RECTANGLE

Team SouL

Department of EECS
Indian Institute of Technology Bhilai

November 27, 2020

Outline

1. Introduction
2. Cipher Specifications
3. Observations
4. Brownie Point Nominations
5. Conclusion

Introduction

- Increased usage of small embedded devices like RFIDs etc, in many applications
- The data exchanged and stored needs cryptographic protection
- Symmetric key block ciphers play a crucial role in their security
- Introduction of lightweight ciphers to provide cost-effective security.

Introduction

- Quite a few lightweight ciphers have been proposed but many don't provide high performance or security.(tradeoff b/w security and performance)
- PRESENT is one such cipher which provides good hardware performance and security and is liked for its simplicity.
- But lacks high software performance needed in classical lightweight operations and is weak in security.

RECTANGLE

RECTANGLE allows fast and lightweight implementations using bitslicing.
How is it better?

- Extremely hardware friendly
- Implementation of bit-slicing gives good software performance/speed compared to other ciphers.
- Asymmetric design in permutation layer of the S-Box achieves a good performance-security tradeoff.

Outline

1 Introduction

2 Cipher Specifications

3 Observations

4 Brownie Point Nominations

5 Conclusion

Specifics

- RECTANGLE is a Block Cipher which uses Substitution and Permutation Network (SPN).
- Each Round of S layer has 16 , 4x4 S boxes in parallel which is called SubColumn.
- P layer has 3 rotations which is called ShiftRows and addition of round key (XOR of state matrix and round key).
- Block length: 64 bits
- Key length: 80 or 128 bits
- Number of Rounds is 25.

Algorithm

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	6	5	c	a	1	e	7	9	b	0	3	d	8	f	4	2

Table: S-box of RECTANGLE

```
GenerateRoundKeys();
for i in 0..25 {
   // For all the 25 rounds
      AddRoundKey(STATE, Ki);
      // State matrix is XORed with round subkey.
      SubColumn(STATE);
      // Each column of state is passed through Sbox (S layer)
      ShiftRow(STATE);
      // Each row of the state is rotated (P layer)
}
AddRoundKey(STATE, K25); // final subkey XOR
```

Key Schedule

For an 80-bit seed key the key is firstly stored in an 80-bit key register and arranged as a 5×16 array of bits

S-box: S-box is applied to 4 rightmost columns and 4 rows

Feistal: The 1-round 5-subblock generalized Feistel transformation is used to provide appropriate diffusion.

$1.row_0 = (row_0 << 8) \oplus row_1$ $2.row_1 = row_2$

$3.row_2 = row_3$ $4.row_3 = (row_3 << 12) \oplus row_4$

$5.row_4 = row_0$

Xor:

Last five bits of 0th row of key-state are XORed with round constant of current round.Use of round constants to eliminate symmetries.

Bit Slicing Technique

- 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 sub-block.
- The bit-sliced design principle allows for both low-cost hardware and efficient software implementations.
- Due to its bit-slice style, RECTANGLE achieves a very competitive software speed among the existing lightweight block ciphers.
- The S-box of RECTANGLE can be implemented using a sequence of 12 basic logical instructions. The P-layer of RECTANGLE is composed of 3 rotations, which makes it very friendly for both hardware and software implementations.

Bit Slicing Technique

The Sub-Column transformation can be computed in the following 12 steps:

1. $T_1 = \neg A_1$;
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. $B_2 = T_6 \oplus T_{11}$;

where "$\neg$" is NOT (negation), "&" is bit-wise AND and "$|$" is bit-wise OR.

Security

Due to careful selection of S box and asymmetric design of the P layer , RECTANGLE achieves a very good security-performance trade-off. The generalized Feistel transformations are designed to provide appropriate diffusion.

- Highest number of attacked rounds is 18.
- So, it is decided to add 7 rounds as a security margin.

A bit-slice implementation of RECTANGLE is safe against implementation attacks such as cache and timing attacks compared with a table-based implementation.

