MITSUBISHI ELECTRIC
Changes for the Better

for a greener tomorrow eco Changes

Minimalism of Software Implementation

– Extensive Performance Analysis of Symmetric Primitives on the RL78 Microcontroller -

Mitsuru Matsui and Yumiko Murakami
Information Technology R&D Center
Mitsubishi Electric Corporation

Agenda

1. Introduction
   - Our motivation
   - Previous work
   - Our aim and contributions
2. RL78 microcontroller
3. Interface and Metrics
4. Comparative Figures
   - Block ciphers
   - Hash functions
5. Implementation highlights
6. Conclusions

Introduction

Our motivation

Recent light-weight cryptography is mainly discussed from the aspect of hardware design.

- **How about SOFTWARE?**
- In particular, EMBEDDED software?

Software implementation of light-weight cryptography is an important issue and needs to be more discussed.

Previous work

ECRYPT II project*

– implemented block ciphers and hash functions on ATtiny45 processor (4KB-ROM, 256B-RAM) in assembly language, and

– published the performance evaluation results, which aimed at the top speed record for each primitive on the processor.

***ECRYPT II, Implementations of low cost block-ciphers/hash-functions in Atmel AVR devices,**
http://perso.uclouvain.be/fstandae/source_codes/lightweight_ciphers/
http://perso.uclouvain.be/fstandae/source_codes/hash_atmel/

Our aim

ROM/RAM sizes available to crypto primitives are usually determined by somebody outside crypto!

What embedded programmers want to know is

- The target primitive can be implemented within the given resource constraints?
- Which primitive is fastest in the given resource?

We aim at demonstrating overall performance figure:

- Various size-and-speed tradeoffs for each primitive.
- What ROM/RAM size combinations are possible or impossible.

Our contributions

To show various size-and-speed tradeoffs for each primitive,

- classified available ROM/RAM size combinations into several categories.
  - 512B, 1KB, and 2KB for ROM-size
  - 64B, 128B, 256B, and 512B for RAM-size
- optimized speed in each category
  e.g., ROM-2KB/RAM-128B.

In addition, we show other tradeoffs for some primitives

- Fastest code (at the cost of ROM size)
- Smallest ROM size (at the cost of speed)

Target primitives

- Block ciphers
  - AES, Camellia (ISO/IEC18033-3)
  - CLEFIA, PRESENT (ISO/IEC29192-2)
- Hash functions
  - SHA-256/512
  - Keccak-256/512, Skein-256/512, Groestl-256/512 (SHA-3 finalists)

Skein-256: Skein-256-256

Skein-512: Skein-512-512

Keccak-256: Keccak[r=1088, c=512]

Keccak-512: Keccak[r=576, c=1024]

RL78 microcontroller

RL78 microcontroller

Our target: the RL78 microcontroller (by Renesas Electronics):

- 8/16-bit low-end microcontroller
- From general-purpose to in-vehicle
- Wide memory variations up to 512KB/32KB ROM/RAM
  - The minimum ROM/RAM sizes are 2KB/256B
- CISC processor with eight general registers (a,x,b,c,d,e,h,l)
  - ECRYPT II's target, ATtiny, is a RISC processor with 32 registers

Instruction examples

Instruction		Byte	Cycle
addw	ax, [hl+byte]	3	1
xor/or/and	reg1, reg2	1	1
shl/shr	a/b/c, cnt	2	1
shlw/shrw	ax/bc, cnt	2	1
rolc/rorc	a,1	2	1
skc/sknc/skz/sknz		2	1
push/pop	regpair	1	1
call	adr	3	3
ret		1	6

- Many instructions allow only register a/ax as a destination register and only register pair hl as a general address pointer.
- On the other hand, it supports read-modify instructions and its average instruction length is short.

