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 vs. ARM

<table>
<thead>
<tr>
<th>8051</th>
<th>ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bit</td>
<td>32 bit</td>
</tr>
<tr>
<td>~ 256 bytes of RAM</td>
<td>~ 1 KB of RAM</td>
</tr>
<tr>
<td>accumulator-based</td>
<td>3-operand instructions</td>
</tr>
<tr>
<td>8 registers (but only 2 for addressing)</td>
<td>16 registers (including PC, FP, …)</td>
</tr>
<tr>
<td>CISC</td>
<td>RISC</td>
</tr>
</tbody>
</table>

- **8051**
  - several addressing modes, but non-orthogonal
  - operands: constants, registers, RAM
  - multiplier: 8 x 8 → 16 bits

- **ARM**
  - even more addressing modes
  - orthogonal
  - operands always in registers or small constants
  - multiplier: 32 x 32 → 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
  → 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
  → 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 (→ many table lookups)
- speed:
  - 8051: 18126 clock cycles
  - ARM: 8406
- ROM: 1400 - 5300 bytes
Summary

**Encryption time on 8051**

- Key setup + encr.
- Encryption only
- Key setup only

**RAM requirements on 8051**

- Rijndael
- Twofish
- E2
- RC6

---

Summary (2)

**Encryption time on ARM**

- Key setup + encr.
- Encryption only
- Key setup only

**RAM requirements on ARM**

- Rijndael
- Twofish
- E2
- RC6
To come …

- Mars
- Serpent
- …

- results will be put on cAESar’s page:
  
  http://www.dice.ucl.ac.be/crypto/CAESAR
8051

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Code size</th>
<th>Table size</th>
<th>RAM usage</th>
<th>Key setup</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>1188</td>
<td>256</td>
<td>344 *</td>
<td>26147</td>
<td>9725</td>
</tr>
<tr>
<td>RC6</td>
<td>596</td>
<td>0</td>
<td>205 *</td>
<td>43200</td>
<td>14400</td>
</tr>
<tr>
<td>Rijndael</td>
<td>512</td>
<td>256</td>
<td>49 *</td>
<td>4065</td>
<td></td>
</tr>
<tr>
<td></td>
<td>760</td>
<td>256</td>
<td>49 *</td>
<td></td>
<td>3168</td>
</tr>
<tr>
<td>Twofish</td>
<td>931</td>
<td>512</td>
<td>68</td>
<td>24422</td>
<td></td>
</tr>
<tr>
<td></td>
<td>879</td>
<td>1024</td>
<td>68</td>
<td>18126</td>
<td></td>
</tr>
</tbody>
</table>

* + 16 bytes if original key is to be preserved

ARM

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

* + 16 bytes if original key is to be preserved