Outline

1 Introduction

2 Cipher Specifications

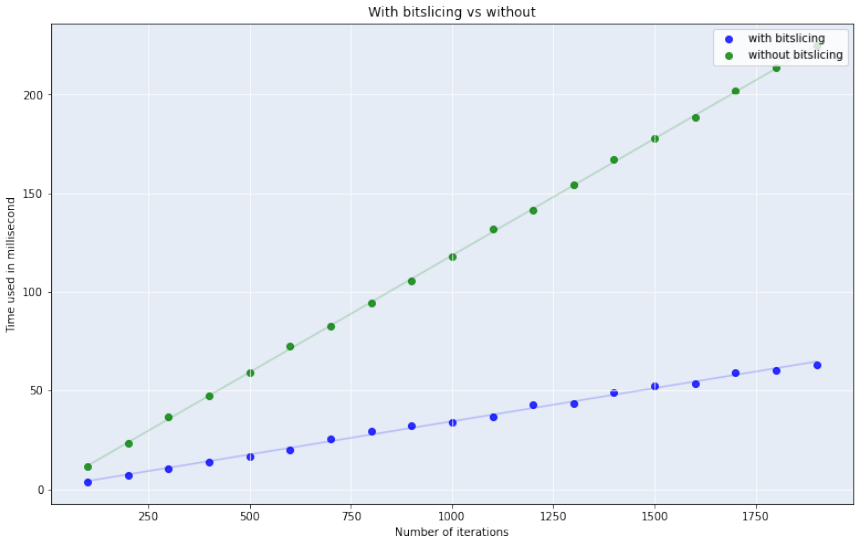
3 Observations

4 Brownie Point Nominations

5 Conclusion

Speed of RECTANGLE

Speed with bitslicing:
34.661±1.186 µs
Speed without bitslicing:
118.429±1.148 µs
(for one encryption of 64 bit)
Measured with implementation in Rust



Linear Cryptanalysis

- RECTANGLE is resistant to linear cryptanalysis attacks.
- Maximum number of rounds that can be attacked using linear cryptanalysis is 14. There is no clustering of linear trails which can make effective distinguisher for more than 15 rounds.
- The authors of RECTANGLE's paper used a modified version of M. Matsui's search algorithm for finding all differential/linear trials which uses branch and bound method. Best differential trails from round 1 to round 15 are obtained using this algorithm

Linear Cryptanalysis

Theorem 1. The square of a correlation (or correlation contribution) 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:

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

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

Differential Cryptanalysis

- Maximum number of rounds that can be attacked using linear cryptanalysis is 14.
- A modified version used in Search algorithm (Matsui M) is also used here to find difference propagations with highest probability
- There is no clustering of linear trails which can make effective distinguisher for more than 15 rounds.