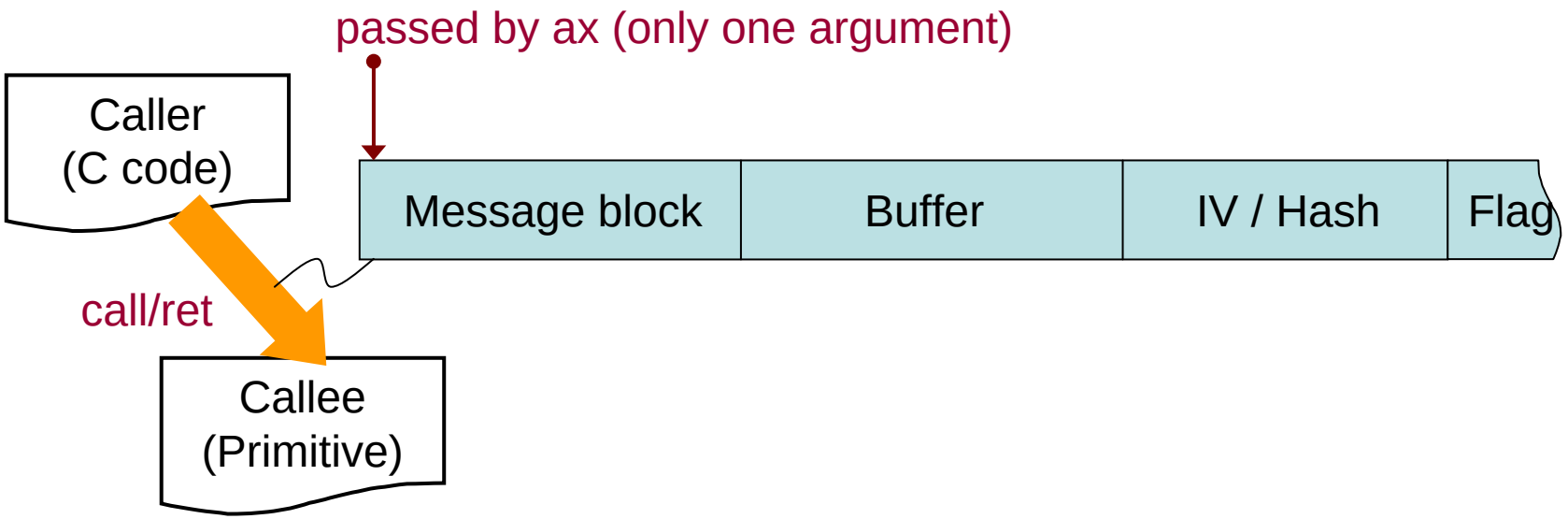
Interface and Metrics

Interface

We adopted a simple and portable program interface -

– commonly accepted in embedded software.
  – a subroutine callable from a high level language

– based on the calling conventions of Renesas's RL78 development tool.
  – using the first argument only, which is passed by `ax`
  – register pair `hl` must be recovered at the end of the routine



Metrics

Our purposes:

– to get an overall performance figure on size-and-speed tradeoffs for each primitive, and

– to reveal that a specific size and speed combination is possible/impossible.

Minimize the ROM size without caring the speed.

ROM-1KB/RAM-128B is enough for this primitive.

Portfolio of a primitive (example)

	ROM-Min(400B)	ROM-512B	ROM-1KB	ROM-2KB
RAM-128B	**20,000**	**9,000**	**3,000**	**—**
RAM-64B	**x**	**x**	**4,000**	**3,500**

(cycles/block)

When only ROM-512B/RAM-64B is available, this primitive is not an option.

`—' : "Satiated": the top speed is already obtained in other category

`x' : The primitive is (seems) impossible to implement in the category

How to count ROM/RAM size

No consensus* of how to count ROM and RAM sizes of a given crypto routine.

How to count RAM size should be unambiguously defined.

- RAM is more expensive than ROM in an embedded system.

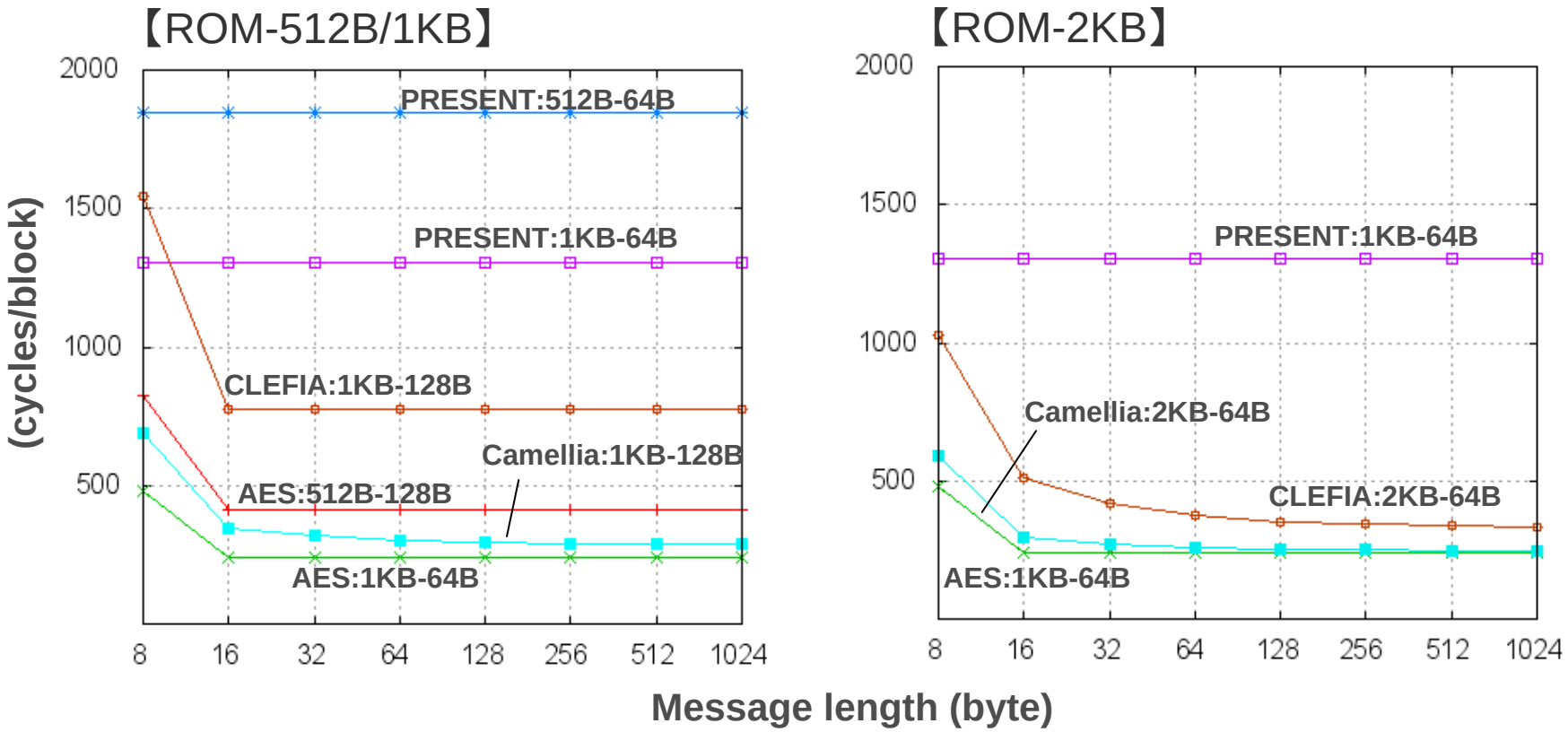
In our metric, ROM and RAM sizes should indicate the entire resource consumption of a target subroutine.

* some examples of previous work:

- mandatory parameters (such as plaintext and key) were not counted;
- stack consumption was not taken into account;
- calling convention was ignored (no register was saved/restored in a subroutine).

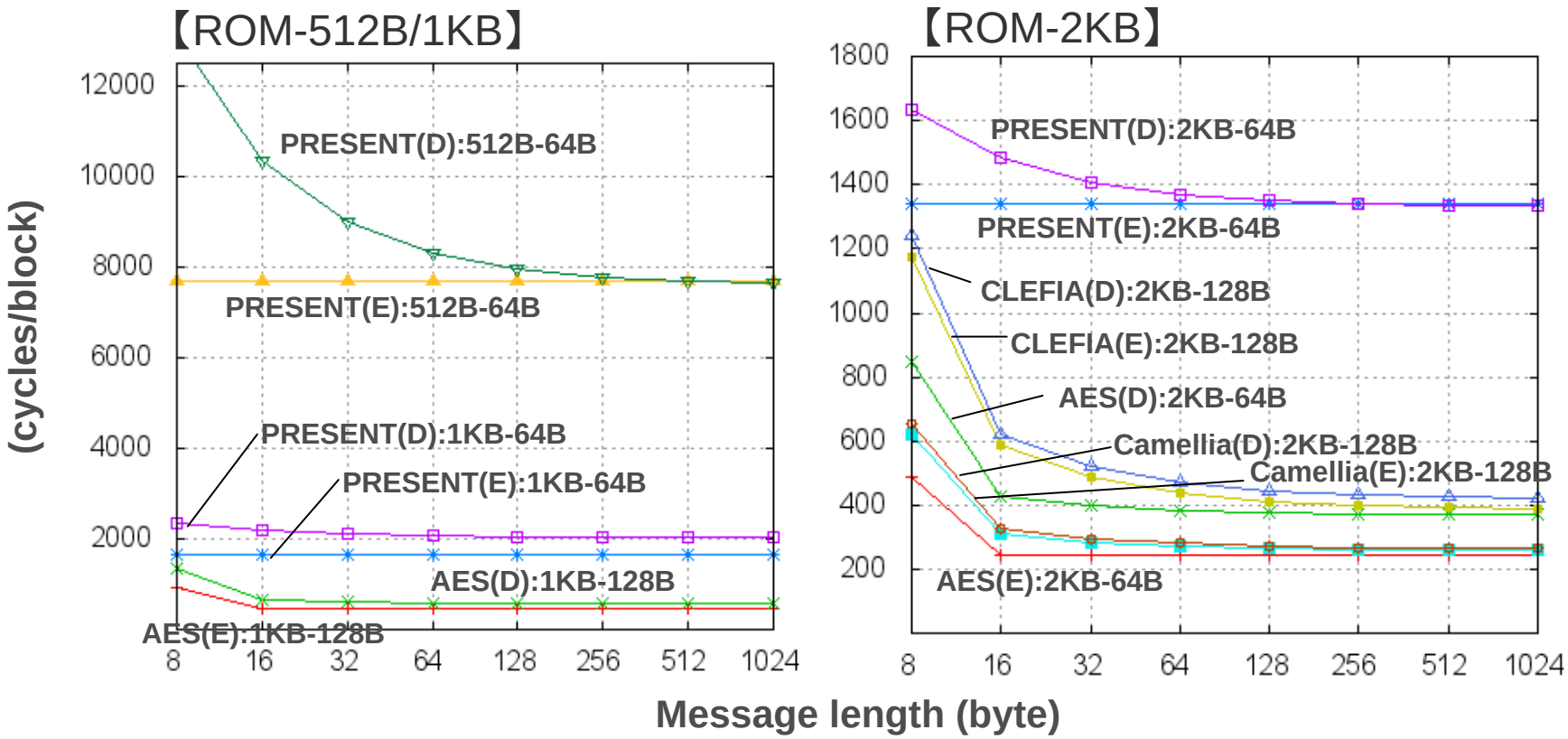
Comparative figures
- Block ciphers -

Speed comparison (Enc-only)



- AES and Camellia show an overall excellent performance
- Only AES and PRESENT are options when 512B ROM is available.
