Compact crypto implementations for embedded security

or

crypto + embedded systems = security?

Ingrid Verbauwhede
KULeuven – COSIC

E-mail: ingrid.verbauwhede@esat.kuleuven.be

Slide Acknowledgements:
Current & former Ph.D. students

Outline

• Merge: Embedded systems & Security
• Definition: secure embedded systems
• Illustrate with examples
• Challenges
• Conclusions
Embedded security: definition (1)

Old Model (simplified view):
- Attack on channel between communicating parties
- Encryption and cryptographic operations in black boxes
- Protection by strong mathematic algorithms and protocols

Embedded security: definition (2)

New Model (also simplified view):
- Attack channel and endpoints
- Encryption and cryptographic operations in gray boxes
- Protection by strong mathematic algorithms and protocols
- Protection by secure implementation

Need secure implementations not only algorithms
Embedded Security: definition (3)

NEED BOTH

• Efficient, lightweight implementations
  – Within power, area, timing budgets
  – Public key: 2048 bits RSA, 200 bit ECC on 8 bit µC and 100 µW
  – Public key on a passive RFID tag

• Trustworthy implementation
  – Resistant to attacks
  – Active attacks: probing, power glitches, JTAG scan chain
  – Passive attacks: side channel attacks

Illustrate with examples

• Example 1: Secret Key: KATAN, KTANTAN
• Example 2: NIST SHA3 – how not to do it
• Example 3: Public key for RFID
Secret key: KATAN, KTANTAN

Christophe De Cannière, Orr Dunkelman and Miroslav Knežević
CHES 2009
[slide courtesy: M. Knežević]

KATAN/KTANTAN Design Goal

- *Minimum* logic (i.e. gates) to implement a secret key algorithm

Alternatives:

- Stream ciphers
  - To ensure security, the internal state must be twice the size of the key.
  - No good methodology on how to design these.

- Use a standardized block cipher: AES
  - The smallest implementation consumes 3.1 Kgates.
  - Designed for HW and SW implementations

- Other block ciphers?
  - HIGHT, mCrypton, DESL, PRESENT, ...
  - Can we do better/different?
Design Goals

- **Secure block cipher**
  - Address Differential/Linear cryptanalysis, Related-Key/Slide attacks, Related-Key differentials, Algebraic attacks.

- **Efficient block cipher**
  - Small foot-print, Low power consumption, Reasonable performance (+ possible speed-ups).

- **Application driven**
  - Does an RFID tag always need to support a key agility?
  - Some low-end devices have one key throughout their life cycle.
  - Some of them encrypt very little data.
  - Tune algorithm to the application!

KATAN/KTANTAN Block Ciphers

- Block ciphers based on Trivium (its 2 register version – Bivium).
- Block size: 32/48/64 bits.
- Key size: 80 bits.
- Share the same number of rounds – 254.
- KATAN and KTANTAN are the same up to the key schedule.
- In KTANTAN, the key is fixed and cannot be changed!
Block Cipher – HW perspective

Datapath + Control "redundant" logic

Design Rationale – Memory Issues (1)

- For a more compact cipher, a larger ratio of the area is dedicated for storing the intermediate values and key bits.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128 [8]</td>
<td>128</td>
<td>128</td>
<td>0.25</td>
<td>3400</td>
<td>60</td>
</tr>
<tr>
<td>AES-128 [10]</td>
<td>128</td>
<td>128</td>
<td>0.13</td>
<td>3100</td>
<td>48</td>
</tr>
<tr>
<td>HIGHT [12]</td>
<td>64</td>
<td>128</td>
<td>0.25</td>
<td>3048</td>
<td>49</td>
</tr>
<tr>
<td>mCrypton [15]</td>
<td>64</td>
<td>64</td>
<td>0.13</td>
<td>2420</td>
<td>26</td>
</tr>
<tr>
<td>DES [19]</td>
<td>64</td>
<td>56</td>
<td>0.18</td>
<td>2309</td>
<td>63</td>
</tr>
<tr>
<td>DESL [19]</td>
<td>64</td>
<td>56</td>
<td>0.18</td>
<td>1848</td>
<td>79</td>
</tr>
<tr>
<td>PRESENT-80 [4]</td>
<td>64</td>
<td>80</td>
<td>0.18</td>
<td>1570</td>
<td>55</td>
</tr>
<tr>
<td>PRESENT-90 [20]</td>
<td>64</td>
<td>90</td>
<td>0.35</td>
<td>1000</td>
<td>&lt;0.01</td>
</tr>
</tbody>
</table>
Design Rationale – A Story of a Single Bit

- Assume we have a parallel load of the key and the plaintext.
- A single Flip-Flop has no relevance – MUXes need to be used.
- 2to1 MUX + FF = Scan FF: Beneficial both for area and power.

\[
\text{A}_\text{init} \quad \begin{array}{c}
0 \\
1 \\
\text{start}
\end{array} \quad \text{MUX2} \quad \begin{array}{c}
\text{D} \\
\text{Q}
\end{array} \quad \text{A}_\text{i}
\]

