troduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations

# CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks

#### Christof Beierle Gregor Leander Amir Moradi Shahram Rasoolzadeh

SnT, University of Luxembourg

Horst Görtz Institute for IT Security, Ruhr University Bochum

FSE 2019 March 25, 2019

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction ●○○○	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Physica	Attacks			

• Secrets stored in/processed by an implementation of a primitive can be recovered by **Physical Attacks**.

▲□▶ ▲□▶ ▲三▶ ▲三▶ - 三 - のへで

Introduction ●○○○	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Physica	l Attacks			

- Secrets stored in/processed by an implementation of a primitive can be recovered by **Physical Attacks**.
- Differential Fault Analysis (DFA) attacks are one of the most powerful class of them.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction ●○○○	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Physica	I Attacks			

- Secrets stored in/processed by an implementation of a primitive can be recovered by **Physical Attacks**.
- Differential Fault Analysis (DFA) attacks are one of the most powerful class of them.

・ロト ・四ト ・ヨト ・ヨト



Introduction ●○○○	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Physica	Attacks			

- Secrets stored in/processed by an implementation of a primitive can be recovered by **Physical Attacks**.
- Differential Fault Analysis (DFA) attacks are one of the most powerful class of them.



Introduction ○●○○	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Impeccable Circuite <sup>1</sup>				

Two general construction for Concurrent Error Detection

#### **Adversary Model**

Univariate(/Multivariate) Model,  $M_t$ : The adversary is able to make at most *t* cells of the entire circuit faulty at only one (/every) clock cycle.

<sup>&</sup>lt;sup>1</sup>Aghaie et. al., Impeccable Circuits. IACR Cryptology ePrint Archive, 2018:203. 🚊 🗠 🔍

	000	0000000	oo	oo	
Impeccable Circuite <sup>1</sup>					

Two general construction for Concurrent Error Detection

#### **Adversary Model**

Univariate(/Multivariate) Model,  $M_t$ : The adversary is able to make at most *t* cells of the entire circuit faulty at only one (/every) clock cycle.

Safe-Error and Stuck-at-0/1 models are not covered.

<sup>&</sup>lt;sup>1</sup>Aghaie et. al., Impeccable Circuits. IACR Cryptology ePrint Archive, 2018:203. 🚊 🗠 ໑....

000	000	0000000	00	00	
Impeccable Circuits <sup>1</sup>					

Two general construction for Concurrent Error Detection

#### **Adversary Model**

Univariate(/Multivariate) Model,  $M_t$ : The adversary is able to make at most *t* cells of the entire circuit faulty at only one (/every) clock cycle.

Safe-Error and Stuck-at-0/1 models are not covered.

#### Independence Property

To prevent fault propagation, the coordinate functions of each operation have to be implemented independently.

<sup>&</sup>lt;sup>1</sup>Aghaie et. al., Impeccable Circuits. IACR Cryptology ePrint Archive, 2018:203. 🚊 🗠 🔍

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
0000				

## Independence Property

#### Example (Skinny's MixColumn:)



Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
0000				

## Independence Property

### Example (Skinny's MixColumn:)





Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations

## Independence Property

#### Example (Skinny's MixColumn:)







Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Motivat	ion			

・ コット (雪) ( 小田) ( コット 日)

#### Results of Impeccable Circuits

 Different Lightweight Block Ciphers: Skinny, LED, Midori, Present, Gift, Simon.

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Motivati	ion			

#### Results of Impeccable Circuits

- Different Lightweight Block Ciphers: Skinny, LED, Midori, Present, Gift, Simon.
- There is a big gap between the implementation size of unprotected and protected circuits.

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Motivot	ion			

#### Results of Impeccable Circuits

- Different Lightweight Block Ciphers: Skinny, LED, Midori, Present, Gift, Simon.
- There is a big gap between the implementation size of unprotected and protected circuits.

#### Goals

- Protection against DFA Attacks with efficient hardware implementation
- Tweakable and providing decryption with little implementation area overhead
- Using known design methods for easier security analysis

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Motivoti	<u></u>			

#### Results of Impeccable Circuits

- Different Lightweight Block Ciphers: Skinny, LED, Midori, Present, Gift, Simon.
- There is a big gap between the implementation size of unprotected and protected circuits.