Difference Propagation

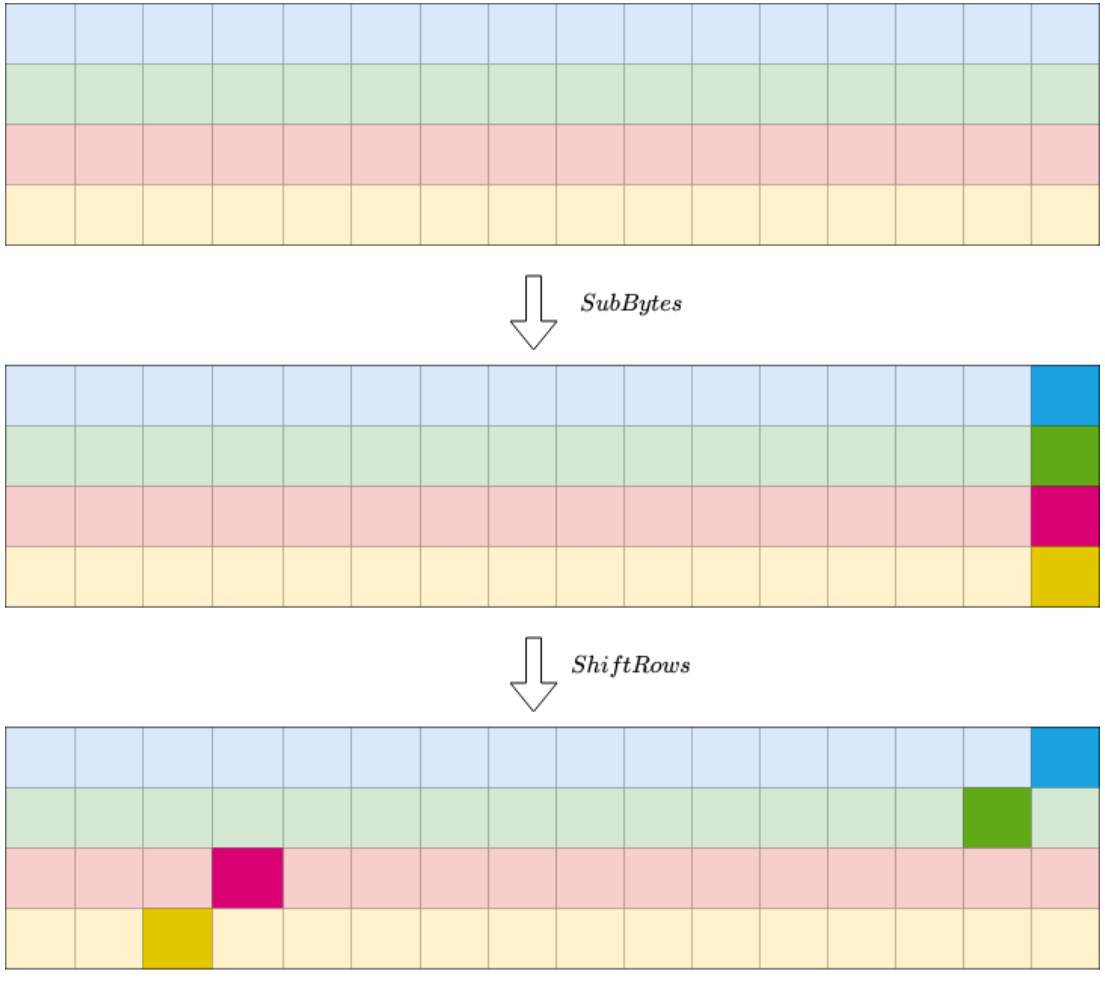


Figure: Difference Propagation 1

Difference Propagation

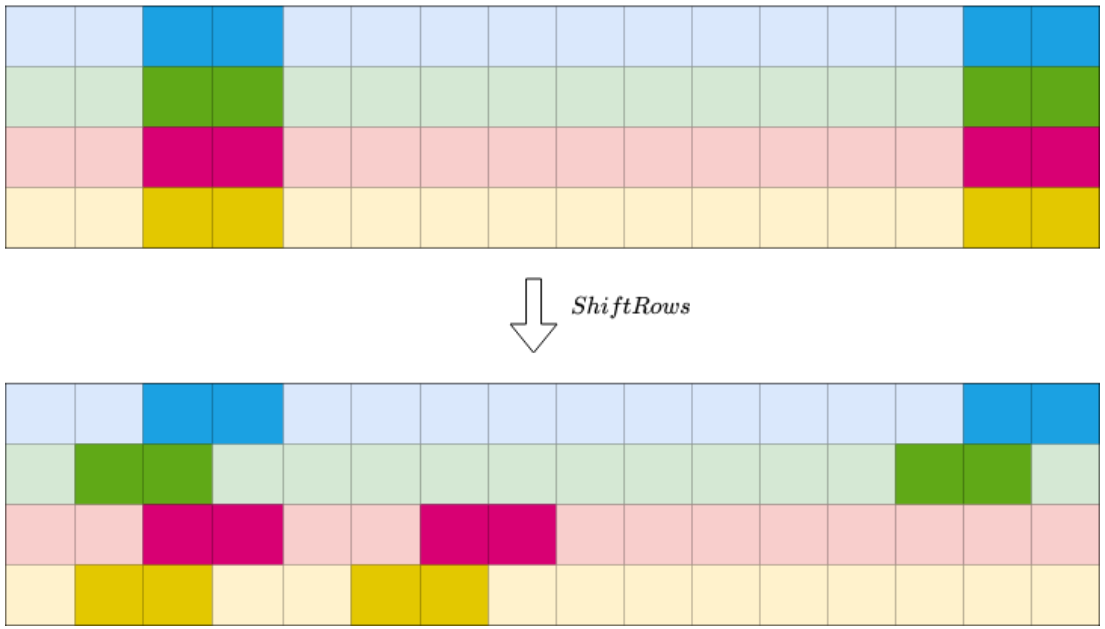


Figure: Difference Propagation 2

Difference Propagation

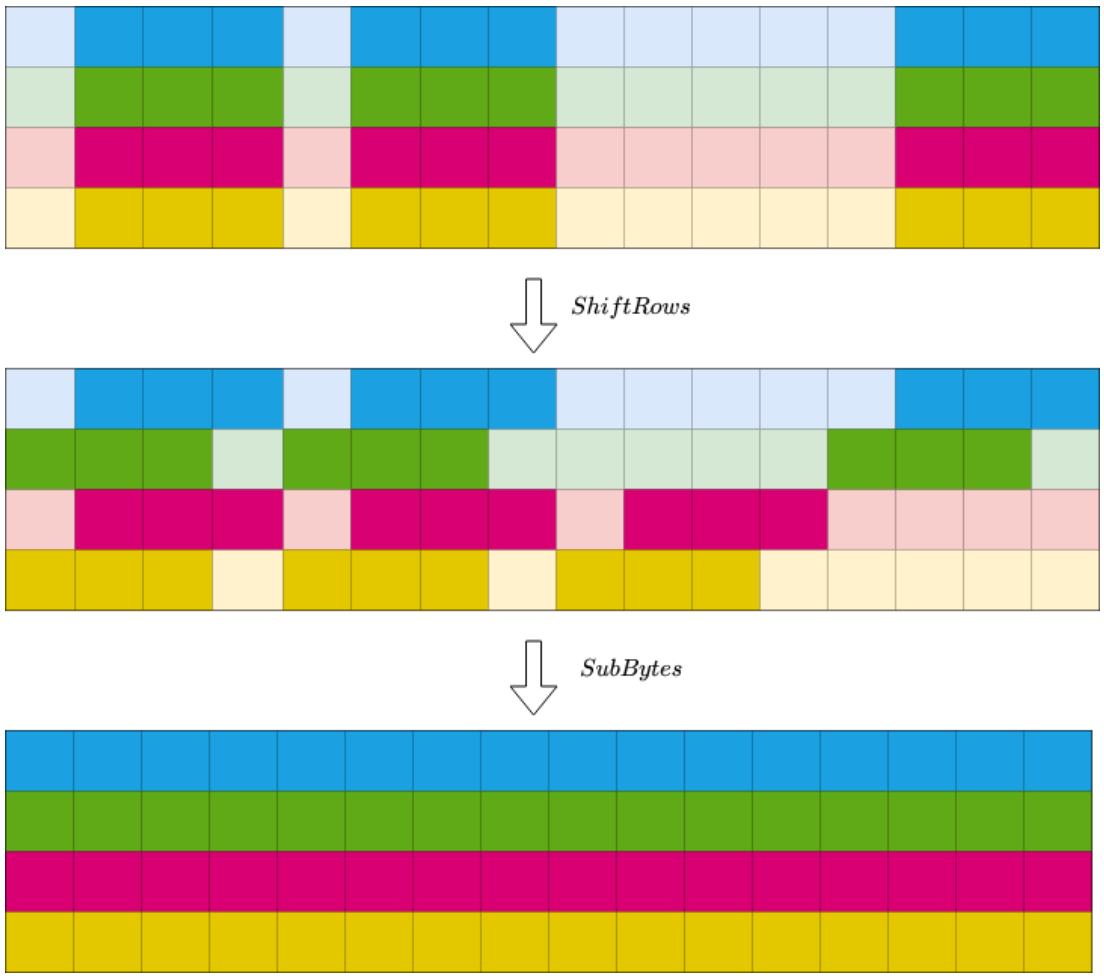


Figure: Difference Propagation 3

Performance

	Key Size	Block Size	Cycles per Block	Tech.(micro m)	Area(GE)	Throughput(Kbps)
AES-128	128	128	226	0.13	2400	56.6
LED-64	64	64	1248	0.18	966	5.1
PICCOLO-80	80	64	27	0.13	1496	237
PRESENT-80	80	64	26	0.13	1570	200
RECTANGLE-80	80	64	26	0.13	1599.5	246
RECTANGLE-128	128	64	26	0.13	2063.5	256

Outline

1 Introduction

2 Cipher Specifications

3 Observations

4 Brownie Point Nominations

5 Conclusion

Point-1

Several visualizations were added wherever required for easy understanding.