5 ~ 7.75 GE

7.25 ~ 13.75 GE

6.25 ~ 11.75 GE

- \((64 + 80 + 8) \times 6.25 = 950 \text{ GE}\)

Design Rationale – Control Part

- How to control such a simple construction?

\[
IR \quad \begin{array}{c}
\lambda \\
k_1 \\
l_1
\end{array} \quad \begin{array}{c}
\lambda \\
k_2 \\
l_2
\end{array} \quad \begin{array}{c}
\lambda \\
k_3
\end{array}
\]

- \(IR\) stands for Irregular update Rule.
- We basically need a counter only. Can it be simpler than that?
- Let the LFSR that is in charge of \(IR\) play the role of a counter.
Implementation Results

- All designs are synthesized with Synopsys Design Vision version Y-2006.06, using UMC 0.13µm Low-Leakage CMOS library.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>KATAN32</td>
<td>32</td>
<td>80</td>
<td>6.18</td>
<td>12.5</td>
<td>802</td>
</tr>
<tr>
<td>KATAN48</td>
<td>48</td>
<td>80</td>
<td>6.19</td>
<td>18.8</td>
<td>927</td>
</tr>
<tr>
<td>KATAN64</td>
<td>64</td>
<td>80</td>
<td>6.15</td>
<td>25.1</td>
<td>1054</td>
</tr>
<tr>
<td>KTANTAN32</td>
<td>32</td>
<td>80</td>
<td>6.10</td>
<td>12.3</td>
<td>462</td>
</tr>
<tr>
<td>KTANTAN48</td>
<td>48</td>
<td>80</td>
<td>6.14</td>
<td>18.8</td>
<td>588</td>
</tr>
<tr>
<td>KTANTAN64</td>
<td>64</td>
<td>80</td>
<td>6.17</td>
<td>25.1</td>
<td>688</td>
</tr>
</tbody>
</table>

* Throughput is estimated for frequency of 100 kHz.

SHA3 – competition: how not to do it
“Flexibility” Requirements

- Wide range of platforms
- Wide range of message digests

[of course, also security requirements]

SHA-3: “cost” requirements

- Power consumption?
- Energy to hash one message?
SHA3- results

- NIST asks for a Swiss knife
- But often you need a specialized knife
- Certainly for embedded applications

High-Throughput Implementations

http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations

Throughput [Mbps]

- SHA-3, Fully Autonomous, Tillich et al. Benchmarking
- SHA-3, Fully Autonomous, Various Authors
- SHA-3, Core Functionality, Various Authors

* Scaled to 0.18µm CMOS technology
SHA3 conclusion

- SHA3 Hash functions are HUGE compared to:
  - Secret key algorithms
    - AES: from 3000 gates and up
    - KATAN: around 1000 gates
  - Public key algorithms
    - ECC: around 10,000 gates
  - Throughput similar to
    - High speed secret key implementations

**NEED:** domain specific hash functions
Public Key: ECC for RFID

[slide courtesy:
Yong Ki Lee,
Lejla Batina]

Challenge 1: security problems

- Anti-cloning
- Privacy
- Scalability
- Replay Attack
- DoS

Public Key Crypto

- Schnorr Protocol
- Okamoto Protocol
- EC-RAC Protocol
- ...

Ingrid Verbauwhede, K.U.Leuven - COSIC
Challenge 1: Security problems

- Current RFID standards:
  - No security
  - Or simple self-destruct password (8 to 32 bits)

- Security challenges RFID:
  - Anti-cloning (make it difficult to 'copy' RFID)
  - Replay attack (query the tag and reuse that info)
  - 'Tracking' attacks => privacy problems
  - Scalability: security for large sets of tags
  - Backward/forward un-traceability

*Needs Public key*

Challenge 2: design constraints

- Area
- Power
- Performance
- Side-channel Attacks

Public key Crypto

ECC/HECC
NTRU
Rabin
Challenge 2: constraints

 Passive RFID tag:
• Area: less than 20,000 gates
• Low Power: total budget varies from 50 to 100 microWatt
• Budget for crypto: less than 15 microWatt!
• Clock frequency: factor of 13.56 MHz
• Execution time target: one point multiplication less than 250 msec.

Design Steps

• Step 1: protocols
• Step 2: algorithms
• Step 3: arithmetic
• Step 4: processor
• Step 5: circuits
Step 1: Solutions with Asymmetric-key Algorithms

- Conventional public-key authentication
  - Schnorr protocol, Okamoto Protocol
  - Vulnerable against the tracking attack
- GPS scheme
  - A variant of Schnorr protocol
  - Secure transfer of a tag’s ID is not solved
- Rabin Encryption
  - Requires a large key size and transmission
  - A compact architecture: WiSec'09(Feldhofer, Oren)

A General EC Authentication Protocol (Schnorr protocol)

\[
\begin{align*}
\{R_1 - yP\} \times r_2^{-1} &= \{r_1 P - (x r_2 + r_1) P\} \times r_2^{-1} \\
&= -x r_2 P \times r_2^{-1} = -x P
\end{align*}
\]

- A tag’s public key can be derived using exchanged messages
  => tracking attack
Observation for RFID Protocols?

