Outline & Goal

- Embedded devices
- Security for embedded devices
- Design challenges
- Crypto engineering for secret key algorithms

Embedded devices
Past, present, future

Embedded crypto everywhere
Ari Juels: RFID tracking problem

The consumer privacy problem

IMEC: Human++ project
Embedded crypto everywhere

Deep Brain stimulation

Past, present, future

Trends in design for embedded security?

- Past: efficiency
- Present: efficiency and security
- Future: efficiency, security and immersed, = “hardware-entangled cryptography & security”

Past: security for Embedded system

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

Past: Power and energy constraint!

- Power is limited
  - Cooling!!
  - Implanted devices only temperature $\Delta < 1 ^\circ C$
- Energy Battery is limited
  - Pace maker battery is not rechargeable
  - One AAA battery is 1300 ... 5000 Joule
- How much crypto in one micro Joule?
Past: efficiency power - Rijndael

- HW and SW ‘friendly’
- Rijndael AES
- Enc + Dec
- 0.18 µm CMOS
- Standard cells
- 2.3 Gbits/sec
- **Only** 56 mW

[JSSC 2003]

Current: security for embedded system

**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*

Design for efficiency AND security

SEMA attack: Simple Electromagnetic Attack on Elliptic Curve
Public Key implementation.

[E. Demulder EUROCON 2005]

Insecure implementation

**Elliptic Curve Public Key**
“Point Multiplication” algorithm
Top level description

In: point P, key k (W bits)
Output: Q = k.P

for j = 0 to W - 1
  Q = 2.Q /* double */
  if (bit j of k) is 1 then
    Q = Q + P /* add */

Return Q

Timing Side-Channel
**Future – UC Berkeley SWARM LAB**

- Terabits, Long lasting Security
- Sensory swarm
- Lightweight HW entangled

**Future: security for immersed system:**
- Attack on the “System”
- Firewalls? There is NO “inside” versus “outside”
- Encryption, trust, security “immersed”
- Devices cooperate to build up trust
- All old requirements still stand

**New Model (also simplified view):**
- Attack on the “System”
- Firewalls? There is NO “inside” versus “outside”
- Encryption, trust, security “immersed”
- Devices cooperate to build up trust
- All old requirements still stand

**HOW? Design methodology!**

*Security is a strong as weakest link!*

- Protocol: low power authentication protocol design
- Algorithm: public key, secret key, hash algorithms
- Architecture: Co-design, HW/SW, SOC
- Micro-Architecture: co-processor design
- Circuit: Circuit techniques to combat side channel analysis

**Design method - Security Partitioning**

- Server
- Client
- Noncritical software
- Architecture-level validation
- Microarchitecture-level validation
- Root-of-trust
- Crypto HW
- Crypto SW
- Software driver
- Architecture-level attacks
- Microarchitecture-level attacks
- DPA-resistant HW

**Source:** J. Rabaey: A Brand New Wireless Day
• Root of trust

• Focus of this Athens week
• Hardware security
• Software security (assumes HW is secure)
• Trusted platform module
• Trust assessment
  • …