#### Goals

- Protection against DFA Attacks with efficient hardware implementation
- Tweakable and providing decryption with little implementation area overhead
- Using known design methods for easier security analysis

• Skinny-like structure with 128-bit key, 64-bit block & tweak

Introduction	CRAFT Specification ●○○	Design Rationale	Security Analysis	Hardware Implementations
Structu	́е			



32 rounds: 31 identical rounds and last linear round

イロト 不得 トイヨト イヨト

3

• Internal state: viewed as  $4 \times 4$  matrix of nibbles

Introduction	CRAFT Specification ●○○	Design Rationale	Security Analysis	Hardware Implementations
Structu	re			



- 32 rounds: 31 identical rounds and last linear round
- Internal state: viewed as 4 × 4 matrix of nibbles



Introduction 0000	CRAFT Specification ○●○	Design Rationale	Security Analysis	Hardware Implementations
	-			



Introduction	CRAFT Specification ○●○	Design Rationale	Security Analysis	Hardware Implementations
D	E			



▲□▶▲□▶▲□▶▲□▶ □ のQ@

• MixColumn(MC):

Involutory binary matrix *M* is multiplied to each column.

Introduction	CRAFT Specification ○●○	Design Rationale	Security Analysis	Hardware Implementations
	the second second			



- MixColumn (MC): Involutory binary matrix *M* is multiplied to each column.
- AddConstants;(ARC;):

4-bit value  $a_i$  and 3-bit  $b_i$  are xored to the 4th & 5th nibbles.

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

Introduction	CRAF ○●○	T Specification	Design Rationale	Security Analysis	Hardware Implementations
	-				



- MixColumn (MC): Involutory binary matrix *M* is multiplied to each column.
- AddConstants; (ARC;):
   4-bit value a; and 3-bit b; are xored to the 4th & 5th nibbles.

▲□▶ ▲□▶ ▲三▶ ▲三▶ - 三 - のへで

• AddTweakey; (ATK;):

Tweakey  $TK_{i \mod 4}$  is xored to the state.

Introduction	CRAFT Specification ○●○	Design Rationale	Security Analysis	Hardware Implementations
	terre d'anne a			



- MixColumn (MC): Involutory binary matrix *M* is multiplied to each column.
- AddConstants; (ARC;):
   4-bit value a; and 3-bit b; are xored to the 4th & 5th nibbles.
- AddTweakey<sub>i</sub> (ATK<sub>i</sub>):
   Tweakey TK<sub>i</sub> mod 4 is xored to the state.
- PermuteNibbles (PN): Involutory permutation *P* is applied on the nibble positions.

Introduction	CRAFT Specification ○●○	Design Rationale	Security Analysis	Hardware Implementations
<b>D</b>				



- MixColumn (MC): Involutory binary matrix *M* is multiplied to each column.
- AddConstants; (ARC;):
   4-bit value a; and 3-bit b; are xored to the 4th & 5th nibbles.
- AddTweakey<sub>i</sub> (ATK<sub>i</sub>):
   Tweakey TK<sub>i</sub> mod 4 is xored to the state.
- PermuteNibbles (PN): Involutory permutation *P* is applied on the nibble positions.
- SubBox(SB):

4-bit involutory Sbox S is applied to each nibble.



## If $(K_0, K_1)$ are two 64-bit halves of the key and T is the tweak, then

$$TK_0 = K_0 \oplus T$$
$$TK_1 = K_1 \oplus T$$

▲口▶▲圖▶▲臣▶▲臣▶ 臣 のへの

Introduction	CRAFT Specification ○○●	Design Rationale	Security Analysis	Hardware Implementations
Tweake	v Schedule			

If  $(K_0, K_1)$  are two 64-bit halves of the key and *T* is the tweak, then

$$TK_0 = K_0 \oplus T$$
  

$$TK_1 = K_1 \oplus T$$
  

$$TK_2 = K_0 \oplus Q(T)$$
  

$$TK_3 = K_1 \oplus Q(T)$$

where Q is a circular permutation on the position of tweak nibbles:

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

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

Introduction	CRAFT Specification	Design Rationale ●000000	Security Analysis	Hardware Implementations

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		• <b>000</b> 000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

 $SB \circ PN = PN \circ SB$ 

$$\label{eq:mc_arc} \begin{split} \text{MC} \circ \text{Arc} \circ \text{Arc} & \circ \text{Arc} \circ \text{Mc} \\ \text{TK}' & = \text{MC}(\text{TK}) \end{split}$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		000000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

$$\begin{array}{l} \mathcal{DEC} \begin{array}{l} {}^{RC_0, \cdots, RC_{31}}_{TK_0, \cdots, TK_3} = \\ = \left( \texttt{ATK}_3 \circ \texttt{ARC}_{31} \circ \texttt{MC} \circ \texttt{SB} \circ \texttt{PN} \circ \texttt{ATK}_2 \circ \texttt{ARC}_{30} \circ \texttt{MC} \circ \cdots \circ \\ \circ \texttt{SB} \circ \texttt{PN} \circ \texttt{ATK}_0 \circ \texttt{ARC}_0 \circ \texttt{MC} \right)^{-1} \end{array}$$

・ロト ・聞ト ・ヨト ・ヨト 三日

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		000000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

$$\mathcal{DEC} \begin{array}{l} \mathcal{RC}_{0}, \cdots, \mathcal{RC}_{31} = \\ = (\mathsf{ATK}_3 \circ \mathsf{ARC}_{31} \circ \mathsf{MC} \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_2 \circ \mathsf{ARC}_{30} \circ \mathsf{MC} \circ \cdots \circ \\ \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_0 \circ \mathsf{ARC}_0 \circ \mathsf{MC})^{-1} \\ = \mathsf{MC} \circ \mathsf{ARC}_0 \circ \mathsf{ATK}_0 \circ \mathsf{PN} \circ \mathsf{SB} \circ \cdots \circ \\ \circ \mathsf{MC} \circ \mathsf{ARC}_{30} \circ \mathsf{ATK}_2 \circ \mathsf{PN} \circ \mathsf{SB} \circ \mathsf{MC} \circ \mathsf{ARC}_{31} \circ \mathsf{ATK}_3 \end{array}$$

・ロト ・聞ト ・ヨト ・ヨト 三日

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		000000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

$$\mathcal{DEC} \begin{array}{l} \mathcal{RC}_{0}, \cdots, \mathcal{RC}_{31} = \\ = (\mathsf{ATK}_3 \circ \mathsf{ARC}_{31} \circ \mathsf{MC} \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_2 \circ \mathsf{ARC}_{30} \circ \mathsf{MC} \circ \cdots \circ \\ \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_0 \circ \mathsf{ARC}_0 \circ \mathsf{MC})^{-1} \\ = \mathsf{MC} \circ \mathsf{ARC}_0 \circ \mathsf{ATK}_0 \circ \mathsf{PN} \circ \mathsf{SB} \circ \cdots \circ \\ \circ \mathsf{MC} \circ \mathsf{ARC}_{30} \circ \mathsf{ATK}_2 \circ \mathsf{PN} \circ \mathsf{SB} \circ \mathsf{MC} \circ \mathsf{ARC}_{31} \circ \mathsf{ATK}_3 \end{array}$$

・ロト ・聞ト ・ヨト ・ヨト 三日

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		000000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
		0000000		

#### Lemma 1:

CRAFT decryption is the same as its encryption with modified tweakeys and reverse order of round constants.

