Physical(ly) Unclonable Functions

An introduction to Intrinsic PUFs

Ingrid Verbauwhede

Slide courtesy: Roel Maes
COSIC – K.U.Leuven
Supported by:
IWT and unique

---

Introduction

Goal of this talk → try to answer the question:

- What is a PUF?
- What is it usage?
- Can we use it as an unclonable device identifier?

- P.U.F.?
  - P.U.F. → Physical Unclonable Function
    a physical function which is unclonable in every sense
  - P.U.F. → Physically Unclonable Function
    a function which is unclonable in a physical sense
Introduction (cont.)

- Need for unique identification of devices or goods
  - Example: RFID tags
  - Secure storage of a key bit string (non volatile memory, battery backed up SRAM, fuses, etc.)

- Problem: personalization step during fabrication
  - Expensive, extra processing steps
  - Post-processing e.g. by blowing fuses

- Idea: use physical uniqueness of devices
- Idea: use CMOS process variations for this
  - Threshold voltage
  - Oxide thickness
  - Metal line shapes

Functional description of PUF

- For use in crypto applications
- PUF = (physical) function which is physically unclonable

- Very hard (“impossible”) to produce two PUFs with similar challenge-response behavior
- Easy to construct and evaluate a random PUF
Basic properties of PUF

Minimum requirements:

- For two random PUFs, difference between expected responses to same challenge, should be large
  - = manufacturing variability is ‘large’

- For single random PUF, difference between two measured responses to same challenge, should be small
  - = noise, aging, temperature effects,… are limited

- For single random PUF, uncertainty about response to challenge is large, when one does not have access to this PUF instance
  - = unpredictability is large

Intrinsic PUFs

- Use inherent manufacturing variability present in CMOS fabrication
- Keep measured PUF responses inside chip
  - Useful for secure key storage!

Requirements:

- PUF + measurement circuit + post-processing inside chip
- Standard processing, no extra processing steps
MOS Transistor

- Gate voltage below “threshold” - No current (there is subthreshold current)
- Gate voltage above “threshold” - current can flow
- Threshold = f(doping concentration)

Delay of gate

- Time to charge or discharge capacitances at output
- Delay $t = C \times \Delta V / I$
- $I = f (C_{ox}, (V_{GS} - V_{Th}))$
- Process variations:
  - Oxide thickness
  - Threshold voltage
  - Capacitance

0-1 transition
Outline

- Introduction
- A few examples of Intrinsic PUFs
- PUF properties
- PUF applications
- Conclusion

First example: Arbiter PUF

*Delay based intrinsic PUF*
**Arbiter PUF: basic operation**

- Initial design [Lee et al, MIT 2004]
  - switch block: e.g. two muxes
  - arbiter: e.g. a latch or a flip-flop
  - $n$ switch blocks $\rightarrow 2^n$ “different” delays

```
0 1 0 0 1 1
```

**Arbiter PUF: experiments [Lee04]**

- Results:
  - 10000 CRPs from 37 ASICs
  - 64-stage arbiter PUF

```
Inter-chip variation (%)
```

- Results on FPGA [Lee at al 04]: $\mu_{\text{inter}} = 1.05\%$, $\mu_{\text{intra}} = 0.3\%$
Arbiter PUF: analysis

- delay ≈ additive!
  - leads to model-building attack (linear programming)
  - also machine-learning techniques (Artificial Neural Networks, Support Vector Machines, …)

- Attack results:
  - ASIC: 3.55% prediction error with SVM trained with 5000 CRPs ($< \mu_{\text{intra}}$ !)
  - FPGA: 0.6% prediction error with perceptron trained with 90000 CRPs

- Extensions [Ozturk et al 2008]
  - tri-state buffer based delay circuit → comparable to switch-based
  - Simulation: prediction error < 3% for linear programming on 4000 CRPs

- Conclusion: (Improved) arbiter PUFs can be accurately modelled (= “cloned”) from polynomial # known CRPs

Second example: Ring Oscillator PUF

Delay based Intrinsic PUF
Ring Oscillator PUF: basic operation

- Initial design [Gassend et al 2003]

![Diagram of Ring Oscillator PUF](image)

- Compensation
  - To eliminate (scaling) environmental influence

- One-bit compensation [Suh 2007]
  - To avoid costly division

Response \( \sim f \)

Delay \( f \) → Edge detector → Counter ++ → Response

\[ r = \frac{f_A}{f_B} \]