### Program this week

<table>
<thead>
<tr>
<th>Topic</th>
<th>Monday, Nov 18</th>
<th>Tuesday, Nov 19</th>
<th>Wednesday, Nov 20</th>
<th>Thursday, Nov 21</th>
<th>Friday, Nov 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Hardware security</td>
<td>Software security</td>
<td>Trusted Platforms</td>
<td>Trusted Platforms</td>
<td>Trusted Platforms</td>
</tr>
<tr>
<td></td>
<td>Auditorium 2de hoofdwet</td>
<td>Auditorium 2de hoofdwet</td>
<td>Auditorium Kastel</td>
<td>Auditorium Kastel</td>
<td>Auditorium Kas tel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>8:30-10:15</td>
<td>8:30-10:15</td>
<td>8:30-10:15</td>
<td>8:30-10:15</td>
<td>8:30-10:15</td>
</tr>
<tr>
<td></td>
<td>HW security session 1: Ingrid Verbauwhede (start at 9:00)</td>
<td>SW security session 1: Frank Piessens, Nick Nikiforakis</td>
<td>Trusted Platforms – Dave Grawrock</td>
<td>Trusted Platforms – Dave Grawrock</td>
<td>Trusted Platforms – Dave Grawrock</td>
</tr>
<tr>
<td></td>
<td>10:15-10:30</td>
<td>10:15-10:30</td>
<td>10:15-10:30</td>
<td>10:15-10:30</td>
<td>10:15-10:30</td>
</tr>
<tr>
<td></td>
<td>Break</td>
<td>Break</td>
<td>Break</td>
<td>Break</td>
<td>Break</td>
</tr>
<tr>
<td></td>
<td>10:30-12:15</td>
<td>10:30-12:15</td>
<td>10:30-12:15</td>
<td>10:30-12:15</td>
<td>10:30-12:15</td>
</tr>
<tr>
<td></td>
<td>Lunch</td>
<td>Lunch</td>
<td>Lunch</td>
<td>Lunch</td>
<td>Lunch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Athens students</td>
<td>Exercise session on HW security (SCA): Benedikt Gierlichs</td>
<td>Exercise session 1 on TC: Dave and Roen</td>
<td>Exercise session on SW security:</td>
<td>Discussion and reception</td>
<td></td>
</tr>
<tr>
<td>Group 1</td>
<td>Room &amp; time: PC room CS dept - TBD</td>
<td>PC room ESAT - 02:53 - 13:30-16:00</td>
<td>PC room CS dept</td>
<td>Auditorium 2de hoofdwet - 14:00-16:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Athens students</td>
<td>Exercise session on HW security (SCA): Group 2</td>
<td>Exercise session 2 on TC: Dave and Roen</td>
<td>Exercise session 1 on TC: Dave and Roen</td>
<td>Exercise session 2 on TC: Dave and Roen</td>
<td></td>
</tr>
<tr>
<td>Group 2</td>
<td>Room &amp; time: PC room ESAT - 02:53-13:30-16:00</td>
<td>PC room CS dept</td>
<td>PC room ESAT - 02:53-13:30-16:00</td>
<td>PC room ESAT - 02:53-13:30-16:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Athens students</td>
<td>Exercise session on HW security (SCA): Group 2</td>
<td>Exercise session 2 on TC:</td>
<td>Exercise session 2 on TC:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 3</td>
<td>Room &amp; time:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes to the program

• Group 1: Athens students + capita selecta
• Group 2: H05B9 – Advanced architectures students
• Morning: lectures
  – Mo/Tue/Wed: 2de hoofdwet
  – Thu/Fri: Auditorium of Arenberg castle
  – Start at 8:30 am Tue-Fri
• Afternoon: exercise sessions
  – Mo/Wed: CS department
  – Tue/Thu: ESAT department

### Instructors

• Prof. Ingrid Verbauwhede – ESAT-COSIC
• Prof. Frank Piessens – CS – Distrinet
• Dr. Nick Nikiforakis – CS- Distrinet
• Dr. Dave Grawrock – senior security architect Intel
• Dr. Benedikt Gierlichs – ESAT-COSIC
• Frank Mennes – Vasco
• Ruan de Clercq & Dr. Dries Schellekens – ESAT-COSIC
Course material

- Website:
  www.cosic.esat.kuleuven.be/secure-embedded-systems/athens13/

- Practical information
- Location of exercise rooms
- Handouts

Please consult website on regular basis!!

Protect root of trust

Security: root of trust & isolation

Design tasks

- Secure storage of keys and other sensitive material
- Secure & efficient execution of cryptographic algorithms
  - During execution, no leak of information and protection against attacks
- Support for SW security functions
  - Secure SW execution
- Security building blocks:
  - RNG = Random Number Generation
  - PUF = Physical Uncloneable Function
- Security protocols: correct usage of HW and SW modules
When Hardware design?