$$\mathcal{DEC} \begin{array}{l} \mathcal{RC}_{0}, \cdots, \mathcal{RC}_{31} = \\ = (\mathsf{ATK}_3 \circ \mathsf{ARC}_{31} \circ \mathsf{MC} \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_2 \circ \mathsf{ARC}_{30} \circ \mathsf{MC} \circ \cdots \circ \\ \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_0 \circ \mathsf{ARC}_0 \circ \mathsf{MC})^{-1} \\ = \mathsf{MC} \circ \mathsf{ARC}_0 \circ \mathsf{ATK}_0 \circ \mathsf{PN} \circ \mathsf{SB} \circ \cdots \circ \\ \circ \mathsf{MC} \circ \mathsf{ARC}_{30} \circ \mathsf{ATK}_2 \circ \mathsf{PN} \circ \mathsf{SB} \circ \mathsf{MC} \circ \mathsf{ARC}_{31} \circ \mathsf{ATK}_3 \\ = \mathsf{ATK}_0' \circ \mathsf{ARC}_0 \circ \mathsf{MC} \circ \mathsf{SB} \circ \mathsf{PN} \circ \cdots \circ \\ \circ \mathsf{ATK}_2' \circ \mathsf{ARC}_{30} \circ \mathsf{MC} \circ \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{ATK}_3' \circ \mathsf{ARC}_{31} \circ \mathsf{MC} \\ = \mathcal{ENC} \begin{array}{c} \mathcal{RC}_{31}, \cdots, \mathcal{RC}_0 \\ \mathcal{TK}_3', \cdots, \mathcal{TK}_0' \end{array} \right\}$$

Introduction	CRAFT Specification	Design Rationale ○○○○●○○	Security Analysis	Hardware Implementations
Sbox				

▲□▶ ▲□▶ ▲三▶ ▲三▶ ▲□▶

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Sbox				

▲□▶ ▲□▶ ▲□▶ ▲□▶ = 三 のへで

#### Sbox & Redundant Sbox

For each Sbox, we need to implement a Redundant Sbox.

Introduction	CRAFT Specification	Design Rationale ○○○○●○○	Security Analysis	Hardware Implementations
Sbox				

#### Sbox & Redundant Sbox

For each Sbox, we need to implement a *Redundant Sbox*. For example, in case of 4-bit redundancy:

$$S_4 = F_4 \circ S \circ F_4^{-1}$$

where  $F_4$  is a multiplication with

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Sbox				

#### Sbox & Redundant Sbox

For each Sbox, we need to implement a *Redundant Sbox*. For example, in case of 4-bit redundancy:

$$S_4 = F_4 \circ S \circ F_4^{-1}$$

where  $F_4$  is a multiplication with

#### Problem

There are 46 206 736 involutory 4-bit Sboxes which implementing and synthesizing all of them is impossible.

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Sbox				

• Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.

・ コット (雪) ( 小田) ( コット 日)

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Shov				

- Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.
- Implementation cost of each operation is sum of area size for its coordinate functions.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Sbox				

- Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.
- Implementation cost of each operation is sum of area size for its coordinate functions.
- For each Sbox, size of 13 Boolean functions are important.

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Sbox				

- Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.
- Implementation cost of each operation is sum of area size for its coordinate functions.
- For each Sbox, size of 13 Boolean functions are important.

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

• There are 12 870 four-bit balanced Boolean functions.

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Shox				

- Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.
- Implementation cost of each operation is sum of area size for its coordinate functions.
- For each Sbox, size of 13 Boolean functions are important.

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

- There are 12 870 four-bit balanced Boolean functions.
- Up to bit permutation-equivalence, there are only 730.

Introduction	CRAFT Specification	Design Rationale ○○○○○●○	Security Analysis	Hardware Implementations
Shox				

- Because of *Independence Property*, each coordinate function of Sbox must be implemented separately.
- Implementation cost of each operation is sum of area size for its coordinate functions.
- For each Sbox, size of 13 Boolean functions are important.
- There are 12 870 four-bit balanced Boolean functions.
- Up to bit permutation-equivalence, there are only 730.

#### **Results for Sbox**

Among all the smallest found Soxes, we use the Midori's one.

Introduction	CRAFT Specification	Design Rationale ○○○○○●	Security Analysis	Hardware Implementations
There is a				

#### Key Schedule

Round key updating method needs at least 128 registers.

▲□▶ ▲□▶ ▲三▶ ▲三▶ - 三 - のへで

• Round key alternating method needs 64 multiplexers.

