Implementation of Four AES Candidates on Two Smart Cards

G. Hachez, F. Koeune,
J.-J. Quisquater

NIST required

- Performances on an IBM-compatible PC, with an Intel Pentium Pro Processor, 200MHz clock speed, 64MB RAM, running Windows95, with ANSI C compiler in the Borland C++ Development Suite 5.0.
- Performance on an 8-bit architecture

Many answers, very different

- Different architectures (6805, 8051, …)
- Full implementation <>
  general considerations
- Not always RAM usage
- ...

→ Real comparison impossible

What we need

- Implement systems on *same platform*
- Compare them on *several points of view*
  - RAM usage
  - speed
  - code size
  - tables size

Outline of the talk

- Chosen smart cards
- Implementation decisions
- Chosen candidates
- Results for each candidate
- Comparison graphics

Platforms

- Two smart cards :
  - basic, low-cost smart card : 8051
  - sophisticated, advanced one: ARM

8051

- 8 bit
- ~ 256 bytes of RAM
- accumulator-based
- 8 registers
  (but only 2 for adressing)
- CISC

ARM

- 32 bit
- ~ 1 KB of RAM
- 3-operand instructions
- 16 registers
  (including PC, FP, …)
- RISC

8051

- several addressing modes, but
- non-orthogonal
- operands: constants, registers, RAM
- multiplier

  8 x 8 $\rightarrow$ 16 bits

ARM

- even more addressing modes
- orthogonal
- operands always in registers or small constants
- multiplier

  32 x 32 $\rightarrow$ 32 bits

ARM: other features

+ Conditional execution

    e.g. ADDCS : ‘‘add if carry bit set’’

+ Barrel shifter

    R0 = R1 + (R2 << 4)

– Slow memory access

    (read takes 3 clock cycles)

Our decisions

- Implement only 128-bit variant
- priorities :
  1. RAM usage
  2. speed
  3. ROM usage (code, table size)
- all non constant data in RAM (no EEPROM, …)
- only key schedule + encryption

The candidates

- Eliminated candidates that were :
  - broken (Frog, Magenta, ...)
  - probably not suitable (HPC)
- chose four among ‘‘likely finalists’’:
  E2, RC6, Rijndael, Twofish
- under progress (master theses at UCL):
  Mars, Serpent

E2

- Does not allow on-the-fly key schedule

$\longrightarrow$ more than 300 bytes RAM needed

- speed :
  - 35800 clock cycles on 8051
  - 10350 clock cycles on ARM
- code + table size : about 1400 bytes

RC6

- Does not allow on-the-fly key schedule

  $\longrightarrow$ RAM :

  - 205 bytes on the 8051
  - 176 bytes on the ARM
- very fast on the ARM : 3021 cycles
- much slower on the 8051 : 57600 cycles (many rotations)
- very short code : 272 - 596 bytes

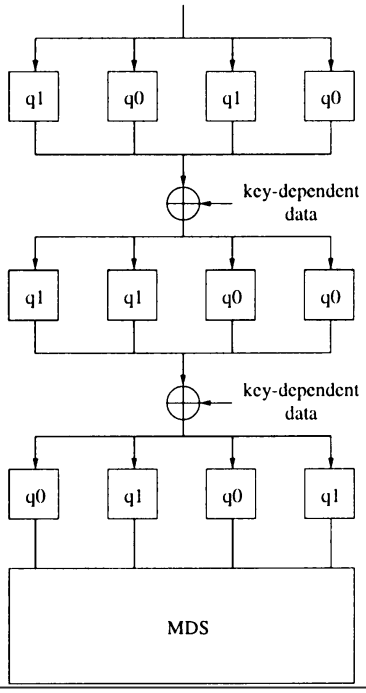
Rijndael

The best candidate for smart cards :

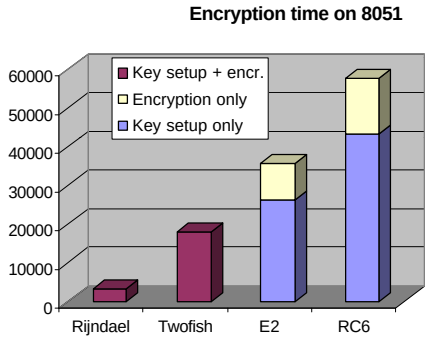
- low RAM :
  - 49 bytes on 8051
  - 0 - 16 bytes on ARM
- speed :
  - 8051 : 3168 cycles (authors' implementation)
  - ARM : 1467 cycles
- code : 768 - 2620 bytes