- Only PRESENT survives with ROM-512B and RAM-64B

Speed comparison (Enc+Dec)

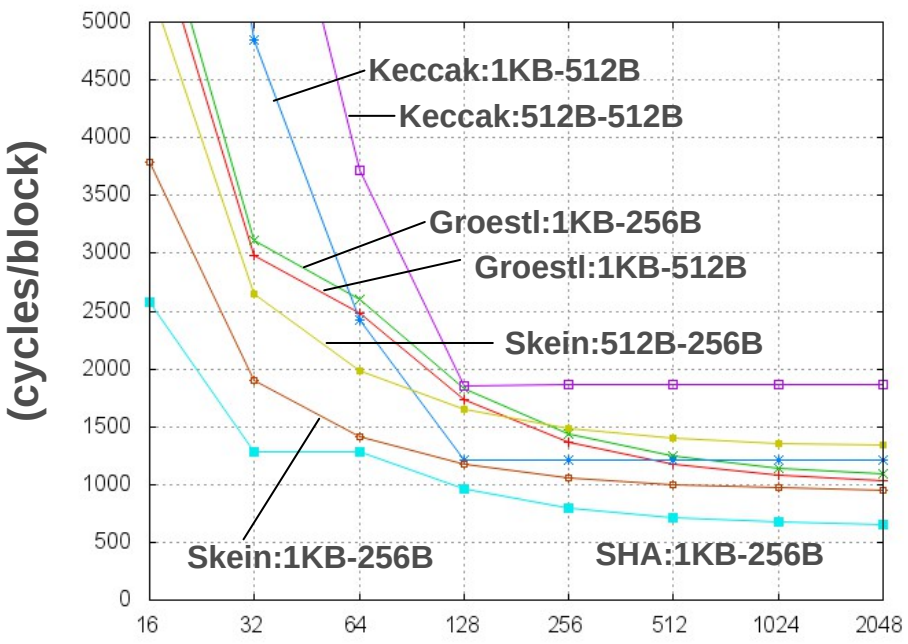


- We can see three speed groups.
- Neither Camellia nor CLEFIA is an option with 1KB ROM.
- Decryption of Camellia is faster than that of AES when 2KB ROM is available.

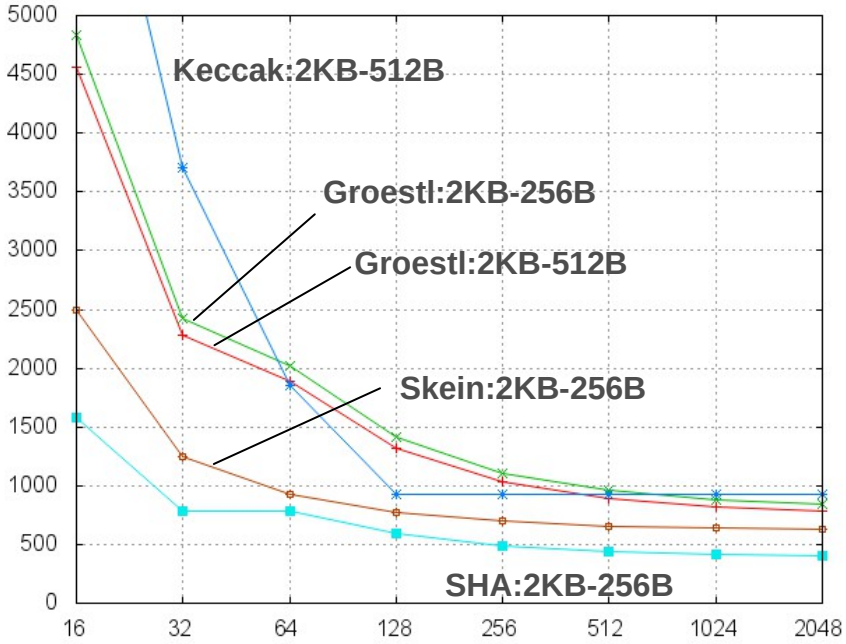
Comparative figures
- Hash functions -

Speed comparison (256-bit Hash)