- Fast
- Small
- Low power
- Secure
- (Analog, RF)

Design Parameters

Embedded security:
Area, delay, power, energy, physical security

HW - SW continuum

When Hardware design?

HW-SW continuum

<table>
<thead>
<tr>
<th>HW</th>
<th>HW-SW</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>FPGA</td>
<td>Domain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specific</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VLIW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>General</td>
</tr>
<tr>
<td></td>
<td></td>
<td>purpose</td>
</tr>
<tr>
<td>High</td>
<td>Area efficiency</td>
<td>Intel AES-NI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Westmere</td>
</tr>
<tr>
<td>Low</td>
<td>Performance/Energy unit</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Programmability</td>
<td>High</td>
</tr>
</tbody>
</table>

KU Leuven - COSIC Athens course-25 Leuven, Nov 2013

KU Leuven - COSIC Athens course-26 Leuven, Nov 2013

KU Leuven - COSIC Athens course-27 Leuven, Nov 2013

KU Leuven - COSIC Athens course-28 Leuven, Nov 2013
Cost definition

- Area
- Time
- Power, Energy
- Physical Security
- NRE (Non Recurring Engineering) cost

Design parameters

- Speed or throughput:
  - HW: Gbits/sec or Mbits/sec/slice
  - SW: Cycles/byte, independent of clock frequency
- Area:
  - HW: mm2 (gate or transistor count)
  - SW: memory footprint
- Power or energy consumption:
  - Power (Watts) for cooling or transmission (RFID)
  - Energy (Joule): battery operated devices
- Security: difficult to measure, but we want it
  - Entropy, leakage functions?
  - Measurements until disclosure?

Throughput: Real-time

- Extremely high throughput (Radar or fiber optics)
  - One operator (= hardware unit, e.g. adder, shifter, register)
  - for each operation (= algorithmic, e.g. addition, multiplication, delay)

  clock frequency = sample frequency

- Most designs: time multiplexing

  clock frequency $\neq$ sample frequency

  clock frequency = number of clock cycles available for the job
  sample frequency

Power density will increase

Cooling!!

Power density too high to keep junctions at low temp

[Author: S. Borkar, Intel]
What can one do with 1 cm³?

**Energy Storage**

<table>
<thead>
<tr>
<th>J/cm³</th>
<th>W/cm³/year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro fuel cell</td>
<td>3500</td>
</tr>
<tr>
<td>Primary battery</td>
<td>2880</td>
</tr>
<tr>
<td>Secondary battery</td>
<td>1080</td>
</tr>
<tr>
<td>Ultra-capacitor</td>
<td>100</td>
</tr>
</tbody>
</table>

One AAA battery: 1300 to 5000 Joule

---

**Power and Energy are not the same!**

- **Power** = \( P = I \times V \) (current \( I \) x voltage \( V \)) (= Watt)
  - instantaneous
  - Typically checked for cooling or for peak performance
- **Energy** = **Power** \( \times \) execution time (= Joule)
  - Battery content is expressed in Joules
  - Gives idea of how much Joules to get the job done

Low power processor ≠ low energy solution!

---

**Heat and parallelism**

Reduce power = reduce WASTE!!

**Power** (Heat)

\[
P_{\text{mono}} = CV^2f \quad (\text{Watt})
\]

\[
4 \left(\frac{C}{4}\right)^2f \left(\frac{f}{4}\right) = P_{\text{mono}}/4
\]

but since \( f \sim V \)

can be even \( P_{\text{mono}}/4^3 \)

---

**What is hardware design?**

**TREND: MULTI-CORE!!**
Type 1: ASIC design

Standard cell based design

Semicustom Design Flow

Logic Design Activities

• Logic and FSM synthesis
  – State minim., coding
  – Multilevel Logic Optimisation

• Technology Mapping
  – Functions to library cells
  – Minimal Area for given delay

• Timing Verification
  – Estimate wiring load C
  – Critical logic path

• Layout
  – P&R C extraction from wiring ...