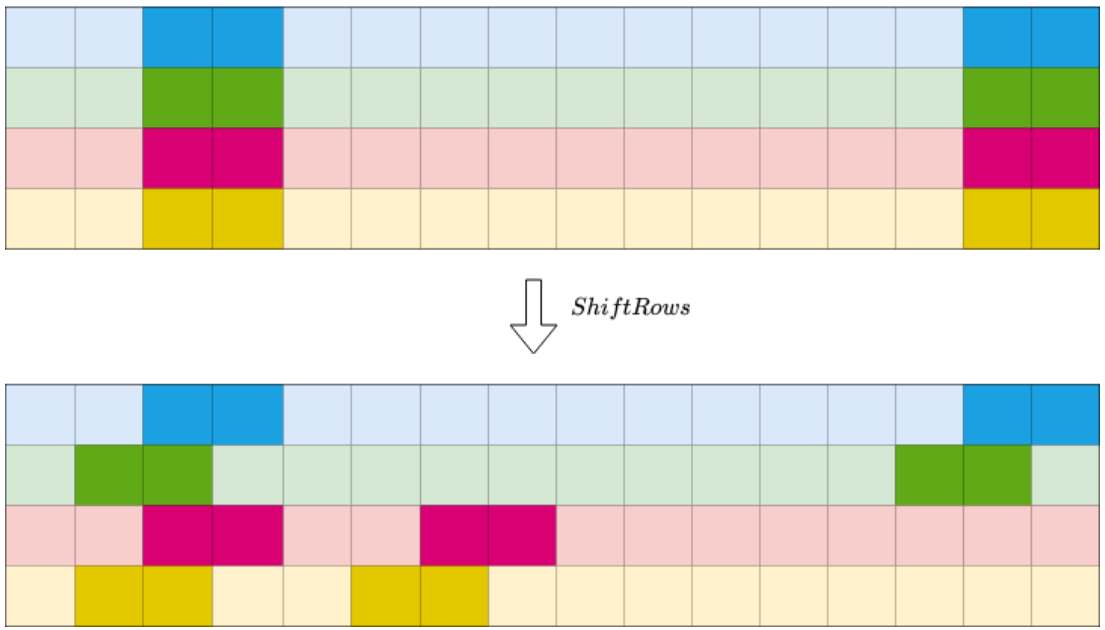
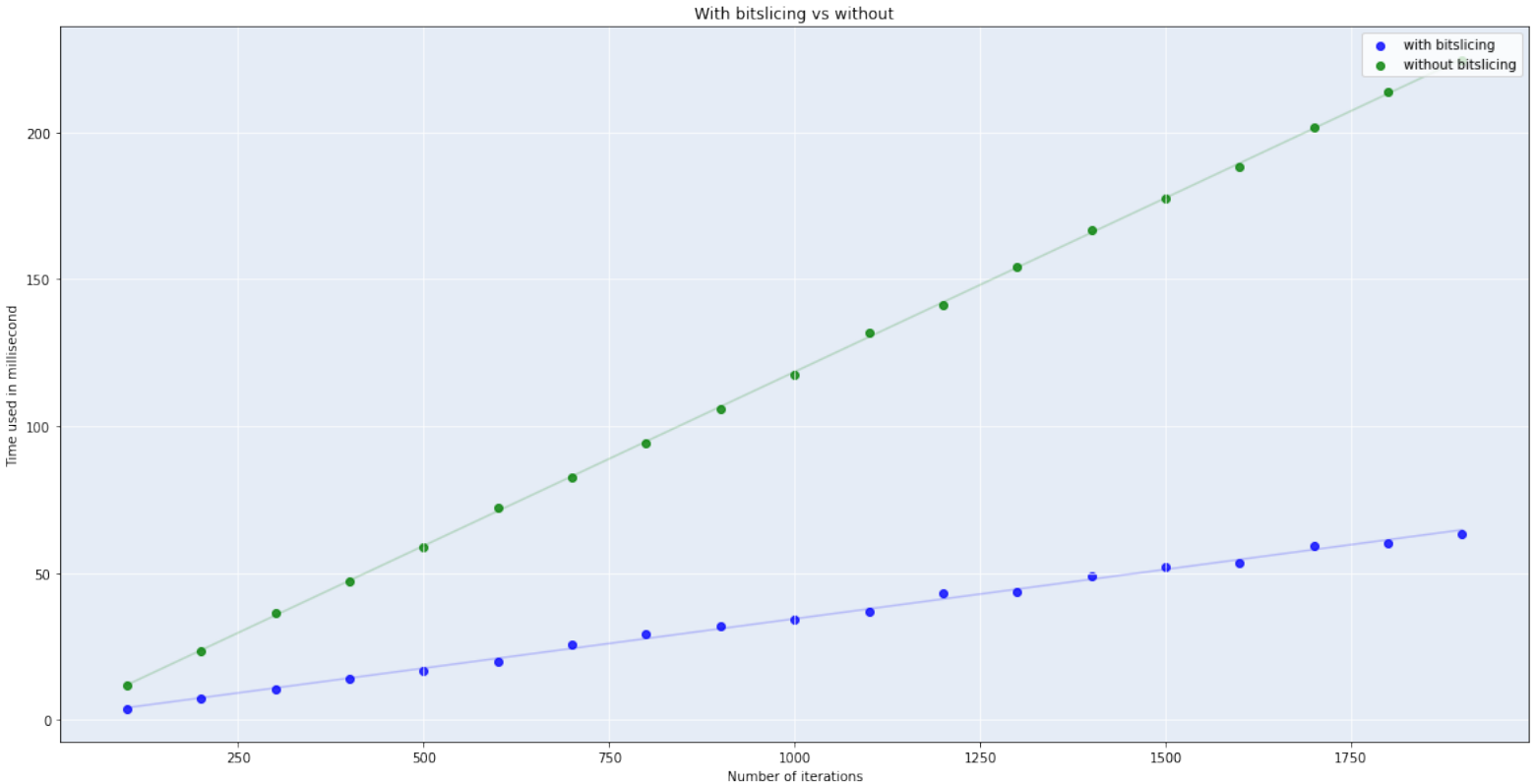