【ROM-512B/1KB】



【ROM-2KB】

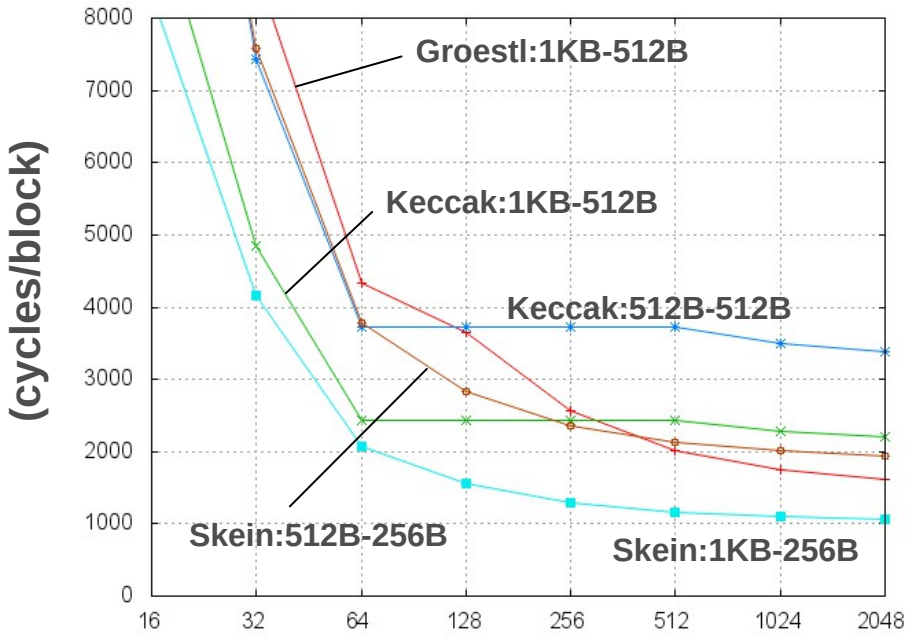


Message length (byte)

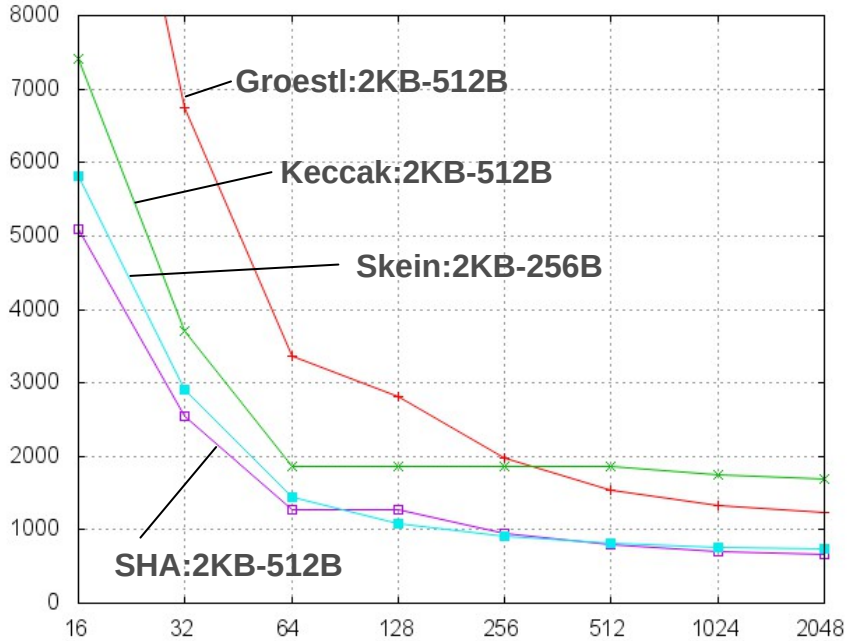
- SHA-256 is still the best choice if 1KB ROM is given.
- When ROM size is limited to 512 bytes, then SHA-256 is excluded and Keccak-256 and Skein-256 survive.
- SHA > Skein > Groestl > Keccak when message is long

Speed comparison (512-bit Hash)