Cell-based Design (or standard cells)
Standard Cell Layout

- Std. Cell Place & Route (RT-Module)
- Std. Cell
- Routing Channel
- Cell Row

(Courtesy: Tanner Tools)

Standard Cell Zoom In

Module Generation

- For data-path operators: structure is in bit-slices
- Computer generated layout as function of wordlength
- Compact, predictable IP

Instruction, Clock

Power

Data

Cell-structure hidden under interconnect layers
Standard Cell and Module

Type 2: FPGA design

Look-up Table Based Logic Cell

LUT-Based Logic Cell

Xilinx 4000 Series
Not most up to date

Courtesy Xilinx
RAM-based FPGA

FPGA benchmarks??

- Area numbers:
  - Slices, LUT’s, CLB’s, …
  - Xilinx application engineer: “The number of CLB’s inside LUT’s changes from generation to generation.” (or was it LUT’s inside CLB’s?)
- Speed: accurately reported by tools
- Power:
  - Poorly reporting by tools
  - Hard to measure on board

Start with easy one:
Block cipher - DES
Design for efficiency
Symmetric key: DES

- DES = Data Encryption Standard
- FIPS Standard 46 effective in July 1977: US government standard for sensitive but unclassified data
- July 26, 2004: FIPS 46-3 is withdrawn: use TDEA or AES
- TDEA = Triple DES encryption algorithm – NIST 800-67

TDEA

- Three Key options:
  - K1, K2, K3 different
  - K1=K3, K2 different
  - K1=K2=K3, backward compatible with single DES
- Two-key triple DES: until 2009
- Three-key triple DES: until 2030

DES = Feistel cipher

- DES has 16 rounds + initial and final permutation
- Basic cipher structure is Feistel cipher
  - other examples of Feistel: IDEA, FEAL, Kasumi

DES- $f$ function

- 32b-to-48b permutation (wiring & bit duplication)
- Input of S-boxes: 8x6b
- $S_i = 6b$-to-$4b$ non linear substitution (ROM or logic based Look up table)
- Output of S-boxes: 8x4b
- 32b-to-32b permutation (wiring)

- Hardware: encryption = decryption! (different for AES)
- Because of Feistel: no need for $f^{-1}$ (different for AES)
DES Key schedule

Initial key $K$

PC1: permute and drop 8 bits

C&D: rotate left 1 or 2 bits each round

DECRIPTION: rotate right

PC2: permute and select 48 output bits

Round Key $K_i$

C&D left/right shift registers: encryption & decryption HW

Key Schedule

Two options:
- On the “fly” = just in time processing
- Pre-compute and store

Typical for Hardware

Key Schedule

Memory

Typical for Software

Key schedule on the fly

- The cost of fast key context switching:
- Example for IPSEC router
  - one 128 bit key = 1408 bits round keys (10 rounds + initial key)
  - half of internet packets are only 64 bytes in length (512 bits)

BANDWIDTH PROBLEM!
AES: Byte substitution

- Byte substitution: each byte individual
- 16 identical Sboxes
- Area - time trade-off: HW multiplexing
- 32 for Rijndael

AES: Shiftrow

- Shiftrow: circularly rotate each row of state array
- Easy wiring

AES: mix column

- matrix multiplication of state array columns
- multiply with constant entries

Mix column - encryption

Mix Column Operation is $GF(2^8)$ Linear Transform

+ $GF(B)$

GF(B x 1)

GF(B x 3)

GF(B x 2)
Key schedule

- Unit is 32 bit words, W[i] = 32 bit = 1 column
- 4 different operations
- One round key is four W[i]'s

Key schedule

- Encryption key
  - HW: on the fly: round key[i] = function (round key[i-1])
  - SW: pre-compute and store in context (176, 208, 240Bytes)
- Decryption key
  - encryption key in reverse order
  - BUT need final round words to start

Combined AES architecture

- AES is not Feistel
- Every operation has its inverse

Sub modules

- Reason that decryption is slower
**Sbox optimization**

- $\text{GF}(2^8)^4$ requires large Look up tables
- Map to isomorphic fields, $\text{GF}((2^4)^2)$ or $\text{GF}(((2^2)^2)^2)$ and invert there

**Sbox experiment**

- 0.18 $\mu$m CMOS, Synopsys experiment
- size of 1 Sbox, push for area or for speed

**Compact SBOX**

- $\text{GF}(((2^2)^2)^2)$ instead of $\text{GF}(2^8)$
- Reduces the gate count to only 280 gates!
  [size depends on the choice of $\lambda$.]

