PUFs: Trust Anchors for Hardware Intrinsic Security

Vincent van der Leest
Intrinsic-ID, Eindhoven (NL)

July 15, 2014

TRUDEVICE, Lisbon
Roots of Trust

Information Security Objectives

Data security
Entity Authentication

Crypto Primitives
Symmetric Ciphers
Hash / MAC
Public Key Crypto
Protocols

Execution Primitives
Key Storage
Secure Computation
Randomness Generation

PUFs
TRNGs
Secure Logic
Shielded Storage
Intrusion Detection
Logistic Control

Physical Primitives
Physical Key Storage

- Alternative to NVM-based key storage: PUF-based key storage
- Main advantages:
  - Key not present when device is powered down
  - Key depends on device intrinsic randomness
**PUFs: Physically Unclonable Functions**

- PUFs are more like fingerprints than like programmed keys:

<table>
<thead>
<tr>
<th>Human Fingerprint</th>
<th>PUF</th>
<th>Programmed Key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unique</strong> per person</td>
<td><strong>Unique</strong> per device</td>
<td><strong>No guarantee</strong> of uniqueness</td>
</tr>
<tr>
<td><strong>Inherent</strong> from birth</td>
<td><strong>Inherent</strong> from production</td>
<td></td>
</tr>
<tr>
<td>Impossible to “clone” humans with the same fingerprints</td>
<td>Infeasible to “clone” devices with the same PUF</td>
<td>Easy to program many devices with the same key</td>
</tr>
</tbody>
</table>
Different PUF types: Optical PUF

Pro’s
Huge set of C/R-pairs

Con’s
Difficult to integrate in IC
Different PUF types: Coating PUF

<table>
<thead>
<tr>
<th>Pro’s</th>
<th>Con’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part of IC</td>
<td>Expensive to produce</td>
</tr>
<tr>
<td></td>
<td>Limited set of C/R-pairs</td>
</tr>
</tbody>
</table>
## Different PUF types: Delay based PUFs

### Pro’s
- Part of IC
- Relatively large set of C/R-pairs

### Con’s
- Place and Route constraints due to non-standard components
- Susceptible to modeling attacks

**Diagram:**
- Challenge → Combinatorial circuit → Response
Different PUF types: Memory-based PUF

<table>
<thead>
<tr>
<th><strong>Pro’s</strong></th>
<th><strong>Con’s</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructed from standard CMOS components</td>
<td>Limited set of C/R-pairs</td>
</tr>
</tbody>
</table>
Basic PUF properties: Reliability

~ 10% errors
Basic PUF properties: Uniqueness

Device 1

Device 2

~ 50% difference
Basic PUF properties: “Physical Unclonability”

- Technical infeasibility/impossibility to create “non-unique” PUF instantiations
  - Due to *uncontrollable* random process variations
Silicon PUF-based applications

• Device identification

• Device authentication
  • Some variant of:
    - Cryptographic key generation

PUF response = device ID

PUF response = authentication secret

PUF challenge

CRYPTO: Encryption, Signing, Key wrapping, ...

Embedded on chip

Key Generator

PUF response = “static” source of entropy for key generation
Key generation/storage with Silicon PUFs

- Discrepancy between PUF response and crypto key:
  - Reproducible:
    e.g. 3% intra-distance
  - Unpredictable:
    e.g. 70% entropy

- Key Generator:
  1. Improves *reproducibility* by taking care of intra-distance of response = *correct bit errors*
  2. Improves *unpredictability* by extracting unpredictable part of response = *compress & accumulate entropy*
Enrollment and reconstruction for key storage

**Enrollment**

C → PUF → R → Helper Data Algorithm → W → Key

One-Time Process

**Reconstruction**

C → PUF → R’ → Helper Data Algorithm → Key

In the field

I(W, Key) < ε  P[Key not Correct] < δ
Practical PUF-based key generators

- To give you some idea of realistic systems (from literature):
  - All for **128-bit keys**:

<table>
<thead>
<tr>
<th>PUF type</th>
<th>PUF size</th>
<th>PUF error rate</th>
<th>Error Correction</th>
<th>Key failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boesch et al., [CHES-2008]</td>
<td>SRAM PUF</td>
<td>3696 bits</td>
<td>Repetition + Golay (hard decision)</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>Maes et al., [CHES-2009]</td>
<td>SRAM PUF</td>
<td>1536 bits</td>
<td>Repetition + Reed-Muller (soft decision, multi enroll)</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>van der Leest et al., [CHES-2012]</td>
<td>SRAM PUF</td>
<td>2880 bits</td>
<td>Repetition + Golay (soft decision, single enroll)</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>Maes et al., [CHES-2012]</td>
<td>Ring Oscillator PUF</td>
<td>848 oscillators</td>
<td>Repetition + BCH (hard decision)</td>
<td>$10^{-9}$</td>
</tr>
</tbody>
</table>

- **PUF error rate significantly affects error correction and PUF size**
  - Key failure rate has less impact
Recent Developments: Attacks on PUFs

• Physical Attacks on PUFs
  • PUFs, like all physical crypto primitives, can be susceptible to physical attacks

• E.g.
  • EM analysis on ring oscillator PUFs
  • Remanence decay attack on SRAM PUFs
  • Photon Emission Analysis (PEA) on SRAM PUFs
  • Invasive attacks

• It is important to learn from these attacks, because countermeasures are possible and should be implemented with PUFs
Recent Developments: Aging and Anti-Aging (SRAM PUFs)

• **SRAM PUF “natural aging”**
  - Power-up behavior: fastest transistor (of matched pair) closes first
  - NBTI aging: closed transistors become slower over time
  - Result: power-up behavior changes over time hence: # bit errors increases over time

• **SRAM PUF “anti-”aging**
  - Long-term storage of the power-up state *inverse* reinforces the power-up behavior
  - Result: # bit errors **decreases** over time! [Maes et al., HOST 2014]
  - A similar effect (HCI) can also be applied in an accelerated manner immediately after production to improve the reliability of an SRAM PUF from the start [Bhargava et al, CHES 2013]
Designing products with PUFs

- More than simply providing PUF implementations
- Offering higher security by replacing NVM with PUFs is not enough to convince chip vendors to make changes
- How can PUF-based security be used to increase the value of devices?

- Quality and focus of IP design are incredibly important:
  - Secure design (design may never introduce weaknesses)
  - Reliability (regarding environment and application requirements)
  - Flexibility (customer requirements, easy integration, etc.)
  - Application oriented (no application → no customers)
Secure Design

To ensure the design is secure:

• Include countermeasures against known (and possibly future) attacks, taking into account e.g. side-channel, fault, and invasive attacks

• Apply “layered” security design (e.g. using obfuscation)

• Implement secure BIST

• **PUF response not accessible for untrusted parties**

Besides secure design, other parameters of importance:

• Area required by PUF implementation

• Power consumption

• Timing requirements
Reliability

Quality of design should be such that it works reliably...

- under different ambient conditions (temperature, supply voltage, power-up curves, ...)
- during lifetime of chip (handle CMOS aging effects)

Reliability should also be ensured in regard to requirements defined by applications, such as:

- Being able to recover from reset
- Performing key reconstruction when required (without repower or reset)
Examples of Reliability Testing at Intrinsic-ID

- Tested: 180, 150, 130, 90, 65, 45, 40, 28, 14 nm
- Temperature cycle / temperature ramp
- Endurance low temperature: IEC 60068-2-1
- Endurance high temperature: IEC 60068-2-2
- Radio frequency electromagnetic field: IEC 61000-4-3
- Ambient electromagnetic fields immunity: EMC: EN55020
- Electromagnetic compatibility
- Humidity
- Voltage ramp-up
- Data retention voltage
- Accelerated lifetime
- Extensive End customer validation
- Millions of measurements performed
Flexibility

- Only use standard CMOS components
- Small silicon area and/or software footprint
- Provide fast-track implementations on ASIC, FPGA and SW
- Provide high-level API for easy porting of security apps

Provide flexible key management:
- