Innovative R&D by NTT

Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs

Jian Guo[1], Jeremy Jean[2], Ivica Nikolić[1], Kexin Qiao[3], Yu Sasaki[4], and Siang Meng Sim[1]

1. Nanyang Technological University, Singapore
2. ANSSI, Paris, France
3. Institute of Information Engineering, Chinese Academy of Sciences, China
4. NTT Secure Platform Laboratories

Midori: a low energy block cipher proposed at Asiacrypt2015

Invariant subspace attack [Lender++ CRYPTO2011]

- Weak key attack on Midori64 with 2^{32} weak keys.
- Distinguisher with 1 chosen plaintext.
- Key recovery with 2^{16} computations.

Feedback to Design

- Searching for S-boxes avoiding invariant subspace attack for any choice of constant.

Attack Overview

Suppose key nibbles $\in \{0,1\}$*, plaintext nibbles* $\in \{8,9\}$*, then ciphertext nibbles* $\in \{8,9\}$*.*

Appendix A: Test Vectors [ePrint2015/1142]

Plaintext	:	0000000000000000
Key	:	00000000000000000000000000000000
Ciphertext	:	3c9cceda2bbd449a

Test vector uses a weak key.

Our experiment

Plaintext	:	8888888888888888
Key	:	00000000000000000000000000000000
Ciphertext	:	9998899889888899

Invariant Subspace Attack [LAA2011]

Find subsets, $(\mathcal{S}, \mathcal{K})$, of the state space and key space which are invariant of the round function.

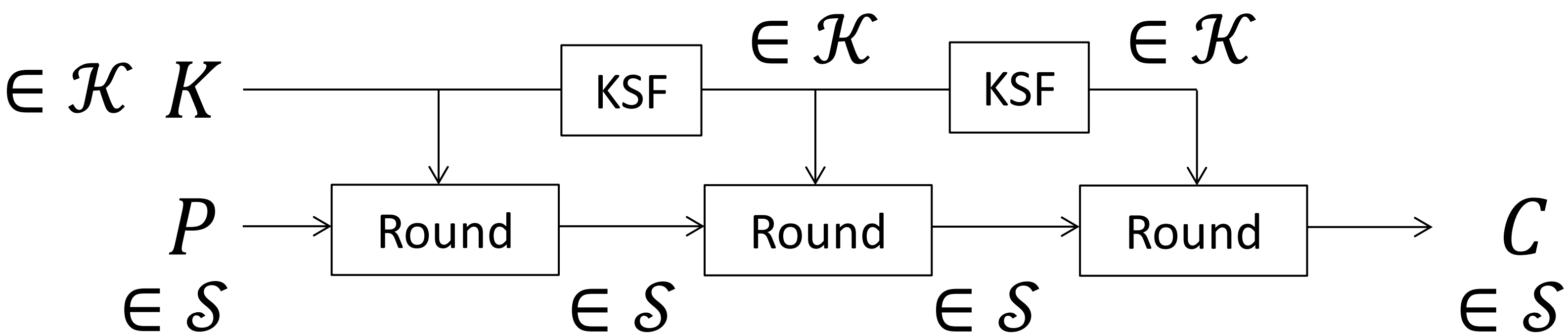
Encrypt plaintext $P \in \mathcal{S}$ under the key $K \in \mathcal{K}$ (weak-key attack). Ciphertext also belongs to $\mathcal{S}$.