Figure: Difference Propagation

Point-2

Comparison of Performance in software with and without bit slicing techniques.



Point-3

Detailed and easy to understand explanations for Algorithm.

2.3 Rounds

Following operations are applied in each round.
AddRoundKey
State matrix is XORed with roundkey.
(Note: In the following figures used for Rounds section, each block represents one bit)

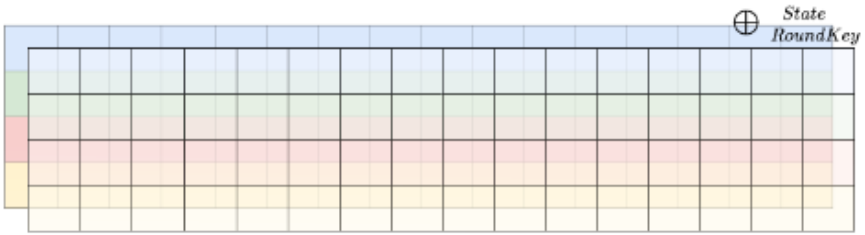


Figure 4: AddRoundKey

SubColumn
Each column of state is passed through Sbox. This operation (SubColumn) can be efficiently implemented using bitslicing instead of using Sbox on each column.

Outline

1 Introduction

2 Cipher Specifications

3 Observations

4 Brownie Point Nominations

5 Conclusion

Conclusion

We have done a survey of lightweight cipher, RECTANGLE.
Algorithm, rounds and key schedule of the cipher are covered in detail.
The significance of bit-slicing techniques can be seen in the performance of the RECTANGLE cipher compared to others.
An implementation in Rust is also made along with speed measurements.

Conclusion

Various observations are made on the design of cipher, Sbox, difference propagation.
Linear cryptanalysis and differential cryptanalysis, the most powerful techniques for attacks are covered.
Review on performance of RECTANGLE in software and hardware is done. Finally unique aspects of this paper are summarized in the Brownie point section.

Thanks

Team Members

- Ambatipudi Abhiram (11840170)
- Kausheek Akella (11840120)
- Manu E (11840700)

Implementation Info

- Github Link:
  `https://github.com/manu156/CS553-Rectangle.git`