\( f_A < f_B \) \( \Rightarrow \) 0
\( f_A \geq f_B \) \( \Rightarrow \) 1

Ring Oscillator PUF: experiments

- Results [Suh2007]:
  - measurements on 15 FPGAs, 1024 loops/FPGA and 1 out of 8 masking

\[ \mu_{\text{inter}} = 46.15\% \]
\[ \mu_{\text{intra}} = 0.48\% \text{ (with temp./volt. var.)} \]
Ring Oscillator PUF: analysis

- Modelling attacks?
  - same as arbiter PUFs for basic design
  - not for fixed delay circuit as in [Suh 2007]
    → but much less challenges!
    \[ \frac{N}{2} < \text{#challenges} < \log_2(N!) \]
    → inefficient area use

SRAM PUF

Memory based Intrinsic PUF
SRAM PUF: basic operation

- **SRAM cell:** (6T-CMOS)

- Which state right after power-up?
  - depends on physical **mismatch** between $M_2$ and $M_4$
  - power-up state = measure for **manufacturing variability**
  - 2 possible stable states → 1-bit storage

---

SRAM PUF: basic operation

- **SRAM PUF**
  - **challenge** = SRAM address
  - **response** = power-up state of addressed cell(s)

- **Guajardo et al 2007**
  - SRAM on FPGA

- **Holcomb et al 2007**
  - COTS SRAM
  - SRAM on embedded micro-controller

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$\bar{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
SRAM PUF: experiments

- Results [Guarjado et al CHES 2007]
  - measurements on FPGA, 8190 bytes from different SRAM blocks
  - $\mu_{\text{intra}} = 3.57\%$, $\sigma_{\text{intra}} = 0.13\%$
  - $\mu_{\text{inter}} = 49.97\%$, $\sigma_{\text{inter}} = 0.3\%$

- Results [Holcomb, Burleson, Fu 2009]
  - measurements on two types of devices
  - COTS SRAM chip: 5120 64-bit blocks on 8 ICs
  - Embedded SRAM in µC: 15 64-bit blocks on 3 ICs
  - $\mu_{\text{inter}} = 43.16\%$, $\mu_{\text{intra}} = 3.8\%$
  - $\mu_{\text{inter}} = 49.34\%$, $\mu_{\text{intra}} = 6.5\%$
Outline

- Introduction
- Examples of Intrinsic PUFs
- **PUF properties**
- PUF applications
- Conclusion

Quick overview

- Evaluable: \( y = \text{PUF}(x) \) is easy
- Unique: PUF(x) contains some unique information
- Reproducible: PUF() has only small error
- Unclonable: hard to make PUF’(x) given PUF(x)
- Unpredictable: hard to find \( y_N = \text{PUF}(x_N) \) given other x, y pairs
- One-way: given y and PUF(), cannot find x
- Tamper evident: tampering changes PUF()
Compare PUF constructions

1. **OPT**
   - **Challenge**: laser orientation
   - **Response**: Gabor hash of speckle pattern

2. **COAT**
   - **Challenge**: sensor pair selection
   - **Response**: quantized capacitance measurement

3. **ARB**
   - **Challenge**: switch-block delay setting
   - **Response**: one-bit arbiter decision

4. **FF-ARB**
   - **Challenge**: (reduced) switch-block delay setting
   - **Response**: one-bit arbiter decision

5. **LW-ARB**
   - **Challenge**: pre-challenge to input-network
   - **Response**: one-bit XOR of arbiter decisions

6. **RO**
   - **Challenge**: switch-block delay setting
   - **Response**: ratio of two counter values

7. **18-RO**
   - **Challenge**: oscillator pair selection
   - **Response**: one-bit comparator output

8. **SRAM/FF**
   - **Challenge**: SRAM cell address
   - **Response**: one-bit power-up state of addressed cell

9. **LATCH/BUTTER**
   - **Challenge**: latch/butterfly cell selection
   - **Response**: one-bit settling state of excited cell

10. **CPUF**
    - **Challenge**: pre-PUF hash input
    - **Response**: post-PUF hash output

11. **ID**
    - **Challenge**: ask identifier
    - **Response**: identifier value

12. **PK**
    - **Challenge**: random nonce
    - **Response**: signature on nonce

13. **TRNG**
    - **Challenge**: ask true random number
    - **Response**: “physically produced” true random number