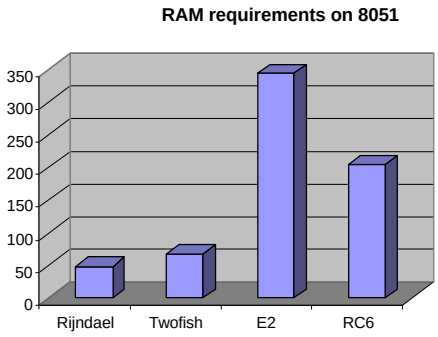
Twofish

- Low RAM usage :
  - 68 bytes on 8051
  - 48 bytes on ARM
- compared to other platforms, rather slow on the ARM ($\rightarrow$ many table lookups)
- speed :
  - 8051 : 18126 clock cycles
  - ARM : 8406
- ROM : 1400 - 5300 bytes

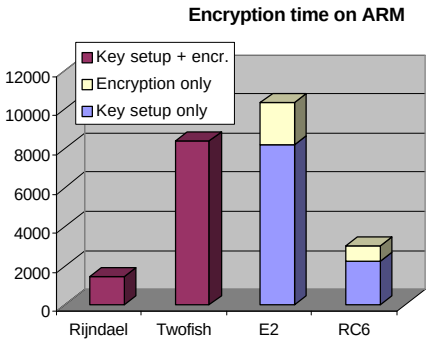


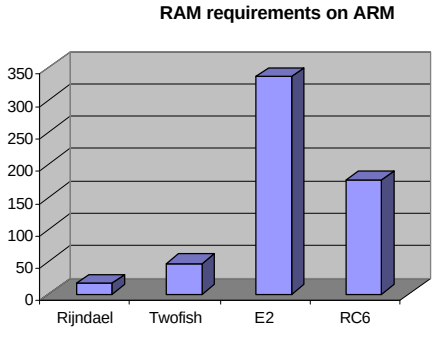
Summary

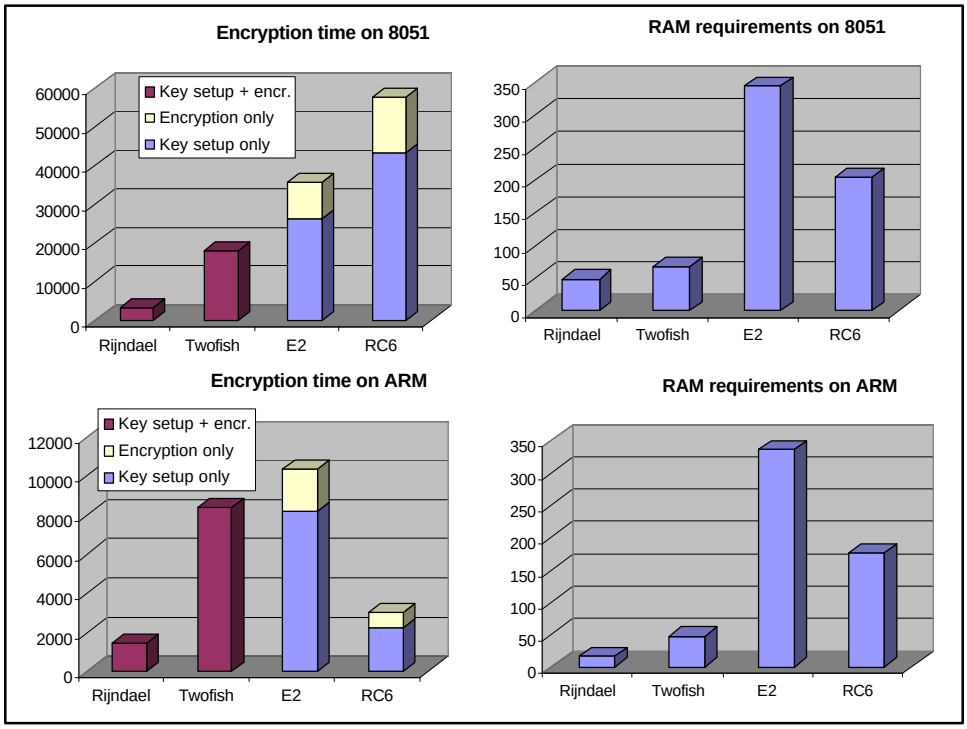




Summary (2)







To come …

- Mars
- Serpent
- …
- results will be put on cAESar's page :
  http://www.dice.ucl.ac.be/crypto/CAESAR

8051

Algorithm	Code size	Table size	RAM usage	Key setup	Encryption
E2	1188	256	344 *	26147	9725
RC6	596	0	205 *	43200	14400
Rijndael	512	256	49 *	4065	
	760	256	49 *	3168	
Twofish	931	512	68	24422	
	879	1024	68	18126	

* + 16 bytes if original key is to be preserved

ARM

Algorithm	Code size	Table size	RAM usage	Key setup	Encryption
E2	1004	256	336 *	8172	2180
RC6	272	0	176 *	3903	790
	460	0	176 *	2231	790
Rijndael	1148	256	0 *	2889	
	2620	1280	16 *	1467	
Twofish	908	512	48	13662	
	696	4608	48	8406	

* + 16 bytes if original key is to be preserved