  [Mentens et al., RSA 2005]

**Ballpark numbers**

- 1 gate = 2input NAND gate = 4 transistors
- Sbox size:
  - $\text{GF}(2^8)^4 = 650$ to 700 gates
  - $\text{GF}((2^4)^2)^4 = 400$ gates ([Wo02])
  - $\text{GF}(((2^2)^2)^2)^4 = 280$ gates ( [Sat01][Men05])
  - but 50 to 100% slower
- AES core encryption only: 20K to 25K gates
  - 128bit data, 128 bit key
  - key schedule on the fly
  - 1 clock cycle per round
- AES core for encryption and decryption: 40Kgates
  - 128 bit data, 128 bit key
  - precompute and store round keys: 128x11bits SRAM
  - 1 clock cycle per round
  - **savings in combining logic, losses in multiplexers!**
- Rijndael
- Enc + Dec
- 0.18um CMOS
- Standard cells

AES, 2nd generation
- Regular & WDDL based implementation
- Standard cells
- 0.18 um CMOS

### Throughput – Energy numbers

<table>
<thead>
<tr>
<th></th>
<th>Throughput</th>
<th>Power</th>
<th>Figure of Merit (Gb/s/W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES 128bit key</td>
<td>3.84 Gbit/sec</td>
<td>350 mW</td>
<td>11 (1/1)</td>
</tr>
<tr>
<td>128bit data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.18m CMOS</td>
<td>3.84 Gbit/sec</td>
<td>350 mW</td>
<td>11 (1/1)</td>
</tr>
<tr>
<td>FPGA[1]</td>
<td>1.32 Gbit/sec</td>
<td>490 mW</td>
<td>2.7 (1/4)</td>
</tr>
<tr>
<td>Intel ISA for AES</td>
<td>32 Gbit/sec</td>
<td>95 W</td>
<td>0.34 (1/33)</td>
</tr>
<tr>
<td>ASM StrongARM [2]</td>
<td>31 Mbit/sec</td>
<td>240 mW</td>
<td>0.13 (1/85)</td>
</tr>
<tr>
<td>Asm Pentium III [3]</td>
<td>648 Mbits/sec</td>
<td>41.4 W</td>
<td>0.015 (1/800)</td>
</tr>
<tr>
<td>C Emb. Sparc [4]</td>
<td>133 Kbits/sec</td>
<td>120 mW</td>
<td>0.0011 (1/10,000)</td>
</tr>
<tr>
<td>Java [5] Emb. Sparc</td>
<td>450 bits/sec</td>
<td>120 mW</td>
<td>0.0000037 (1/3,000,000)</td>
</tr>
</tbody>
</table>

[1] Amphion CS5230 on Virtex2 + Xilinx Virtex2 Power Estimator
[4] gcc, 1 mW/MHz @ 120 Mhz Sparc – assumes 0.25 u CMOS
[5] Java on KVM (Sun J2ME, non-JIT) on 1 mW/MHz @ 120 MHz Sparc – assumes 0.25 u CMOS

### Match between algorithm & architecture

Close the gap:
- Dedicated HW: ASIC
- Programmable HW: FPGA
- Custom instructions, hand-coded assembly
- Compiled code
- JAVA on virtual machine, compiled on a real machine
Conclusions

• Trust in embedded devices
  = HW security
  + SW security
  + Security partitioning and isolation
• Needs a light weight root of trust