000000000000000	0000000000000000 0010000000000100 0000000000000100 0000000000000000
8	5	0000000000000000 0100000000001000 0100000000000000 0000000000000000	0000000000000000 0100000000001000 0000000000001000 0000000000000000
9	5	0000000000000000 1000000000010000 1000000000000000 0000000000000000	0000000000000000 1000000000010000 0000000000010000 0000000000000000
10	5	0000000000000000 0000000000100001 0000000000000001 0000000000000000	0000000000000000 0000000000100001 0000000000100000 0000000000000000
11	5	0000000000000000 0000000001000010 0000000000000010 0000000000000000	0000000000000000 0000000001000010 0000000001000000 0000000000000000
12	5	0000000000000000 0000000010000100 0000000000000100 0000000000000000	0000000000000000 0000000010000100 0000000010000000 0000000000000000
13	5	0000000000000000 0000000100001000 0000000000001000 0000000000000000	0000000000000000 0000000100001000 0000000100000000 0000000000000000

The second differential trail with probability 2^{-66}:

Round Index	Round Prob. ($-log_2$)	Input Difference of the round	Output Difference of the S-box
0	2	0000000000000000 0000000000000001 0000000000000001 0000000000000000	0000000000000001 0000000000000000 0000000000000000 0000000000000000
1	3	0000000000000001 0000000000000000 0000000000000000 0000000000000000	0000000000000001 0000000000000000 0000000000000000 0000000000000001
2	6	0000000000000001 0000000000000000 0000000000000000 0010000000000000	0010000000000000 0010000000000000 0000000000000000 0000000000000001
3	5	0010000000000000 0100000000000000 0000000000000000 0010000000000000	0000000000000000 0100000000000000 0100000000000000 0010000000000000
4	5	0000000000000000 1000000000000000 0000010000000000 0000010000000000	0000000000000000 1000010000000000 1000000000000000 0000000000000000
5	5	0000000000000000 0000100000000001 0000100000000000 0000000000000000	0000000000000000 0000100000000001 0000000000000001 0000000000000000
6	5	0000000000000000 0001000000000010 0001000000000000 0000000000000000	0000000000000000 0001000000000010 0000000000000010 0000000000000000
7	5	0000000000000000 0010000000000100 0010000000000000 0000000000000000	0000000000000000 0010000000000100 0000000000000100 0000000000000000
8	5	0000000000000000 0100000000001000 0100000000000000 0000000000000000	0000000000000000 0100000000001000 0000000000001000 0000000000000000
9	5	0000000000000000 1000000000010000 1000000000000000 0000000000000000	0000000000000000 1000000000010000 0000000000010000 0000000000000000
10	5	0000000000000000 0000000000100001 0000000000000001 0000000000000000	0000000000000000 0000000000100001 0000000000100000 0000000000000000
11	5	0000000000000000 0000000001000010 0000000000000010 0000000000000000	0000000000000000 0000000001000010 0000000001000000 0000000000000000
12	5	0000000000000000 0000000010000100 0000000000000100 0000000000000000	0000000000000000 0000000010000100 0000000010000000 0000000000000000
13	5	0000000000000000 0000000100001000 0000000000001000 0000000000000000	0000000000000000 0000000100001000 0000000100000000 0000000000000000

E.2 An Attack on 18-round RECTANGLE

Notations

– For r-round RECTANGLE, the subkey K_i is used in round $i(i = 0, 1, 2, r-1)$, finally a subkey XOR with K_r.
– Let X_i^I, X_i^{SO} and X_i^O respectively denote the input, the intermediate value after the application of SubColumn and the output of round i.
– A cipher state has 16 columns, as Figure 2 shows, let $(X_i)_{Col(j)}$ denote column j of X_i, $j = 0, 1, 2, \cdots, 15$.

