CryptArchi 2019

Acceleration of Lightweight Block Cipher Execution on Microprocessors

E. Tehrani, T. Graba and J.L. Danger

23-26 June 2019

Outline

Outline

TELECOM Paris

Context

- IoT, connected cars, wearable medical devices
- Such devices require security:
  - Lots of **communication**
  - Avoid remote access to intern functions
  - Need for cryptographic strong ciphers with low overhead
    ➜ Lightweight Block Ciphers (LBC)
- Lack of standard means a plethora of cipher exist:
  - Some applications require using **multiple** ciphers
  - Microprocessors use common instructions
  - Lightweight Cryptography is not common enough
  - Implementations need to **high security** and **low complexity**
- Microprocessors need **specific instructions** to execute ciphers faster

Algorithm Decomposition

- Each round of an algorithm can be decomposed in three main steps:
  - **Key Addition**: Adding the secret
  - **Confusion**: Making sure the output is different from the input
  - **Diffusion**: Making sure that a single change will impact as much of the result as possible
- Only the datapath is considered, Key Scheduling will be executed offline
- Additional *hardware* instructions will be part of the **cryptographic extension** of an ASIP

Restriction on the Studied Ciphers

There is a plethora of ciphers but not all fit modern criteria, we therefore limited our study to:

- 128-bit minimal key size, to ensure **security**
- 4x4 Sbox, to ensure **coherence** between the ciphers
- 64-bit block size, as a way of **minimizing the cost**

Considered Ciphers

	Key size (in bits)	4x4 Sbox	Block size (in bits)
GOST	256	✓	64
RC5	0-255	✗	64
Rectangle	128	✓	64
Simeck	128	✗	64
Twine	128	✓	64
XTEA	128	✗	64
Skinny	128	✓	64
Midori	128	✓	64
PRESENT	128	✓	64
GIFT	128	✓	64

Outline

Identifying Key Functions

- Each algorithm has similarities with the others
  - **3 main steps**
- Each algorithm has its **specificities**
  - What are those specificities ?
  - Can they be gathered to minimize the amount of instructions needed ?

Implementing multiple ciphers with as little instructions as possible requires a precise **classification**

Key Functions

The three main steps are:

- Key Addition ➜ XOR with the key
- Confusion ➜ 4x4 LUTs
- Diffusion ➜ Requires classification

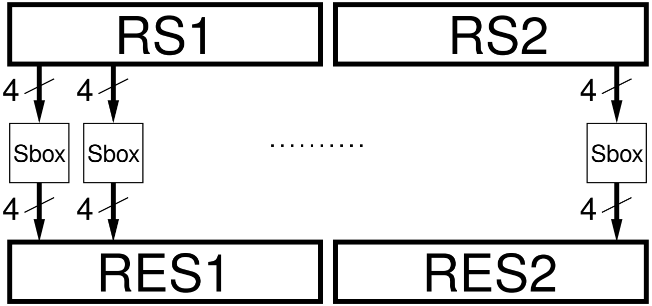
Confusion

4x4 Sbox are used in most ciphers

- Each algorithm uses 8 or 16 **identical** 4x4 Sbox
- LUTs is a **generic** way to implement such a non-linear function
- These LUTs can be used in **parallel**

Adding this instruction offers an important gain

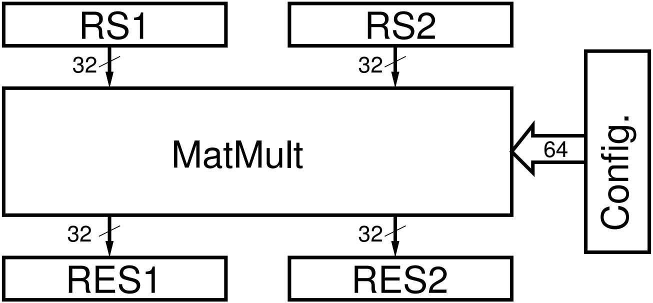
Confusion



Different Types of Diffusion

- **Bit-Level:**
  - Simple Rotation
    - *GOST, RC5, Rectangle, Simeck, XTea, Midori, Skinny, Twine*
  - Complex Permutation: specific instructions
    - *PRESENT, GIFT*
- **Nibble-Level:**
  - Simple Rotation: "ShiftRow"
    - *Midori, Skinny*
  - Matrix Multiplication: "MixColumn"
    - *Midori, Skinny, Twine*

