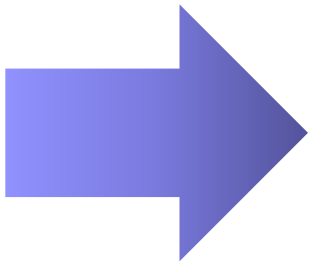
Piccolo: An Ultra-Lightweight Blockcipher

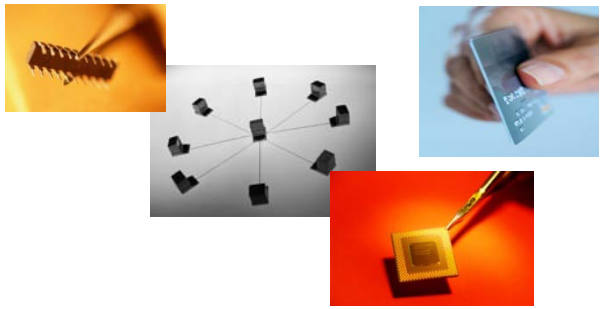
Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita and Taizo Shirai

Sony Corporation

Motivation for New Design

- **Cryptography is required everywhere**
  - RFID, sensor nodes, IoT, low resource devices,..
  - Strong demands for lightweight cryptography





Personal **Pervasive**

- **Our Target**
  - Blockcipher

Bulk encryption, MAC, entity authentication protocol,...

Piccolo is

- Feistel-type lightweight blockcipher that achieves:
  - **High security**
    - Secure against known attacks including MITM and RKA
  - **Compact implementation**
    - Less than 700 GE ⇒ low power consumption
    - Low required GE keeping high throughput
      ⇒ low energy consumption
- **"General purpose"** lightweight blockcipher
  - Not limited to applications
  - Decryption can be supported without much cost
    - Because of involution structure
  - Suitable for both flexible key and fixed key setting
    - Because of permutation based key scheduling

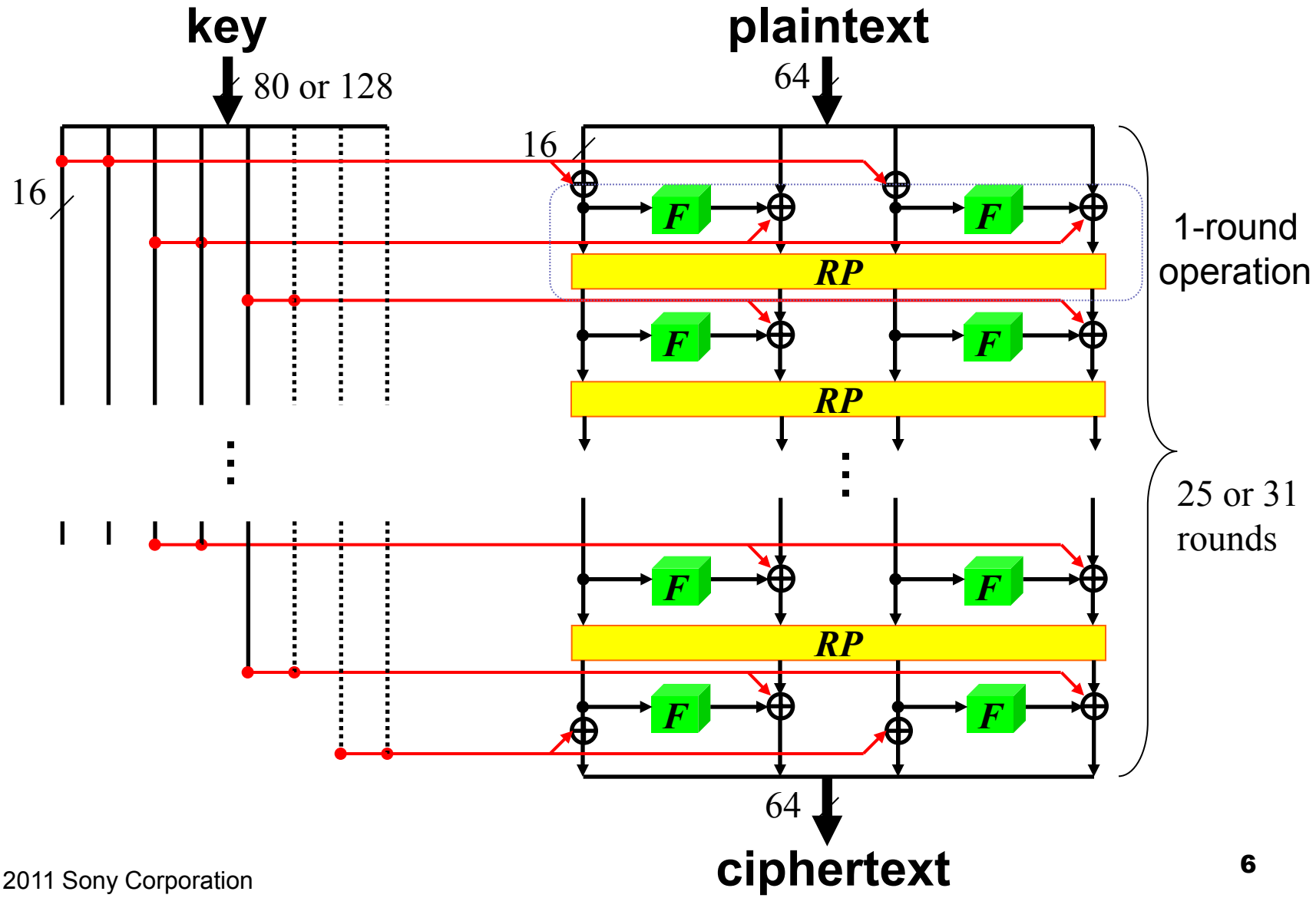
Specification

The Blockcipher Piccolo

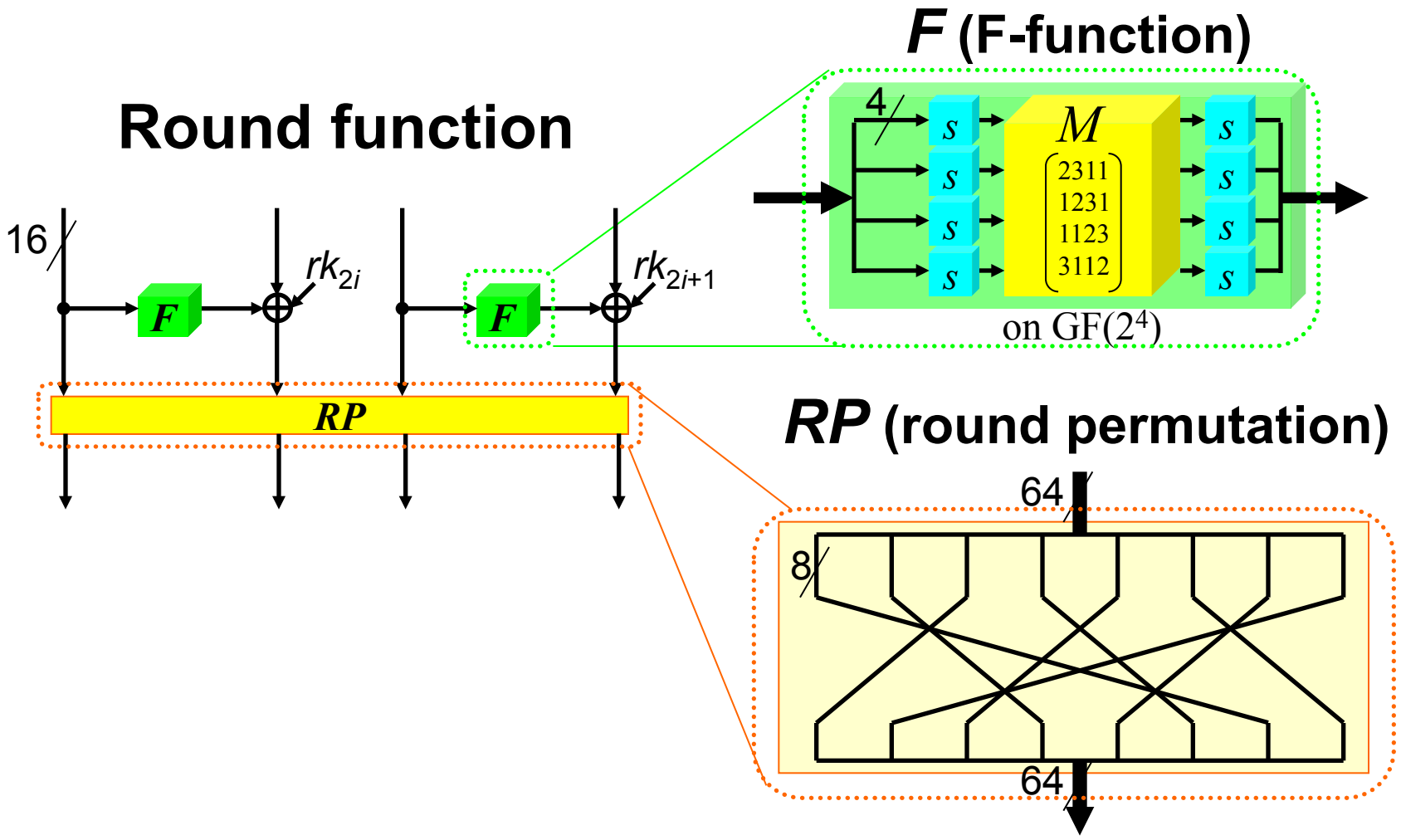
Basic Information

- **Block Size** : **64-bit**
- **Key Size** : **80-bit, 128-bit**
  (referred as **Piccolo-80/128**)
- **Structure** : **variant of 4-line type-II generalized Feistel network**
- **# Rounds** : **25** (80-bit key), **31** (128-bit key)

The Blockcipher Piccolo

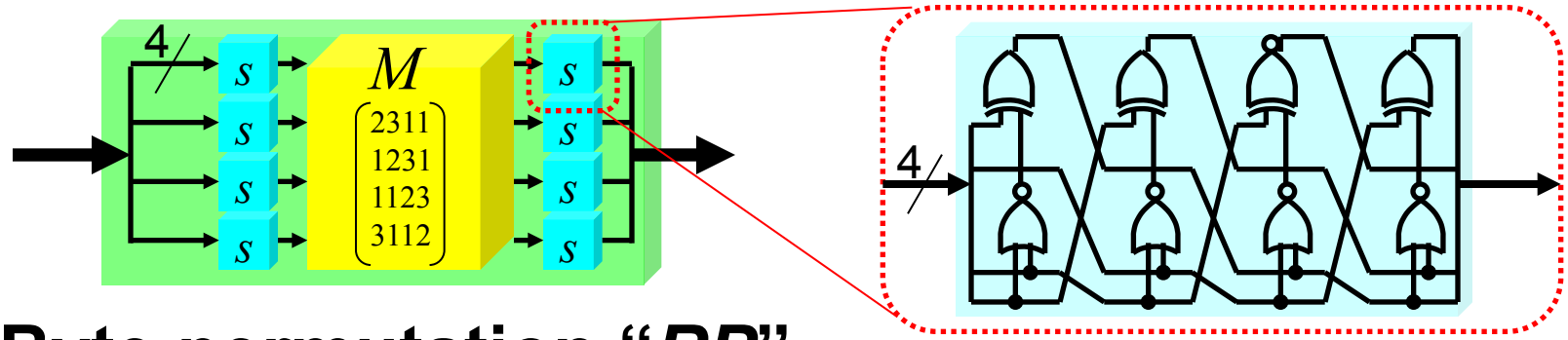


Round function (*F* and *RP*)

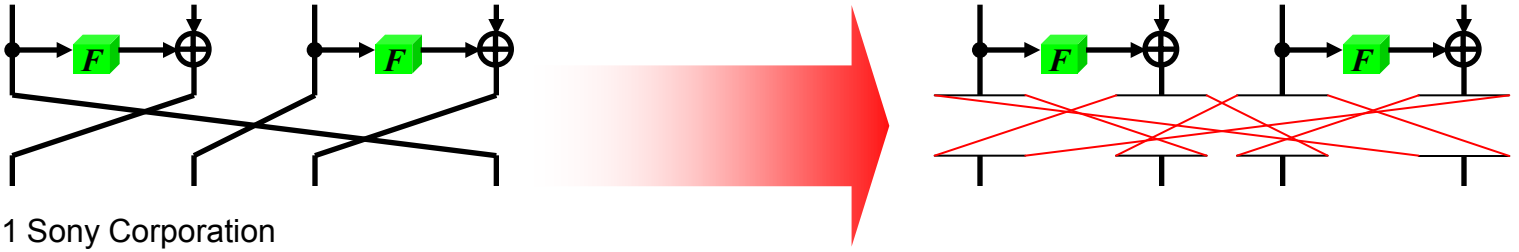


What's new in round function

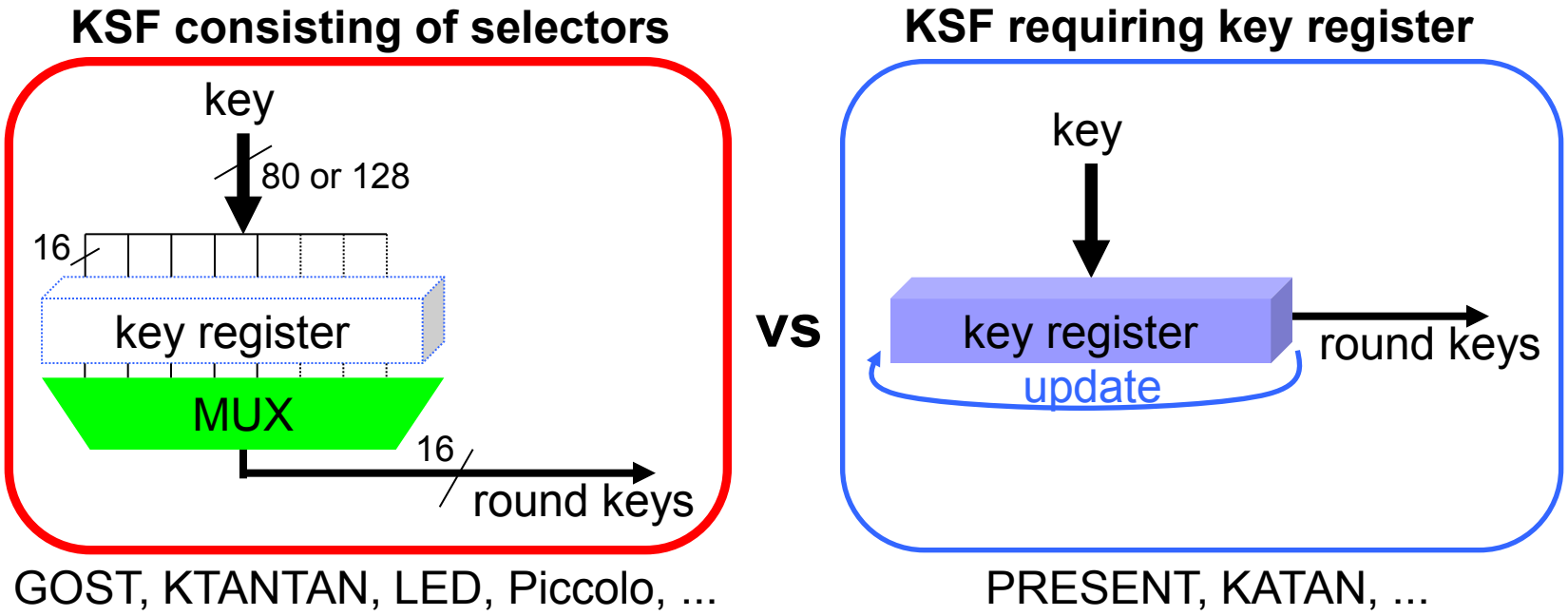
- **Compact 4-bit S-box**
  - only 4 NORs, 3 XORs and 1 XNOR (about 12 GE)
  - sandwich construction makes F-function strong



- **Byte permutation "*RP*"**
  - provides fast diffusion without HW implementation cost
    - enhance security against impossible diff., saturation, MITM, ...



Key scheduling function (KSF)



- **MUX based KSF**
  - Key register is not necessary
  - Suitable for both fixed and flexible key settings
  - Carefully chose the permutation to have enough immunity against RKA and MITM

Security analysis

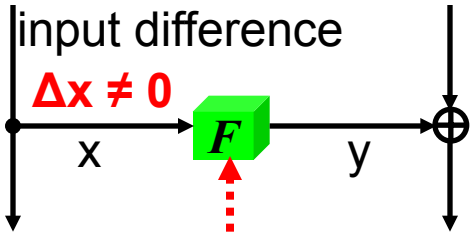
Security analysis

- Active F-functions based evaluation
  - Differential attack
  - Linear attack
  - Boomerang-type attacks
  - Related key differential-type attacks
    - Related key boomerang/rectangle attacks
    - Related key impossible differential attack
- Diffusion property based evaluation
  - Impossible differential attack
  - Saturation attack
  - Meet-in-the-middle attack
- Others
  - Higher-order differential attack
  - Algebraic attack

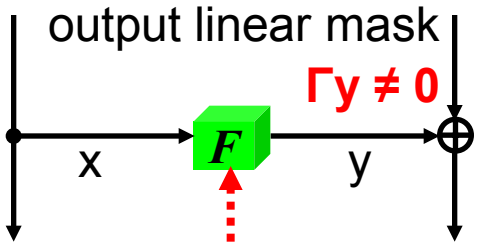
Security analysis

- Active F-functions based evaluation
  - Differential attack
  - Linear attack
  - Boomerang-type attacks
  - Related key differential-type attacks
    - Related key boomerang/rectangle attacks
    - Related key impossible differential attack
- Diffusion property based evaluation
  - Impossible differential attack
  - Saturation attack
  - Meet-in-the-middle attack
- Others
  - Higher-order differential attack
  - Algebraic attack

Active F-function



differentially active F-function

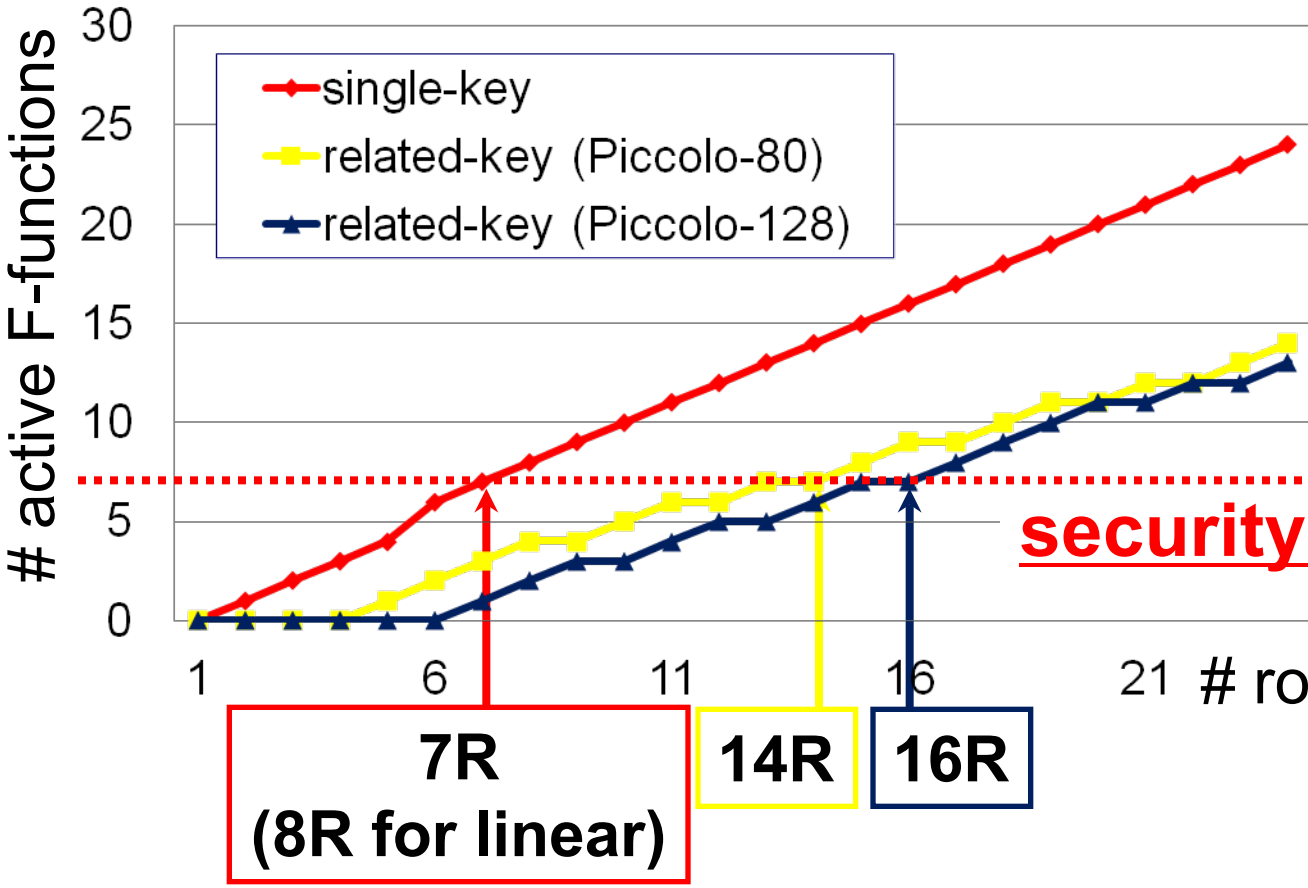


linearly active F-function

- Each differentially/linearly active F-function reduces differential/linear probability
- ⇒ minimum number of active F-function implies the security against differential and linear type attacks

- Counted the number of active F-functions by exhaustively searching all possible differential/linear trails

active F-functions of Piccolo



total # rounds

25 / 31 rounds
(Piccolo-80 / 128)

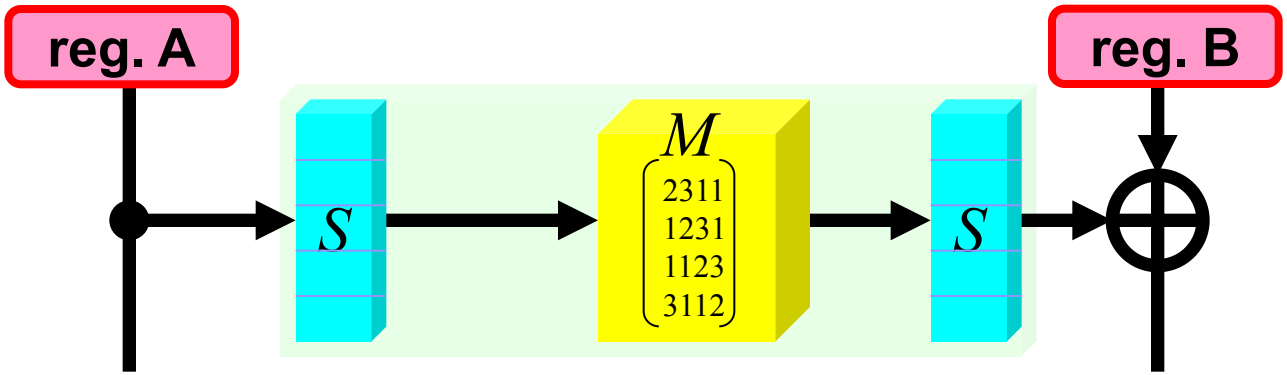
MDP of F = $2^{-9.299}$ (**7** active F-functions needed)

MLP of F = $2^{-8.0}$ (**8** active F-functions needed)

Implementation aspects

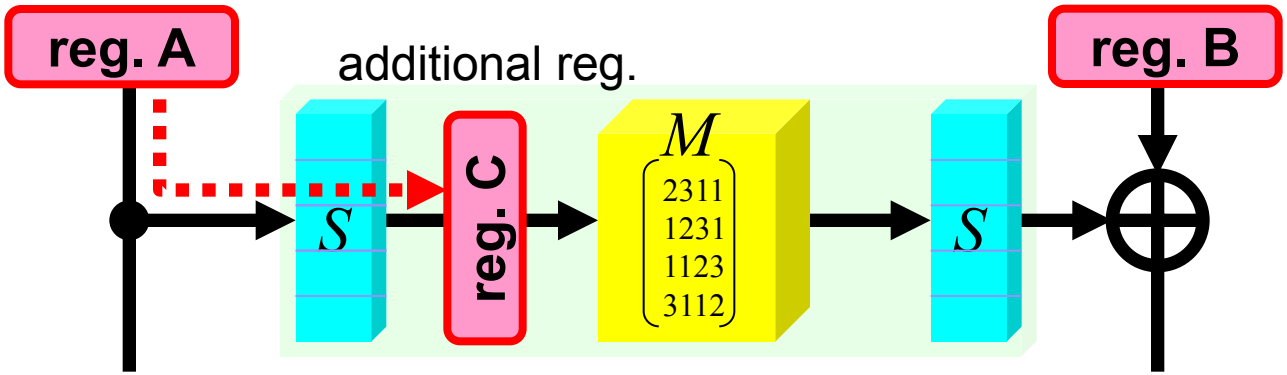
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function



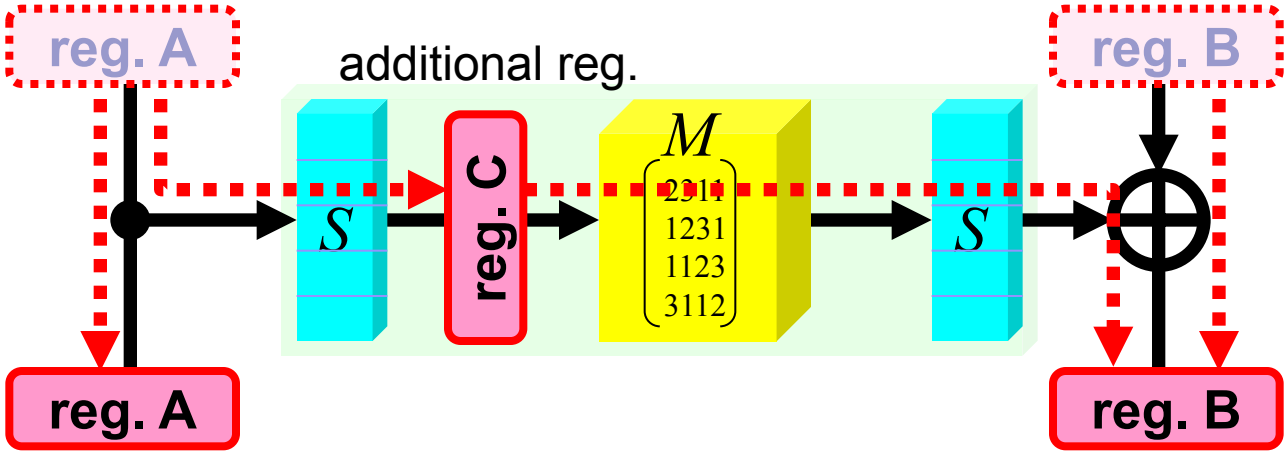
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function



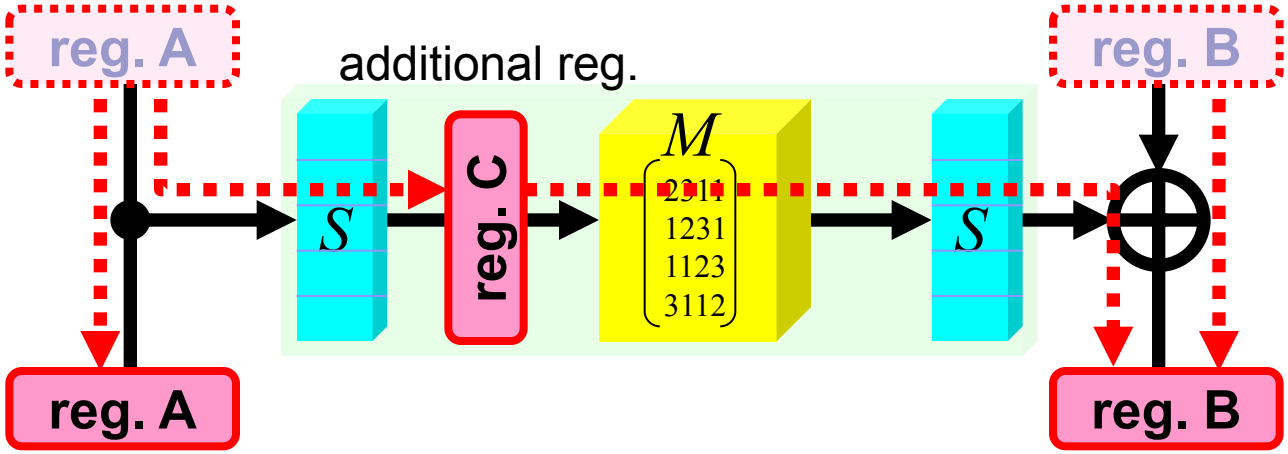
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function



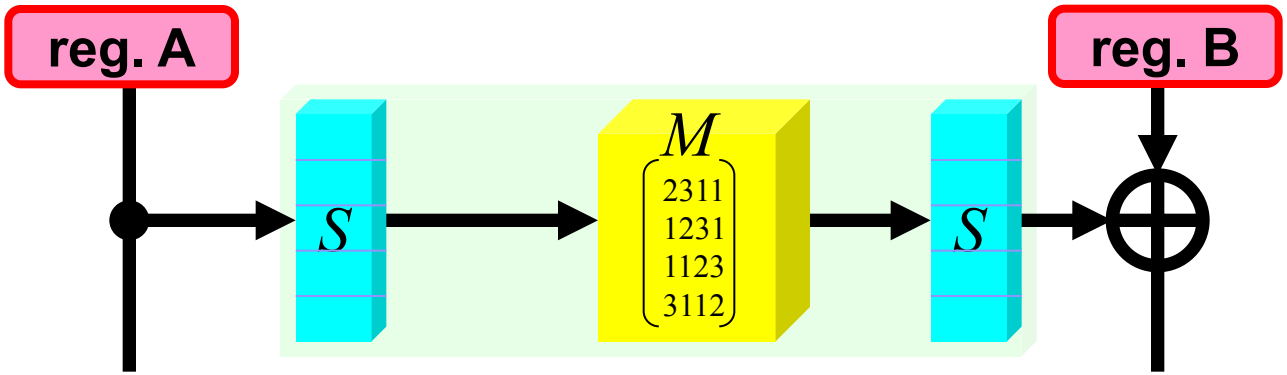
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function



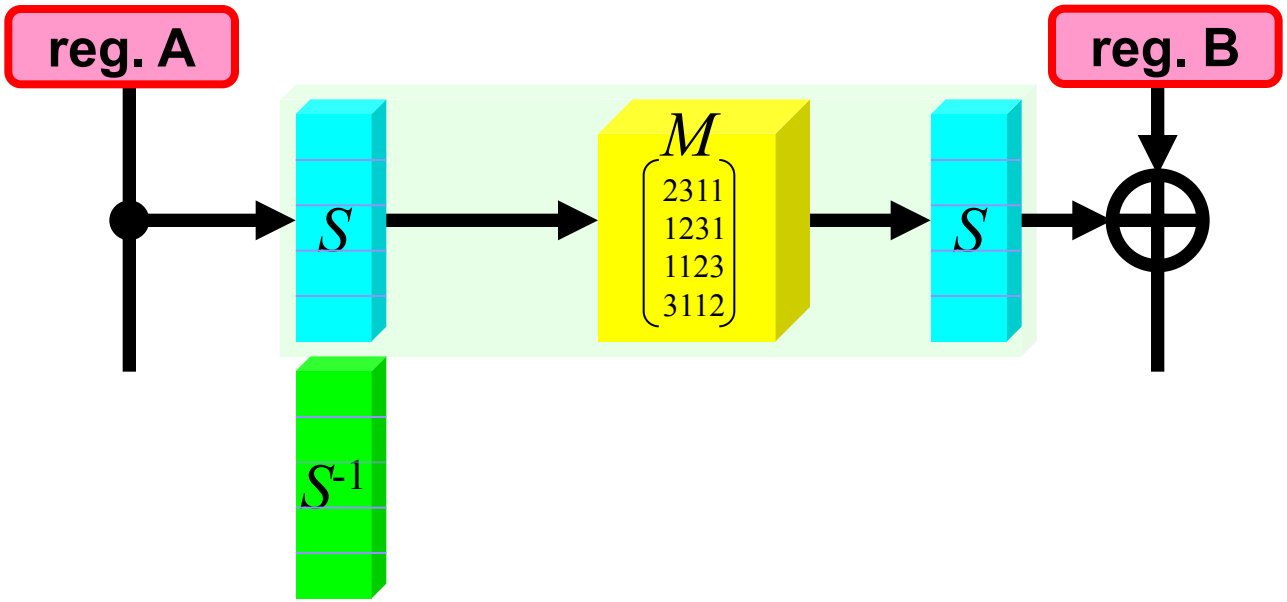
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function



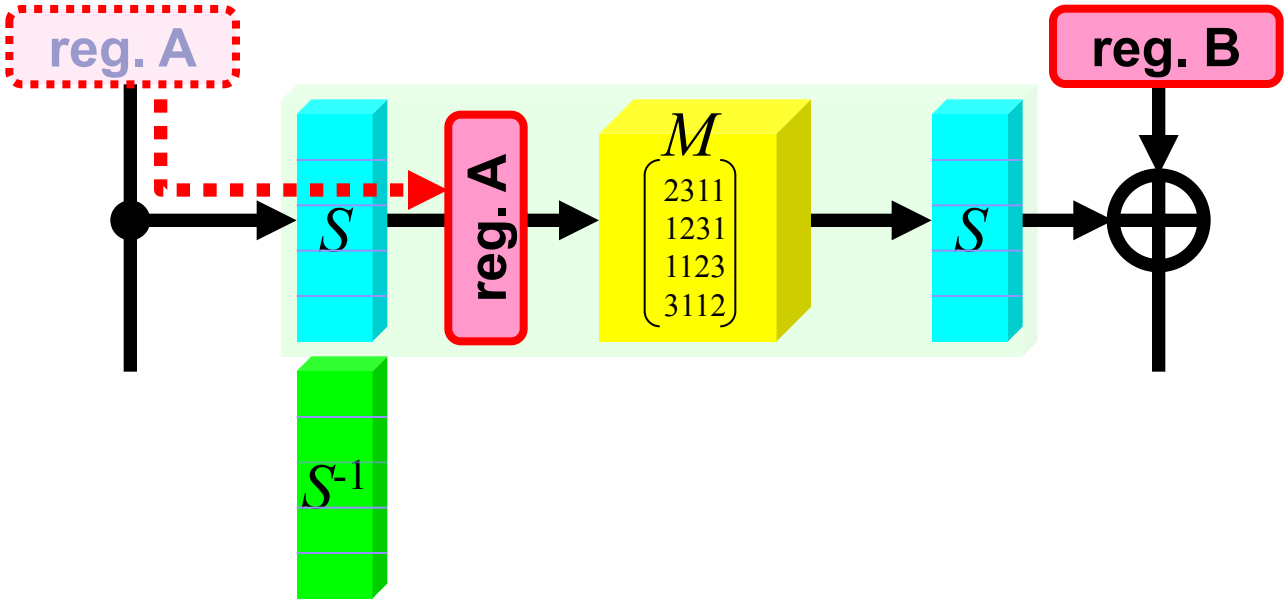
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function