Set the 14-round difference propagation at rounds 2-15, adding two rounds at the beginning, adding two rounds and a final subkey XOR at the end. The attacker needs to guess some subkey bits of K_0, K_1, K_{17} and K_{18}. To get the input difference of round 2, the attacker needs to guess 28 bits of K_0 and 8 bits of K_1, then partially encrypts the plaintexts. To get the output difference of round 15, the attacker needs to guess 28 bits of K_{18} and 8 bits of K_{17}, then partially decrypts the ciphertexts. For right pairs, there are 9 non-active S-boxes in round 17, thus the corresponding ciphertext difference in the $9 \times 4 = 36$ bit positions must be zero. The attack algorithm is as follows:

1. Choose a set of 2^{28} plaintexts which have certain fixed values in all but 7 columns 2,3,4,7,8,14,15. We call this a structure, one structure can form about $2^{28} \times (2^{28} - 1) \approx 2^{55}$ plaintext pairs. Generate 2^m structures, thus 2^{m+28} plaintexts, 2^{m+55} plaintext pairs.
2. For each structure:
   (a) Insert all the ciphertexts into a hash table indexed by the 36 non-active bits. Choose only the pairs in the same entry, i.e., the ciphertext pairs having zero difference in these 36 bits. The expected number of such pairs is $2^{m+55-36} = 2^{m+19}$.
   (b) Guess the value of a part of subkey bits of K_0, and do partial encryption:
      i. Guess the 4 subkey bits $(K_0)_{Col(2)}$, and encrypt the plaintext pairs to get the difference of $(X_0^{SO})_{Col(2)}$. Check up whether the difference in the 3 bit positions (0,1,2) of $(X_0^{SO})_{Col(2)}$ are zero. If no, then discard the pair. The expected remaining pairs is 2^{m+16}.
      ii. Respectively guess the 4 subkey bits $(K_0)_{Col(i)}(i = 4, 7, 8, 14, 15)$, and encrypt the remaining pairs to get the difference of $(X_0^{SO})_{Col(i)}$. Check up whether the difference in 3 bit positions of $(X_0^{SO})_{Col(i)}$ are zero. If no, discard the pair. The expected remaining pairs is 2^{m+1}.
      iii. Guess the 4 subkey bits $(K_0)_{Col(3)}$, and encrypt the remaining pairs to get the difference of $(X_0^{SO})_{Col(3)}$. Check up whether the difference in the 2 bit positions (0,4) of $(X_0^{SO})_{Col(3)}$ are zero. If no, then discard the pair. The expected remaining pairs is $2^{m+1-2} = 2^{m-1}$.
   (c) Guess the value of a part of subkey bits of K_1, and do partial encryption:
      i. Guess the 4 subkey bits $(K_1)_{Col(4)}$, and encrypt the remaining 2^{m-1} pairs to get the difference of $(X_1^{SO})_{Col(4)}$. Check up whether the difference in $(X_0^{SO})_{Col(4)}$ equals to 4. If no, then discard the pair. The expected remaining pairs is 2^{m-5}.
      ii. Guess the 4 subkey bits $(K_0)_{Col(15)}$, and encrypt the remaining 2^{m-5} pairs to get the difference of $(X_1^{SO})_{Col(15)}$. Check up whether the difference in

$(X_0^{SO})_{Col(15)}$ equals to 2. If no, then discard the pair. The expected remaining pairs is 2^{m-9}.

(d) Guess the value of a part of subkey bits of K_{18}, and do partial decryption:
   i. Guess the 4 subkey bits $(ShiftRow^{-1}(K_{18}))_{Col(0)}$, and decrypt the ciphertext pairs to get the difference of $(X_{17}^I)_{Col(0)}$. Check up whether the difference in the 3 bit positions (0,1,3) of $(X_{17}^I)_{Col(0)}$ are zero. The expected remaining pairs is 2^{m-12}.
   ii. Respectively guess the 4 subkey bits $(ShiftRow^{-1}(K_{18}))_{Col(i)} (i = 1,4,6,9,10)$, and decrypt the ciphertext pairs to get the difference of $(X_{17}^I)_{Col(i)}$. Check up whether the difference in 3 bit positions of $(X_{17}^I)_{Col(i)}$ are zero. The expected remaining pairs is 2^{m-27}.
   iii. Guess the 4 subkey bits $(ShiftRow^{-1}(K_{18}))_{Col(5)}$, and decrypt the ciphertext pairs to get the difference of $(X_{17}^I)_{Col(5)}$. Check up whether the difference in 2 bit positions of $(X_{17}^I)_{Col(3)}$ are zero. The expected remaining pairs is 2^{m-29}.