【ROM-512B/1KB】



【ROM-2KB】



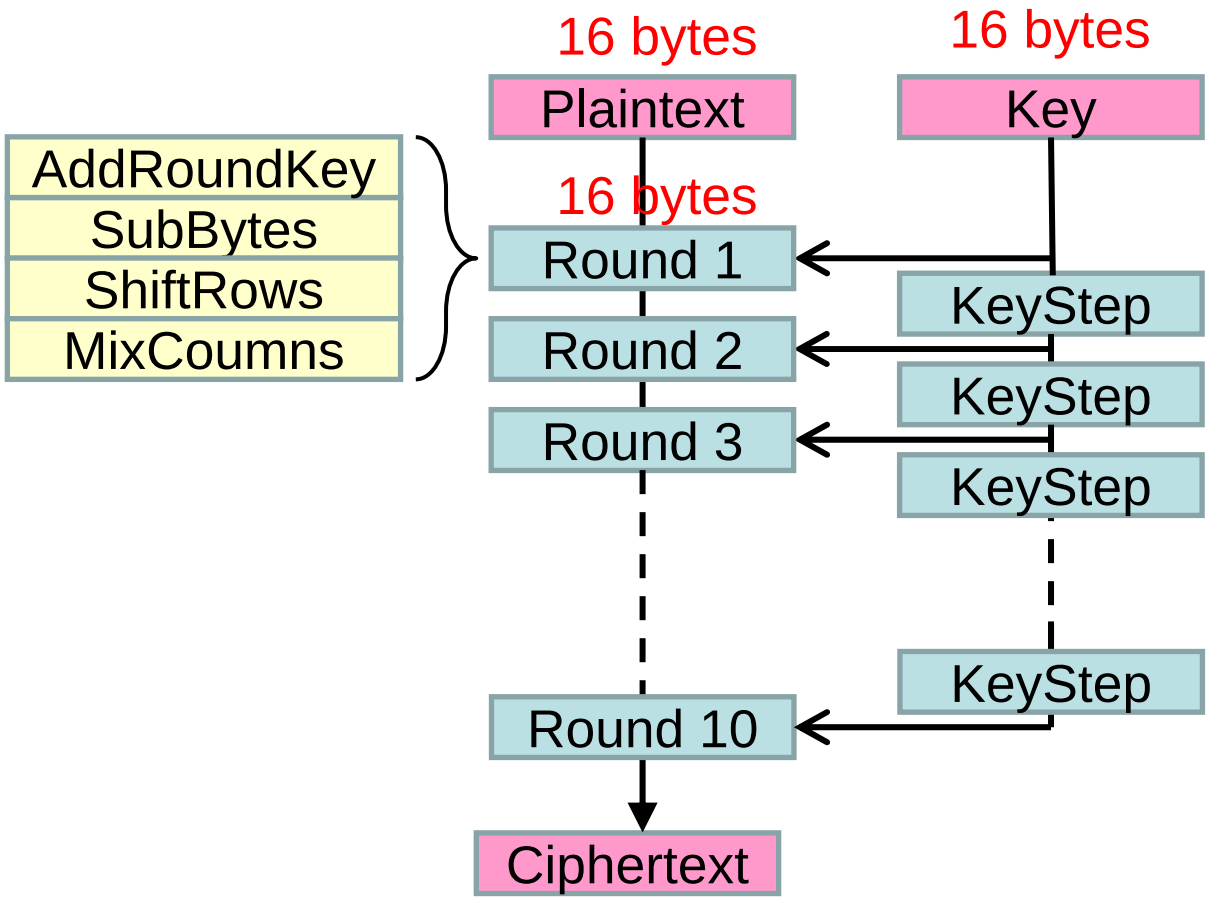
Message length (byte)

- Only Skein-512 is an option when RAM is limited to 256B.
- SHA-512 and Skein-512 are fastest with 2KB ROM.
- Only Keccak-512 and Skein-512 survive with 512B ROM.

Implementation highlights

AES-128

Initial Observation (Algorithm and Required Memory)



RAM:
64B (not easy) or
128B (enough)

Constant ROM:
256B (S-box)
256B (Inv S-Box)

AES-128

Implementation of MixColumn (+SubButes+ShiftRows)

44 instructions

$$\begin{bmatrix} x \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[in+0] \\ S[in+1] \\ S[in+2] \\ S[in+3] \end{bmatrix}$$

```
SBOX    [in+0]
mov     c,a
mov     d,a
mov     e,a
GMUL2
mov     x,a
xor     e,a

SBOX    [in+1]
xor     x,a
xor     d,a
xor     e,a
GMUL2
xor     x,a
xor     c,a
```

```
SBOX    [in+2]
xor     x,a
xor     c,a
xor     e,a
GMUL2
xor     c,a
xor     d,a

SBOX    [in+3]
xor     x,a
xor     c,a
xor     d,a
GMUL2
xor     d,a
xor     e,a
```

```
SBOX [mem] =
  mov   a,[mem]
  mov   b,a
  mov   a,S[b]

GMUL2 =
  shl   a,1
  sknc
  xor   a,#01BH
```

a ←2a (in GF(2^8))

AES-128

incl. S-box table (256B)

Enc-only	ROM-Min(486B)	ROM-512B	ROM-1024B
RAM-128B	7,288	6,622	-
RAM-64B	x	x	3855

Loop in MixColumns (one MixColumn code)