Different Types of Diffusion



Different Types of Diffusion

- **Bit-Level:**
  - Simple Rotation
    - *GOST, RC5, Rectangle, Simeck, XTea, Midori, Skinny, Twine*
  - Complex Permutation: specific instructions
    - *PRESENT, GIFT*
- **Nibble-Level:**
  - Simple Rotation: "ShiftRow"
    - *Midori, Skinny*
  - Matrix Multiplication: "MixColumn"
    - *Midori, Skinny, Twine*

Diffusion

- Bit-level shift: **Rotation**
  - *Small gain but often used*
- Nibble-level Rotation and Permutation: **Matrix multiplication**
  - *Important gain and used quite often*
- Bit-level Permutation: **Specific instructions**, with very low hardware cost
  - *Huge gain but works for a single cipher*

Classification

What makes an LBC specific is its **Diffusion step**, their classification is therefore according to it:

- Bit-level Simple Rotation Ciphers
- Bit-level Complex Permutation Ciphers
- Nibble-level Matrix Multiplication Ciphers

Outline

Targeted ISA: RiscV

RiscV is a modern open-source ISA with existing extensions such as:

- **I** (Base Integer) extension is all the basic operations
- **E** (Embedded) with only 16 registers
- **C** (Compressed) with 16-bit instructions

We chose to work on the **32-bit** RiscV

RiscV Implementation: VexRisc

After an overview of multiple RiscV implementations we elected the **VexRisc**:

- **C** and **E** extensions available
- **SpinalHDL** makes the implementation easy to modify
- Adding instructions can be achieved through simple plug-ins

C Language Gain

- A: Basic 32I assembly-language (with rotation)

Algorithm	Instruction per round			
	A	B	C	D
GOST	216			
RC5	15			
Rectangle	238			
Simeck	12			
Twine	186			
XTEA	24			
Skinny	234			
Midori	351			
PRESENT	555			
GIFT	645			

C Language Gain

- A: Basic 32I assembly-language (with rotation)
- B: With Sbox instruction

Algorithm	Instruction per round			
	A	B	C	D
GOST	216	11		
RC5	15	15		
Rectangle	238	33		
Simeck	12	12		
Twine	186	149		
XTEA	24	24		
Skinny	234	29		
Midori	351	146		
PRESENT	555	350		
GIFT	645	440		

C Language Gain

- A: Basic 32I assembly-language (with rotation)
- B: With Sbox instruction
- C: + Matmult instruction

Algorithm	Instruction per round			
	A	B	C	D
GOST	216	11	11	
RC5	15	15	15	
Rectangle	238	33	33	
Simeck	12	12	12	
Twine	186	149	9	
XTEA	24	24	24	
Skinny	234	29	9	
Midori	351	146	9	
PRESENT	555	350	350	
GIFT	645	440	440	

C Language Gain

- A: Basic 32I assembly-language (with rotation)
- B: With Sbox instruction
- C: + Matmult instruction
- D: + Specific 64-bit permutation

Algorithm	Instruction per round			
	A	B	C	D
GOST	216	11	11	11
RC5	15	15	15	15
Rectangle	238	33	33	33
Simeck	12	12	12	12
Twine	186	149	9	9
XTEA	24	24	24	24
Skinny	234	29	9	9
Midori	351	146	9	9
PRESENT	555	350	350	5
GIFT	645	440	440	5

Outline

Introduction: Lightweight Block Ciphers

Classification of Lightweight Block Ciphers

Implementation Results

Conclusion & Future Work

Conclusion & Future Work

Conclusion:

- Microprocessors are not adapted to the implementation of LBCs
- Lightweight Block Ciphers can be gathered within sub-groups
- Each sub-group corresponds to a specific instruction which reduces instruction cost drastically
- This issue can be solved through a cryptographic extension

Future Work:

- Optimizing the implementation of the Matrix Multiplication instruction
- Using a simulator to study the real cost of those instructions
- Gathering EM datas on an FPGA chip

Thank you for listening

Any questions ?