Introduction	CRAFT Specification	Design Rationale ○○○○○●	Security Analysis	Hardware Implementations
-				

#### Key Schedule

- Round key updating method needs at least 128 registers.
- Round key alternating method needs 64 multiplexers.

#### **Tweak Schedule**

• Xoring the tweak with key.

Introduction	CRAFT Specification	Design Rationale ○○○○○●	Security Analysis	Hardware Implementations
-				

#### Key Schedule

- Round key updating method needs at least 128 registers.
- Round key alternating method needs 64 multiplexers.

#### Tweak Schedule

- Xoring the tweak with key.
- To prevent Time-Data-Memory Trade-off attacks, tweak cannot be always the same when round keys are equal.

Introduction	CRAFT Specification	Design Rationale ○○○○○●	Security Analysis	Hardware Implementations
-				

#### Key Schedule

- Round key updating method needs at least 128 registers.
- Round key alternating method needs 64 multiplexers.

#### **Tweak Schedule**

- Xoring the tweak with key.
- To prevent Time-Data-Memory Trade-off attacks, tweak cannot be always the same when round keys are equal.
- Solution: using 64 multiplexers to choose T or a nibble-wise permutation of it, Q(T).

Introduction	CRAFT Specification	Design Rationale ○○○○○●	Security Analysis	Hardware Implementations
-				

#### Key Schedule

- Round key updating method needs at least 128 registers.
- Round key alternating method needs 64 multiplexers.

#### **Tweak Schedule**

- Xoring the tweak with key.
- To prevent Time-Data-Memory Trade-off attacks, tweak cannot be always the same when round keys are equal.
- Solution: using 64 multiplexers to choose T or a nibble-wise permutation of it, Q(T).
- To provide maximum possible security against TDM-TO attack, *Q* must be circular (there are  $15! \approx 2^{40}$ ).
- Trying 1000 of them, *Q* is the one with most active Sboxes in related-tweak differential attack.

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Security	/			

#### Security Analysis

- Time-Data-Memory Trade-off
- (Truncated / Impossible) (ST/RT) Differential
- (Linear Hulls / Zero-Correlation) Linear
- Integral
- Meet in the Middle
- (Linear Subspace/Nonlinear) Invariant Attacks

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
Security	1			

#### Security Analysis

- Time-Data-Memory Trade-off
- (Truncated / Impossible) (ST/RT) Differential
- (Linear Hulls / Zero-Correlation) Linear
- Integral
- Meet in the Middle
- (Linear Subspace/Nonlinear) Invariant Attacks

#### Security Claim

- 124 bit security in the related-tweak model
- No claim in chosen-key, known-key or related-key models

Introduction	CRAFT Specification	Design Rationale	Security Analysis ○●	Hardware Implementations

#### Accelerated Exhaustive Search

#### Related-key Property:

If  $\Delta = (x, x, ..., x)$ , since  $Q(\Delta) = \Delta$ , both  $(K_0, K_1, T)$  and  $(K_0 + \Delta, K_1 + \Delta, T + \Delta)$  cause the same tweakeys:

 $TK_i = TK'_i \quad (0 \le i \le 3)$ 

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

Introduction	CRAFT Specification	Design Rationale	Security Analysis ○●	Hardware Implementations

#### Accelerated Exhaustive Search

#### Related-key Property:

If  $\Delta = (x, x, ..., x)$ , since  $Q(\Delta) = \Delta$ , both  $(K_0, K_1, T)$  and  $(K_0 + \Delta, K_1 + \Delta, T + \Delta)$  cause the same tweakeys:

 $TK_i = TK'_i \quad (0 \le i \le 3)$ 

#### Attack Procedure:

- Attacker asks for encryption of the same plaintext *P* under 16 different tweaks of  $T, T + \Delta_1, \ldots, T + \Delta_{15}$ :  $C_0, C_1, \ldots, C_{15}$ .
- By setting one of the key nibbles to zero, for each of 2<sup>124</sup> possible key candidate (K<sup>\*</sup><sub>0</sub>, K<sup>\*</sup><sub>1</sub>), he computes C<sup>\*</sup>, the encryption of P using K<sup>\*</sup><sub>0</sub>, K<sup>\*</sup><sub>1</sub> and T.
- If C<sup>\*</sup> is equal to C<sub>x</sub>, then (K<sup>\*</sup><sub>0</sub> + Δ<sub>x</sub>, K<sup>\*</sup><sub>1</sub> + Δ<sub>x</sub>) is a candidate for the master key.