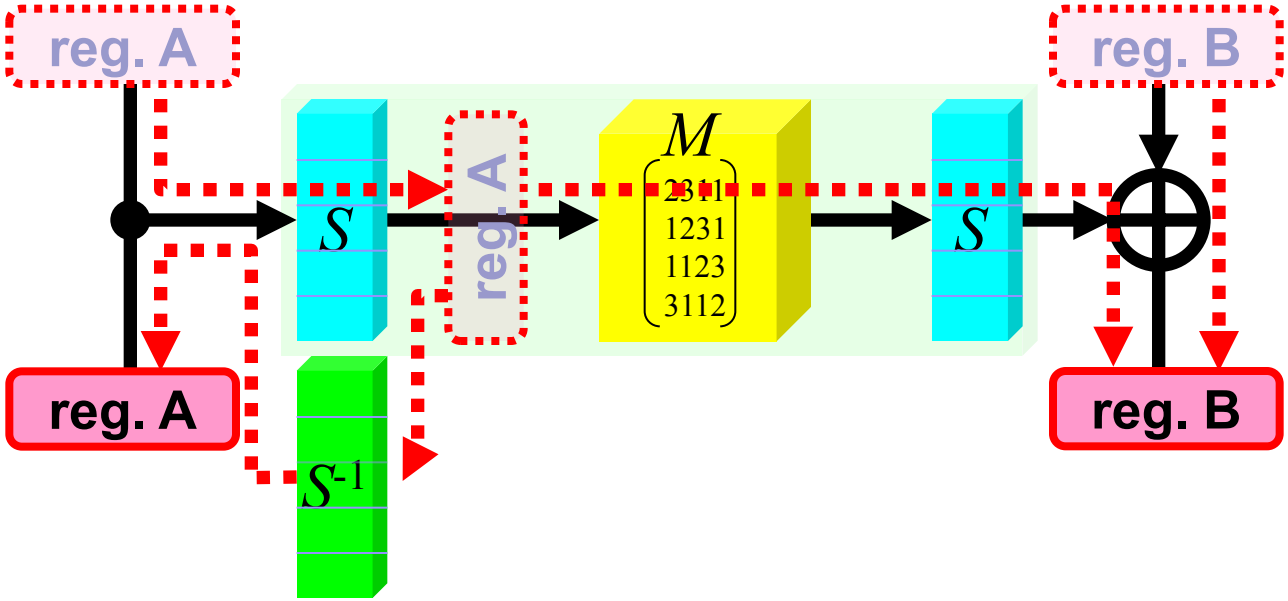
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function



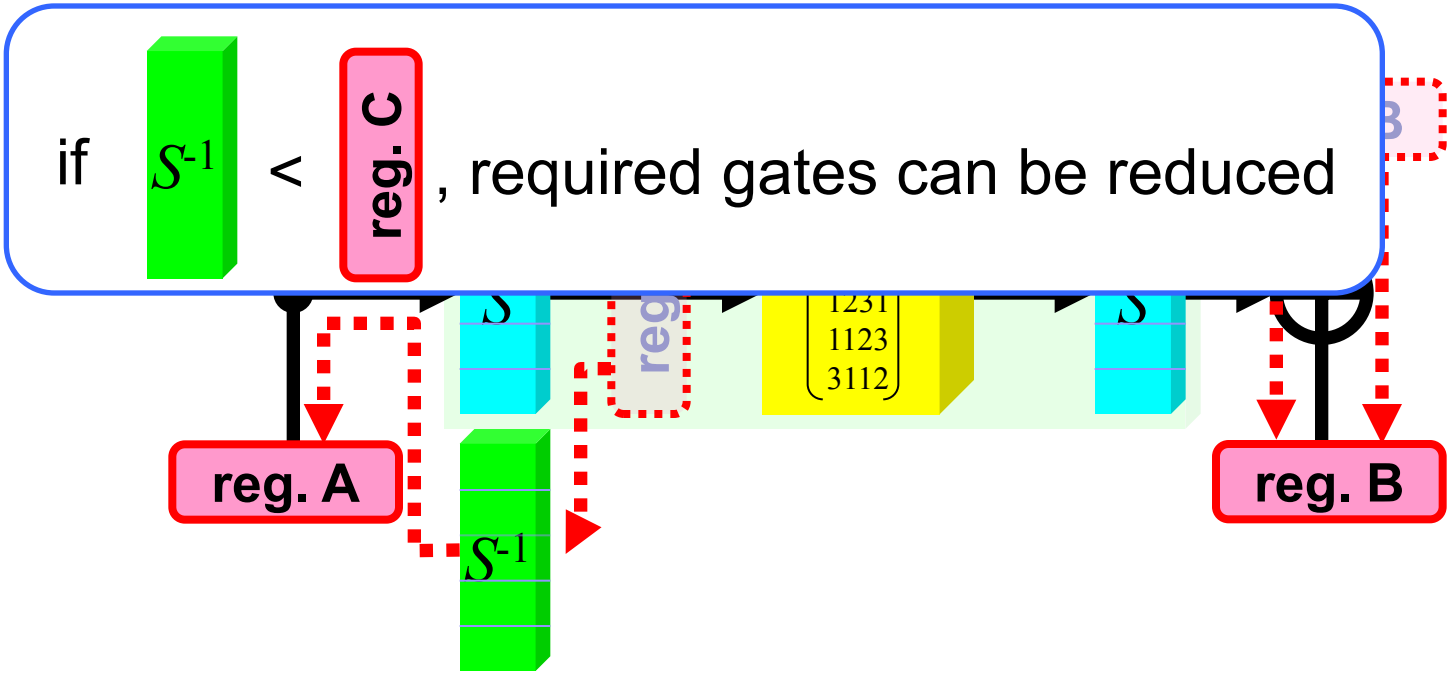
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function