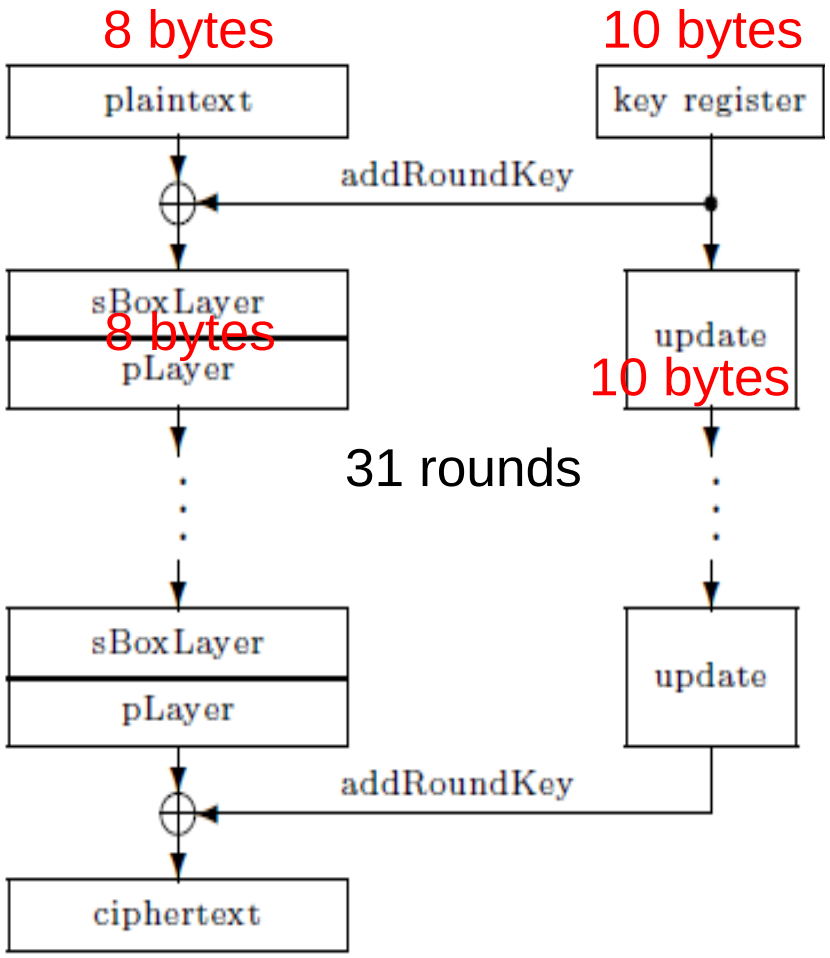
"Flat" Implementation (four MixColumn codes)

incl. S-box tables (512B)

Enc+Dec		ROM-Min(970B)	ROM-1024B	ROM-2048B	Fast (2380B)
RAM-128B	Enc	7,743	7,339	-	-
	Dec	12,683 / 10862	10,636 / 9,106		
RAM-64B	Enc	x	x	3,917	3,865
	Dec			6,804 / 5,911	6,541 / 5,706

PRESENT-80

Hardware “Ultra-Lightweight” 64-bit Block Cipher



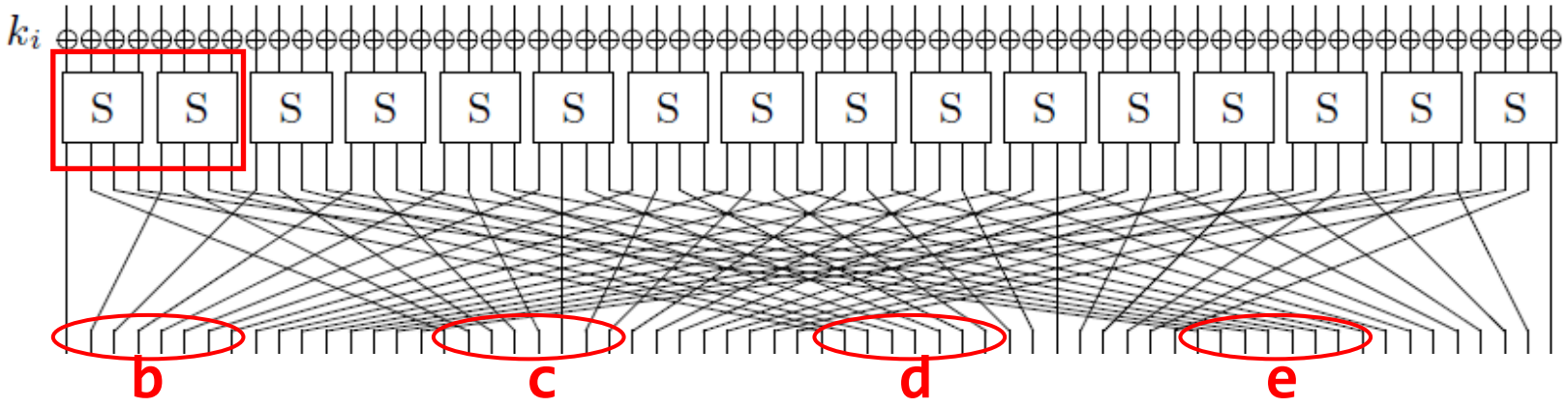
RAM:
64B is enough

Constant ROM:
16B (S-box) or
256B (S-box||S-box)