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations ●○

## Hardware Implementations



Area (GE) Comparison of Round-based Implementation using IBM 130nm ASIC Library

◆□▶ ◆□▶ ◆三▶ ◆三▶ ・三 の々で

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations ○●
Summa	ry			

## CRAFT:

#### Implementation

- A lightweight tweakable block cipher with effiCient pRotection Against DFA aTtacks
- The smallest block cipher with 128-bit key in the round-based implementation (950 GE)
- Lower area overhead to support a 64-bit tweak (245 GE)

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

Lower area overhead to support decryption (140 GE)

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations ○●
Summai	ſV			

## CRAFT:

#### Implementation

- A lightweight tweakable block cipher with effiCient pRotection Against DFA aTtacks
- The smallest block cipher with 128-bit key in the round-based implementation (950 GE)
- Lower area overhead to support a 64-bit tweak (245 GE)
- Lower area overhead to support decryption (140 GE)

#### Security

Providing 124-bit security in the related-tweak model

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
The End	ł			

## Thank you for your attention.

## Looking forward for further analysis by you



▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations

## Time-Data-Memory Trade-off Attack

- Attacker fixes the tweakeys to  $TK_0 = 0$ ,  $TK_1 = X$ ,  $TK_2 = T'$ and  $TK_3 = X + T'$ .
- For plaintext P and all possible X and T', he computes the ciphertext C<sub>T',X</sub> and saves X in the index (T', C) of table T.
- For all possible tweaks *T*, attacker requests for encryption of *P*; *C*<sub>T</sub>.
- For each of *T*, he gets a candidate for  $K_0 + K_1$  by looking up to the index  $(T + Q(T), C_T)$  of T.
- 2<sup>64+dim{T+Q(T)}</sup> pre-computations, 2<sup>64+dim{T+Q(T)}</sup> memory, 2<sup>65</sup> online computions and 2<sup>64</sup> data.
- All online attack:  $2^{64+\dim\{T+Q(T)\}}$  computations,  $2^{64}$  data and memory.



**CRAFT** Specification Introduction **Design Rationale** Hardware Implementations 15-Round Meet-in-the-Middle Attacks  $K_0$  $K_1$  $\mathcal{R}_0$ MC  $\mathcal{R}_1$ MC PN PN ARC<sub>0</sub> ARC<sub>1</sub> SB SB  $ATK_0$ ATK<sub>1</sub>  $\mathcal{R}_2$  $\mathcal{R}_3$  $\mathcal{R}_4$ MC MC MC PN PN PN ARC<sub>2</sub> ARC<sub>3</sub> ARC4 SB SB SB  $ATK_2$ ATK<sub>3</sub>  $ATK_0$  $\mathcal{R}_5$ MC  $\mathcal{R}_6$ MC ARC<sub>6</sub>  $\mathcal{R}_7$ MC PN PN PN ARC<sub>7</sub> ARC<sub>5</sub> SB SB SB ATK<sub>1</sub> ATK<sub>2</sub> ATK3 matching  $\mathcal{R}_8$ MC  $\mathcal{R}_{9}$  $\mathcal{R}_{10}$ MC MC PN PN SB PN ARC<sub>9</sub> ARC<sub>8</sub> ARC10 SB SB ATKo ATK1 ATK<sub>2</sub>  $\mathcal{R}_{11}$ MC  $\mathcal{R}_{12}$ MC  $\mathcal{R}_{13}$ MC PN PN PN ARC11 ARC12 ARC13 SB SB SB ATK<sub>3</sub> ATK<sub>0</sub> ATK<sub>1</sub>