- Minimize the computation load on tags
  - We need to transfer computation load to the reader/server as much as possible

- We cannot just transfer ID of a tag
  - A tag’s ID is what we need to keep in secret to avoid tracking

- The protocol is a “many to one” protocol
  - A tag’s public key (xP) does not need to be publicly known
  - It can be securely stored and used in the server

Step 2: EC based Security Processor

- Operations we need (e.g. EC-RAC)
  - Modular Operation
    - Modular Multiplication: \( r_{s1} \cdot x_1 \pmod{n} \)
    - Modular Addition: \( r_{s1} + r_{s1} x_1 \pmod{n} \)
  => Perform on a 8-bit specialized Micro-Controller
  - EC Point multiplication
    - \( r_{t1} \cdot P_1, (r_{t1} + r_{s1} x_1) \cdot Y \)
  => Perform on a 163 bit Elliptic Curve co-processor

8 bit versus 163 bit ?? modulo operations are less frequent and not time critical, hence multiplex mod operations
**Overall architecture**

**Step 3: EC Point Multiplication**

- $k \cdot P$: Scalar Multiplication
  - Montgomery Algorithm
  - ≈ 600 GE (Control)

- $P + Q$, $2 \cdot P$: Point Addition, Point Doubling
  - Lopez-Dahab Algorithm
  - ≈ 1.2k GE (Control) + 6×163 registers

- $a + b$, $a \cdot b$, $a^2$: Modular Arithmetic Operation (Addition, Multiplication, Squaring)
  - Sakiyama Modular ALU
  - ≈ 900 GE (Control) + 3×163 registers

Total Area = 2.7 GE (Control) + 9×163 registers ≈ 80% !!

* GE: Gate Equivalent (a 2-input NAND)
Step 4: Optimization Approach

- Reduce Registers: 9→5 (4 registers reduction)
  - Common Z-coordinate system: 1 register ↓
  - Redesign Modular ALU: 1 register ↓
  - Register reuse: 2 registers ↓
    - ‘Point Add/Db1 algorithm’ and ‘Modular ALU’

- Reduce Multiplexer Complexity
  - A special Circular Shift Register File
  - Extra 30% reduction in the register file

- Side Channel Resistant

Modular ALU (MALU)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication: 163/d</td>
<td></td>
</tr>
<tr>
<td>Squaring:</td>
<td>1</td>
</tr>
<tr>
<td>Addition:</td>
<td>1</td>
</tr>
</tbody>
</table>

Share XOR array
Circular Shift Register

Cost: need more cycles to get data in correct register ...
Overall cost: less than 2% compared to point multiplication

Register File Management: shift example

<table>
<thead>
<tr>
<th>Step</th>
<th>Field Operation</th>
<th>RegA</th>
<th>RegB</th>
<th>RegC</th>
<th>RegD</th>
<th>RegE</th>
<th>cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Initial</td>
<td>X_2</td>
<td>X_1</td>
<td>X_1</td>
<td>X_2</td>
<td>Z</td>
<td>–</td>
</tr>
<tr>
<td>(2)</td>
<td>T_2 ← X_1 + X_2</td>
<td>T_2</td>
<td>X_1</td>
<td>X_1</td>
<td>X_2</td>
<td>Z</td>
<td>1</td>
</tr>
<tr>
<td>(3)</td>
<td>T_2 ← T_2^2</td>
<td>T_2</td>
<td>X_1</td>
<td>X_1</td>
<td>X_2</td>
<td>Z</td>
<td>1</td>
</tr>
<tr>
<td>(4)</td>
<td>X_1</td>
<td>X_1</td>
<td>X_1</td>
<td>X_2</td>
<td>Z</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>Z</td>
<td>X_2</td>
<td>X_2</td>
<td>Z</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td>X_1</td>
<td>Z</td>
<td>T_2</td>
<td>X_2</td>
<td>X_2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td>X_1</td>
<td>X_1</td>
<td>Z</td>
<td>T_2</td>
<td>X_2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(8)</td>
<td>Z ← Z · X_1</td>
<td>Z</td>
<td>X_1</td>
<td>–</td>
<td>T_2</td>
<td>X_2</td>
<td>[163/d]</td>
</tr>
</tbody>
</table>
Estimated numbers

- Results: ECC co-processor that can compute:
  - ECC point multiplications (163 by 4)
  - Scalar modular operations (8 bit processor with redundancy)
- Schnorr (secure ID transfer, but no tracking protection): **one PM**
- More advanced protocols: up to **four PM** on tag
- Technology: 0.13 micron CMOS low power version
- Size: d=4, 14,500 gates,
- Time: 60,000 cycles for one PM
- Clock at 616 KHz, 97 msec for one PM at 22 microWatt

![Area Pie Chart]

Conclusion

**Shows ONE path:**

- Protocol design: randomized access
- Public key: ECC many design options
- Architecture: 8 bit micro & 163 EC processor
- Specialized register file
- Full custom layout
Future work:

- Do we have all the required properties covered?
- Can privacy issues be optional, at least for some applications?
- Light weight crypto?
- How to store keys securely?
- Use physical properties: PUF based security?
- What about side-channel security?