All subkeys must be in $\mathcal{K}$. The main target is ciphers with no key schedule, common structure for lightweight cryptography

Applications: PRINTcipher, Robin, iScream, Zorro

Invariant Subspace Attack [LAA2011]

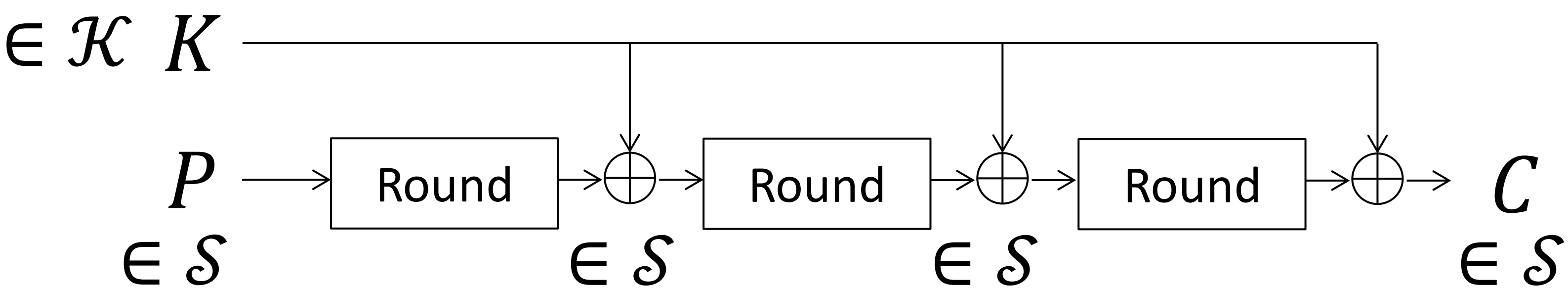
General Form:



- A weak-key distinguisher often with 1 query.
- Possibility of the extension to key recovery depends on the cipher's structure.

Invariant Subspace Attack [LAA2011]

In practice:



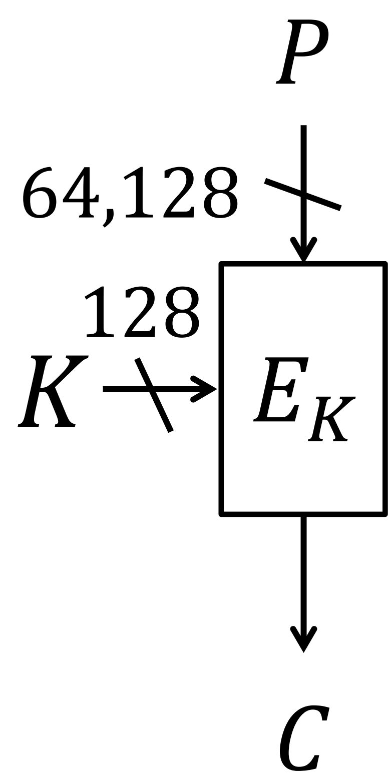
- $\mathcal{S}$ is an affine space with dimension i, namely $< x_1, x_2, \cdots, x_i > \oplus u$, where $< \cdots >$ is a linear space and u is a constant.
- $K \in < x_1, x_2, \cdots, x_i >$ preserves subspace even after subkey XOR.

Block Cipher Midori

"Midori: A Block Cipher for Low Energy"

Banik et al. at Asiacrypt 2015

- Midori64: 64-bit block, 128-bit key
  SPN with 4-bit cell, 16 rounds
- Midro128: 128-bit block, 128-bit key
  SPN with 8-bit cell, 20 rounds