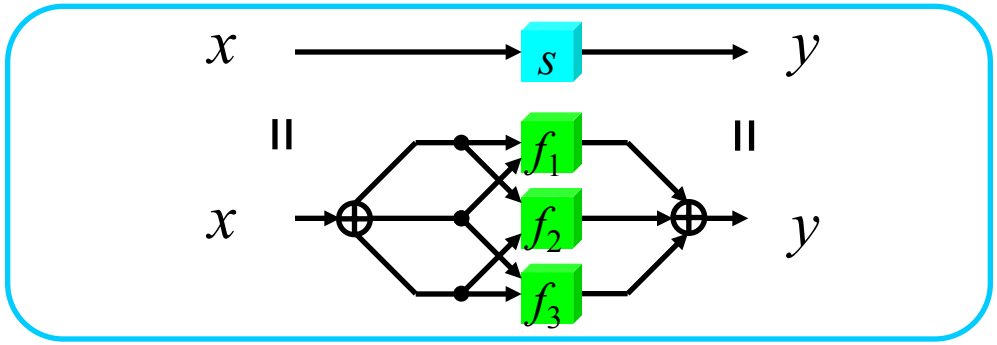
Optimization for F-function
in 4-bit serialized architecture

- Feistel-type requires intermediate registers for F-function
- intermediate registers can be reduced by adding S^{-1} function
- S^{-1} of Piccolo is small ⇒ required gate is reduced



Countermeasure against SCA

- Threshold implementations [ICICS06]
  - provably secure countermeasure against 1st order SCA
  - at least 3 shares are necessary
    (required gates depend on # shares, and 3 is the smallest)



- S-box of Piccolo
  - belongs to the alternating group A_{16}
    ⇒ can be decomposed using quadratic bijections [CHES10]
  - Thus, Piccolo S-box requires **only 3 shares** when applying threshold implementation

Hardware performance (summary)

		serialized arch.		round-based arch.	
algorithm		Piccolo-80	Piccolo-128	Piccolo-80	Piccolo-128
cycles/block		432	528	27	33
fixed key	area	**616**	**654**	1051	1083
	FOM			**2145**	**1653**
flexible key	area	1043	1334	1496	1773
	FOM			1059	616
fixed key	area	676	714	1189	1284
flexible key	area	1103	1394	1634	1938

+60 +60 +60 +60

including decryption function

Adding decryption functions is almost free!

* FOM = (nanobit per cycles) / area squared [GE^2]

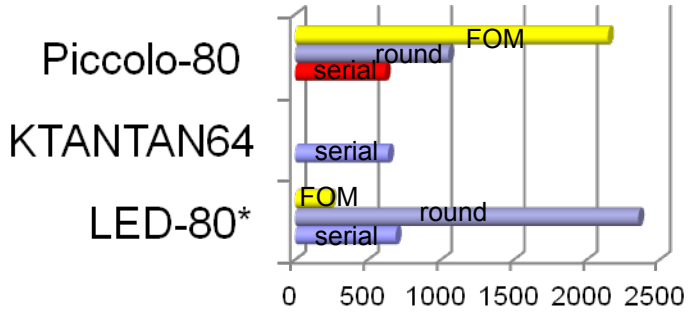
** 0.13 um standard cell library

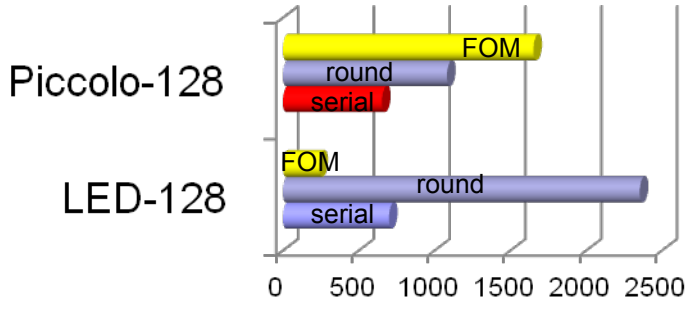
*** 1 GE = 2-way NAND

Efficiency comparison

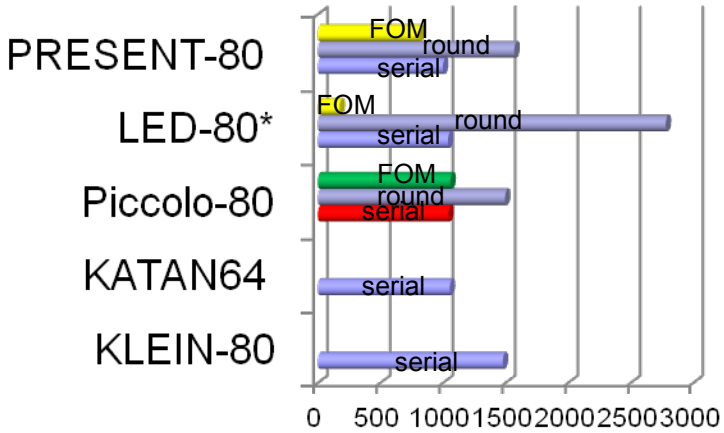
Target:
same block size
same key size

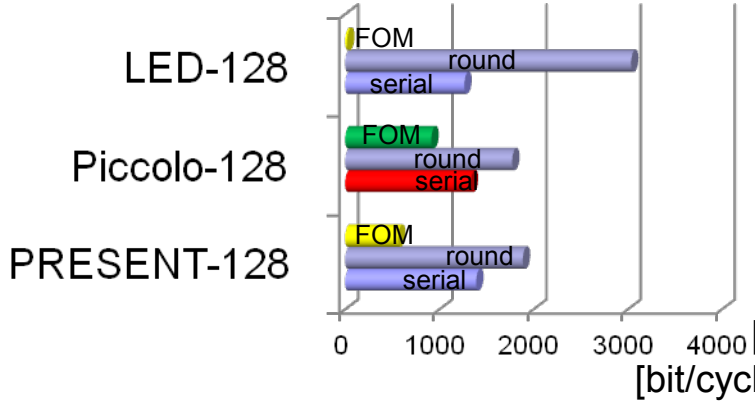
- **Piccolo is smallest in fixed key setting !**





- **Piccolo is very small and high FOM in flexible key setting !**





Conclusion

- Proposed an ultra-lightweight blockcipher “Piccolo”
  - Security
    - secure against known attack including MITM and RKA
  - Performance
    - one of the most compact ciphers
    - achieved the best performance w.r.t. energy consumption
- Further analysis is very welcome!

Thank you for your attention!