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations
13-Ro	ound Integral	Distinguishe	r	
			$ \begin{array}{c c} \mathcal{R}_{0} & \text{MC} \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \Lambda & \Lambda & \Lambda & \Lambda \\ \hline \end{array} \\ \begin{array}{c} \mathcal{R}_{0} & \text{MC} \\ \mathcal{R}_{0} & \text{MC} $	
	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	
	R4         MC         PN           A         A         AAC4         SB           A         A         ATK0         A           A         A         A         A	R5         MC         PN           A         ARC5         SB           A         A         A	Re         MC         PN           ARCe         ARCe         SB           ATK2         A         SB	
	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c} C & A & C & C \\ \hline A & A & C & C \\ \hline C & C & A & C \\ \hline A & U & C & A \end{array}$
	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c} \mathcal{R}_{11} & \text{WC} \\ \hline A & A & U & C \\ \hline A & C & C & C \\ C & U & U & A \\ \hline U & A & A \\ \hline A & C & C & C \\ \hline \end{array} \xrightarrow{\text{ATK}_3} \begin{array}{c} U & U & U & U & B \\ \hline U & A & A \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & A \\ \hline C & U & U & A \\ \hline U & A & A \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & A \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & U \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & U \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & U \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & A \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & U & A \\ \hline \end{array} \xrightarrow{\text{VC}} \begin{array}{c} U & A \\ \end{array} \xrightarrow{\text{VC}} \begin{array}{\tilde$	$\begin{array}{c c} \mathcal{R}_{12} & MC \\ \hline \mathbf{A} & \mathbf{U} & \mathbf{A} & \mathbf{A} & ABC_{12} & \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} \\ \hline \mathbf{U} & \mathbf{U} & \mathbf{C} & \mathbf{A} & ATK_0 & \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} \\ \hline \mathbf{A} & \mathbf{A} & \mathbf{U} & \mathbf{A} \\ \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} \\ \hline \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} & \mathbf{U} \\ \hline \end{array} \right)$	

Introduction CRAFT Specification Design Rationale ooooo Security Analysis oo

## Enc. & Dec. Algorithms

**Input** : X: plaintext  $K_0||K_1: cipher key$ T: tweak **Output**: *Y*: *ciphertext*  $TK_0 \leftarrow K_0 \oplus T$  $TK_1 \leftarrow K_1 \oplus T$  $TK_2 \leftarrow K_0 \oplus Q(T)$  $TK_3 \leftarrow K_1 \oplus Q(T)$  $Y \leftarrow X$ for  $i \leftarrow 0$  to 31 do  $Y \leftarrow MC(Y)$  $Y_{4.5} \leftarrow Y_{4.5} \oplus RC_i$  $Y \leftarrow Y \oplus TK_{i \mod 4}$ if  $i \neq 31$  then  $\begin{array}{c|c} Y \leftarrow \mathsf{PN}(Y) \\ Y \leftarrow \mathsf{SB}(Y) \end{array}$ end end

**Input** :X: ciphertext  $K_0||K_1: cipher key$ T: tweak **Output**: Y: plaintext  $TK_0 \leftarrow MC(K_0 \oplus T)$  $TK_1 \leftarrow MC(K_1 \oplus T)$  $TK_2 \leftarrow MC(K_0 \oplus Q(T))$  $TK_3 \leftarrow MC(K_1 \oplus Q(T))$  $Y \leftarrow X$ for  $i \leftarrow 31$  to 0 do  $Y \leftarrow MC(Y)$  $Y_{4,5} \leftarrow Y_{4,5} \oplus RC_i$  $Y \leftarrow Y \oplus TK_i \mod 4$ if  $i \neq 0$  then  $\begin{array}{c|c} Y \leftarrow \mathsf{PN}(Y) \\ Y \leftarrow \mathsf{SB}(Y) \end{array}$ end end

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - 釣��

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementations

#### Implementation Results



◆□ > ◆□ > ◆三 > ◆三 > ・三 ・ のへぐ

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementation

## Round-based Implementation with Fault Detection



▲□▶ ▲□▶ ▲ □▶ ▲ □▶ ▲ □ ● ● ● ●

Introduction	CRAFT Specification	Design Rationale	Security Analysis	Hardware Implementation

## Round-based Implementation with Fault Detection



▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