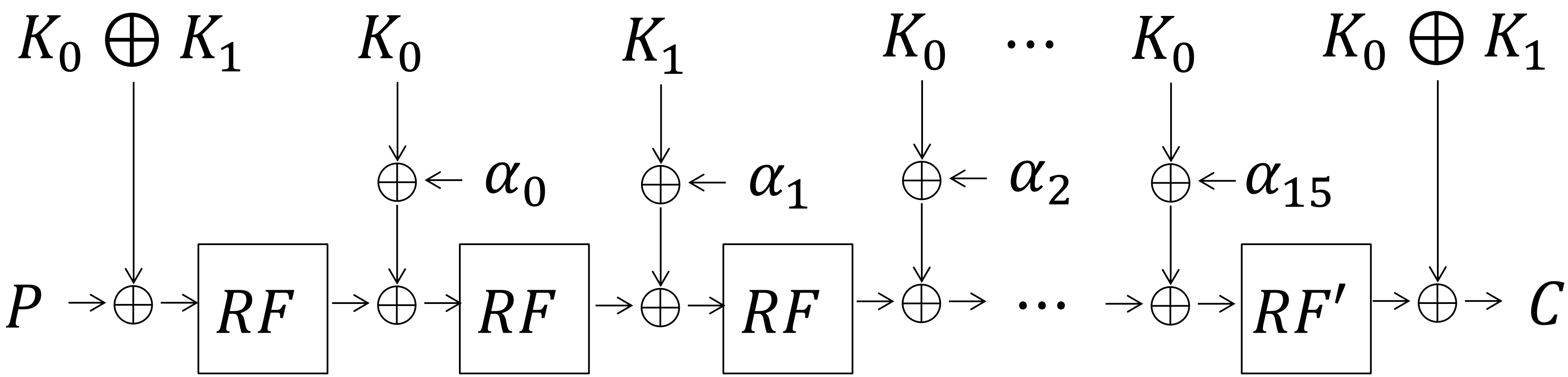
Midori64: Overall Structure

State

1 cell = 4 bits

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Master key: K_0, K_1



$$RF = MixColumn \circ ShuffuleCell \circ SubCell(state)$$

$$RF' = SubCell(state)$$

Midori64: Round Function

SubCell (4-bit involution S-box)

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathrm{Sb}_0[x]$	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

[ePrint2015/1142, Table 4]

ShuffleCell (cell permutation)

$$\begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix} \longrightarrow \begin{bmatrix} s_0 & s_{14} & s_9 & s_7 \\ s_{10} & s_4 & s_3 & s_{13} \\ s_5 & s_{11} & s_{12} & s_2 \\ s_{15} & s_1 & s_6 & s_8 \end{bmatrix}$$

MixColumn (multiplication with non-MDS binary matrix)

$$\begin{pmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{pmatrix}, \quad \text{for } i \in \{0, 4, 8, 12\}$$

Midori64: Round Constant

All cells in all round constants, α_i, are binary.

0	0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1	1	0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0	2	1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1	3	0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1	4	0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1	5	1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0	6	0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0
7	0 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0	8	1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1	9	0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0	10	0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1	11	0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0	12	0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0	13	1 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0
14	1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0	15	0 1 1 0 1 1 0 0 1 0 0 0 1 0 0 1	16	0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0	17	0 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0	18	0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0				

[ePrint2015/1142, Table 5]

Invariant Subspace Attack on Midori64

Analysis on Key Generation

Whitening key: $K_0 \oplus K_1$

Round key: $K_{i \bmod 2} \oplus \alpha_i$ for $i = 0,1, \ldots, 14$

Subspace $\mathcal{K} \triangleq \{0,1\}^{16} \triangleq$

0/1	0/1	0/1	0/1
0/1	0/1	0/1	0/1
0/1	0/1	0/1	0/1
0/1	0/1	0/1	0/1

As explained before, all α_i belong to $\mathcal{K}$.

When K_0 and K_1 belongs to $\mathcal{K}$, all round keys belong to $\mathcal{K}$. (There are 2^{32} such keys.)

Analysis on Sbox

AddKey operation XORs 0/1 to each state nibble.

SubCell :

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathrm{Sb}_0[x]$	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

$$Sb_0(8) = 8 \text{ and } Sb_0(9) = 9$$

Subspace $\mathcal{S} \triangleq \{0,1\}^{16} + 8 \triangleq$

8/9	8/9	8/9	8/9
8/9	8/9	8/9	8/9
8/9	8/9	8/9	8/9
8/9	8/9	8/9	8/9

Key $\in \mathcal{K}$

$$\text{State} \in \mathcal{S} \longrightarrow \oplus \longrightarrow \mathcal{S} \xrightarrow{SubCell} \mathcal{S} \xrightarrow{ShuffleCell} \mathcal{S}$$