16B (Inv S-box) or
256B (Inv Sbox||Inv S-box)

PRESENT-80

Implementation of sBoxLayer+pLayer




```
mov   a,SS[mem]
mov   x,a
addw  ax,ax
xch   a,b
addw  ax,ax
xch   a,c
addw  ax,ax
xch   a,d
addw  ax,ax
xch   a,e
```

x4

repetition of this code makes one round

PRESENT-80

Enc-only	ROM-Min(210B)	ROM-512B	ROM-1024B
RAM-64B	144,879	12,200	9,007

4x4 Sbox　　4x4 Sbox　　8x8 Sbox

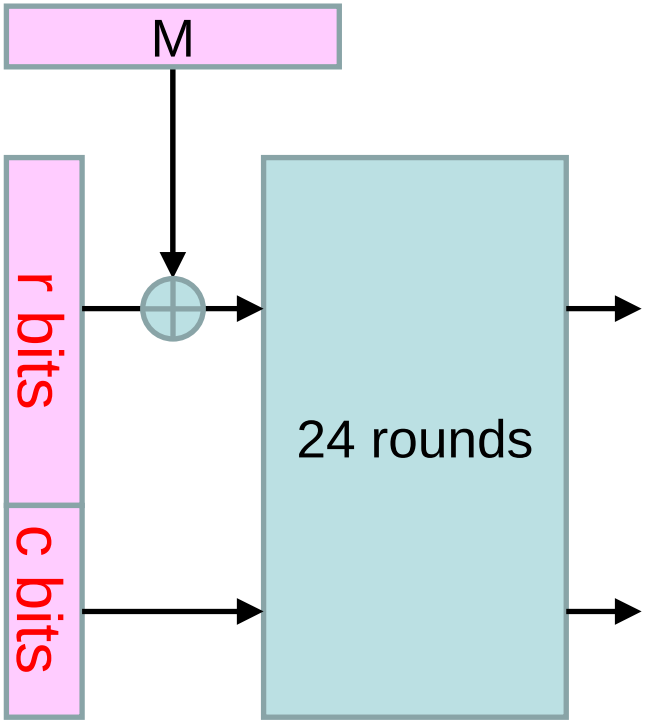
```
mov  a,[mem]
addw ax,ax
mov  [mem],a
```

repetition

Enc+Dec		ROM-512B	ROM-1024B	ROM-2048B
RAM-64B	Enc	61,634	13,883	9,007
	Dec	104,902 / 60,384	16,046 / 14,014	10,823 / 8,920

Keccak

Initial Observation



Hash Size	r	c
224	1152	448
256	1088	512
384	832	768
512	576	1024

At least 256B RAM is needed
→ Assume 512B RAM is given

Constant ROM Data
96B (RC: round constant)

can be reduced by “on-the-fly”
but usually very expensive

r+c = 1600 (SHA-3 parameters)

Keccak

Round Function: Input A[5][5], Output E[5][5]

θ step

for $x = 0$ to 4 **do**
  $C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$
end for
for $x = 0$ to 4 **do**
  $D[x] = C[x-1] \oplus \mathrm{ROT}(C[x+1], 1)$
end for

for $y = 0$ to 4 **do**

ρ, π steps

  for $x = 0$ to 4 **do**
    $B[x] = \mathrm{ROT}((A[x', y'] \oplus D[x']), r[x', y'])$, with $\begin{pmatrix} x' \\ y' \end{pmatrix} = M^{-1} \begin{pmatrix} x \\ y \end{pmatrix}$
  end for

χ step

  for $x = 0$ to 4 **do**
    $E[x,y] = B[x] \oplus ((\mathrm{NOT}\, B[x+1]) \,\mathrm{AND}\, B[x+2])$
  end for

end for

ι step

$E[0,0] = E[0,0] \oplus \mathrm{RC}[i]$

24 different shift counts

Inner most operations

Keccak

How to treat 24 different shift counts

	ROM-1024B	ROM-2048B	Fast (2,214B)
RAM-512B	155,209 / 155,703	118,705 / 119,171	110,185 / 110,651

$r=8r_1+r_2$
$r_1,r_2=0,1,2,..,7$

14 shift routines

23 shift routines

24 shift routines

	ROM-Min(453B)	ROM-512B
RAM-512B	516,528 / 517,022	237,960 / 238,454

1 shift routine
(1bit only)

2 shift routines
(1bit+1byte)

MITSUBISHI ELECTRIC
Changes for the Better

Conclusions

for a greener tomorrow eco Changes

- Discussed embedded software implementation of light-weight cryptographic primitives.
- Explored size-speed tradeoffs with various “given” ROM/RAM combinations. This looks a new approach.
- Reducing program size with minimizing performance penalty is a tricky puzzle.
  - e.g. how to select a small number of “good” rotate-shift routines when a target primitive contains many rotate-shifts with different shift counts.