(e) Guess the value of a part of subkey bits of K_{17}, and do partial decryption:
   i. Guess the 4 subkey bits $(ShiftRow^{-1}(K17))_{Col(4)}$, and decrypt the pairs to get the difference of $(X_{16}^I)_{Col(4)}$. Check up whether the difference in $(X_{16}^I)_{Col(4)}$ is equal to 6. If no, then discard the pairs. The expected remaining pairs is 2^{m-33}.
   ii. Guess the 4 subkey bits $(ShiftRow^{-1}(K17))_{Col(9)}$, and decrypt the pairs to get the difference of $(X_{16}^I)_{Col(9)}$. Check up whether the difference in $(X_{16}^I)_{Col(9)}$ is equal to 2. If so, add one to the corresponding counter. The expected remaining pairs is 2^{m-37}.

(f) If the number of the remaining pairs is larger than 1, then keep the guess of the subkey bits as the candidates of the right subkeys.

(g) For the survived candidates of the right subkeys, compute the seed key by doing an exhaustive search to find the unique right seed key.

Let us evaluate the time complexity of step 2.a - step 2.e first. Step 2.a requires 2^{m+28} memory accesses. Step 2.b.i requires about $2 \times 2^{m+19} \times 2^4 = 2^{m+24}$ one-round encryptions, step 2.b.ii requires about $2 \times 2^{m+16} \times 2^8 + 2 \times 2^{m+13} \times 2^{12} + 2 \times 2^{m+10} \times 2^{16} + 2 \times 2^{m+7} \times 2^{20} + 2 \times 2^{m+4} \times 2^{24} = 2^{m+25} + 2^{m+26} + 2^{m+27} + 2^{m+28} + 2^{m+29}$ one-round encryptions, step 2.b.iii requires about $2 \times 2^{m+1} \times 2^{28} = 2^{m+30}$ one-round encryptions. Step 2.c.i requires about $2 \times 2^{m-1} \times 2^{32} = 2^{m+32}$ one-round encryptions, step 2.c.ii requires about $2 \times 2^{m-5} \times 2^{36} = 2^{m+32}$ one-round encryptions. Step 2.d.i requires about 2^{m+32} one-round encryptions, step 2.d.ii requires about $2^{m+33} + 2^{m+34} + 2^{m+35} + 2^{m+36} + 2^{m+37}$ one-round encryptions, step 2.d.iii requires about 2^{38} one-round encryptions. Step 2.e.i requires about $2 \times 2^{m-29} \times 2^{68} = 2^{40}$ one-round encryptions, step 2.e.i requires about $2 \times 2^{m-33} \times 2^{72} = 2^{40}$ one-round encryptions. The overall time complexity of step 2.a - step 2.e is about $2^{m+41.32}$ one-round encryptions, or $2^{m+37.15}$ 18-round encryptions.

Let $m = 36$. The data complexity of the attack is 2^{64} plaintexts. The memory complexity is 2^{72} key counters. For wrong key guesses, the expected remaining pairs is

$1/2$. For the right key guess, the expected remaining pairs is $2^{m+55-28-62.83} \approx 1.125$. By the Poisson distribution, $X \sim Poi(\lambda = 1/2)$, $Prob[X >= 1] \approx 2^{-1.34}$, thus the time complexity of step 2.g is $2^{80-1.34} = 2^{78.66}$ for a 80-bit user-supplied key, the overall time complexity of the attack is about $2^{36+37.15} + 2^{78.66} \approx 2^{78.69}$ 18-round encryptions. For a 128-bit user-supplied key, the overall time complexity of the attack is about $2^{128-1.34} = 2^{126.66}$ 18-round encryptions.

By the Poisson distribution, $X \sim Poi(\lambda = 1.125)$, $Prob[X >= 1] \approx 67.5\%$, so the success rate of the above attack is about 67.5%.

E.3 On 14-round Linear Distinguishers

For 14-round RECTANGLE, we also investigated the clustering of linear trails by searching for all the linear trails with an amplitude of the correlation coefficient between 2^{-34} and 2^{-37}. The results show that the best 14-round linear propagation has an amplitude of the average correlation $2^{-31.62}$. Considering the two facts: (1). the gap between $2^{-31.62}$ and 2^{-32} is very small; (2). let h_1 and h_2 respectively denote the bit hamming weight of the input and output selection pattern, then the minimum value of $h_1 + h_2$ is 7 among all the effective 14-round linear distinguishers. Compared with the case in differential cryptanalysis, we prefer to believe that it is better to use 14-round differential distinguisher rather than 14-round linear distinguisher with respect to the highest number of attacked rounds.