Analysis on MixColumn

$$MixColumn: \begin{pmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{pmatrix}, \text{ for } i \in \{0, 4, 8, 12\}$$

Each nibble becomes XOR of 3 elements in state:

$$(\{0,1\} \oplus 8) \oplus (\{0,1\} \oplus 8) \oplus (\{0,1\} \oplus 8)$$

$$= \{0,1\} \oplus 8$$

For any number of rounds, state belongs to $\mathcal{S}$.

$$\mathcal{S} \xrightarrow{MixColumn} \mathcal{S}$$

Weak-key distinguisher

- #weak keys = 2^{32}.
- Distinguisher with 1 CP query.

Key Recovery

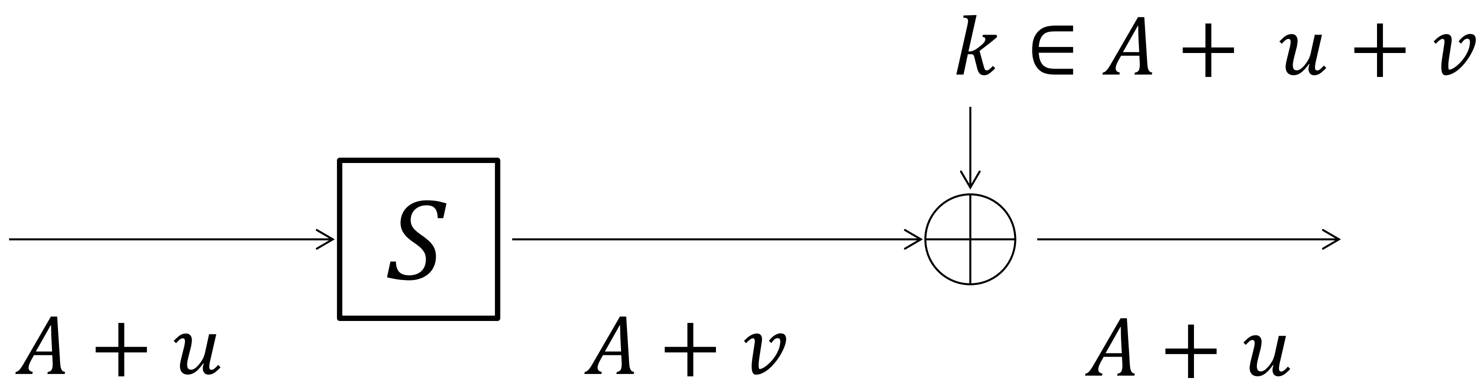
$$\mathrm{Sb}_0(8) = 8 \text{ and } \mathrm{Sb}_0(9) = 9$$

- When state belongs to $\mathcal{S}$, $SubCell$ is equivalent to identity mapping.
- The entire encryption algorithm falls into a linear transformation.
- With 1 (P, C) pair, key space is reduced to 2^{16}. Brute force search with another pair. Complexity is $16^3+2^{16} \approx 2^{16}$.

Extension to Find Weaker Constant

Searching for Weaker Constant

The essence is the following property of S-box.



With the choice of Midori64's round constant,
$A = \{0,1\}, u = v = 8.$

What occurs when the constant is changed?