**PUF properties table**

<table>
<thead>
<tr>
<th>OPT</th>
<th>COAT</th>
<th>ARB</th>
<th>FF-ARB</th>
<th>LW-ARB</th>
<th>RO</th>
<th>18-RO</th>
<th>SRAM/FF</th>
<th>LATCH/BUTTER</th>
<th>CPUF</th>
<th>ID</th>
<th>PK</th>
<th>TRNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval.</td>
<td>Mechan.</td>
<td>Fusion</td>
<td>Integrity</td>
<td>Fusion</td>
<td>Integrity</td>
<td>Integrity</td>
<td>Integrity</td>
<td>Integrity</td>
<td>Integrity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Unique</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Reprod.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Phys. Unclon.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Math. Unclon.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>One-way</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Unpred.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tamper Evident</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

ECRYPT, ALBENA, MAY 2011

13
**Outline**

- Introduction
- Examples of Intrinsic PUFs
- PUF properties
- **PUF applications**
- Conclusion

---

**PUF Applications**

- System identification
  - anti-counterfeiting
  - hardware binding
  - hardware metering (passive)
- Secret Key Generation
  - secure key storage
  - secure key distribution
    \[ \rightarrow \text{key-based crypto} \rightarrow \ldots \]
- Hardware Entangled Cryptography
  - side-channel resistance \(\rightarrow\) provable physical security?
System Identification

- Very similar to biometrical identification systems
  - Easy, fast, straightforward, but limited security
    - e.g. no authentication

Key Generation

- Integration of PUFs with \textit{keyed} crypto primitives
- Key extraction
  - “non-volatile” key storage in a \textit{“non-digital”} way
    - key only digitally present when needed in crypto computation
      → no long-term digital storage required
  - unique key material \textit{“intrinsically”} present
    - no key programming step required
      → however, two-phase key generation: enrollment - reproduction
Hardware Entangled Cryptography

- Embedded integration of PUFs in crypto primitives

- Secret parameter = device-unique PUF behavior
  - no digital key present at any point
  - no digital key-storage required whatsoever
  - Basic complexity/cryptographic assumptions about PUFs are needed

- No digital key → constrains application scenarios!

- Topic of active research

Conclusion

- “What is a PUF?” is not an easy question
- We identified some minimal requirements which
  - ...are common to all known PUF constructions (exhaustively)
  - ..., but distinguish them from other primitives
- “Which is the best PUF?” is an even harder question
  - there are contradicting goals → trade-offs
  - application dependent
- Still many open questions...

For an extended reference list, see website Roel Maes:

Future work: “situating PUFs”

uniqueness vs reproducibility ($\mu_{\text{inter}} / \mu_{\text{intra}}$)

- $\mu_{\text{intra}} = 0$
- $\mu_{\text{inter}} > \mu_{\text{intra}}$
- $\mu_{\text{inter}} < \mu_{\text{intra}}$

hard-programmed keys
PUFs
TRNGs
practically physically clonable
practical physical cloning is hard
physically unclonable
physically unclonable in practice
physically unclonable

References

- [GCVD02] Blaise Gassend and Dwaine Clarke and Marten van Dijk and Srinivas Devadas, "Controlled Physical Random Functions", ACSCAC, 2002
- [GCVD02b] Blaise Gassend and Dwaine Clarke and Marten van Dijk and Srinivas Devadas, "Silicon physical random functions", ACM CCS, 2002
- [LLGVD04] Daihyun Lim and Jae W. Lee and Blaise Gassend and G. Edward Suh and Marten van Dijk and Srinivas Devadas, "A technique to build a secret key in integrated circuits for identification and authentication application", Symposium on VLSI Circuits, 2004
- [GCVD04] Blaise Gassend and Daihyun Lim and Dwaine Clarke and Marten van Dijk and Srinivas Devadas, "Extracting secret keys from integrated circuits", Master's Thesis MIT, 2004
- [ST06] B. Skoric and P. Tuyls and W. Ophey, "Robust key extraction from Physical Uncloneable Functions", ACM (LNCS 3531), 2005
- [TSVW06] Pim Tuyls and Geert-Jan Schrijen and Boris Skoric and Jan van Geelven and Nynke Verhaegh and Rob Wolter, "Read-proof hardware from protective coatings", CHES, 2006
- [TS06] Pim Tuyls and Boris Skoric, "Physical Uncloneable Functions for enhanced security of tokens and tags", ESSP, 2006
References