Search Results

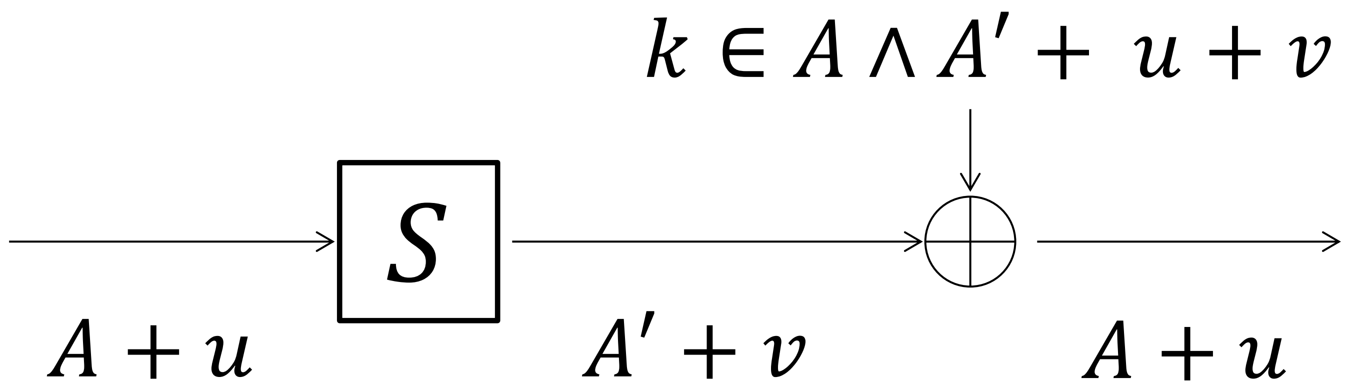
u	v	A	u	v	A
0	c	0 c	3	3	0 4
1	a	0 b	4	e	0 a
1	a	0 2 9 b	5	b	0 e
2	d	0 f	5	b	0 2 c e
2	d	0 5 a f	6	f	0 9
3	3	0 b	7	7	0 f
3	3	0 a	7	7	0 e
3	3	0 7 a d	8	8	0 1

For example, if RoundConstant $\subseteq A \oplus 2 \oplus d = A$, the weak-key class will be bigger (2^{64}).

NTT

Further Extension

The search space can be enlarged to $A + u$ and $A' + v$ as shown below, as long A and A' has non-empty intersection.



Results:

u	A	v	A'
c	0 1 2 3	0	0 2 4 6
0	0 5 a f	1	0 7 a d

Feedback to S-box Design

Motivation of Considering S-box Design

- S-box analysis is the essence of the invariant subspace.
- Is it possible to choose S-box such that the power of invariant subspace can be upper-bounded with any choice of constant?
- Such S-boxes reduce the designer's work load.
- (standard S-box criteria should not be compromised such as maxDP=2^{-2}, maxLP=2^{-2}.)

Search Setting

Assumptions: 4-bit S-box,
good maxDP, maxLP (variants of golden S-boxes),
weakest linear layer (never changes affine subspace).

Choices of S-boxes

- involution or non-involution

Maximal effect of invariant subspace attack

- Up to dim=1 or even avoiding dim=1

Key schedule

- Identical subkey or independent subkeys

Search Results

Involution S-box

	Up to dim=1	Avoiding dim=1
Identical K	✓	–
Independent K	–	–

Non-involution S-box

	Up to dim=1	Avoiding dim=1
Identical K	✓	✓
Independent K	✓	–

Difficulties for Involution S-box

For any x such that $S(x) \neq x$, affine subspace transition of $A \oplus u \overset{S}{\leftrightarrow} A \oplus U$ always exits because of $S\big(S(x)\big) = x$.

Then $\{0, x \oplus S(x)\} + x$ is mapped to itself by applying S or S^{-1}.

This shows impossibility of avoiding dim=1 with an involution S-box for any constant.

Approach for Non-involution S-box

1. First we ensure up to dim=1 by checking:
   - no affine subspace of dim=3 or more.
   - no affine subspace of dim=2 that can be connected (output subspace of one coincides with input subspace of another).

2. To resist dim=1 for identical subkeys, we avoid iterative affine transformations, which can be ensured by making diagonal entries of DDT zero.

Concluding Remarks

Concluding Remarks

Invariant subspace attack on Midori64.

- Weak-key attack for 2^{32} weak keys.
- Distinguisher with 1 CP, key recovery with 2^{16}.

S-box search to prevent invariant subspace

- Involution S-box avoiding weak constant for identical subkeys
- Non-involution S-box avoiding weak constant for any key schedule
- Non-involution S-box avoiding invariant subspace attack for identical subkeys.

Thank you for your attention !!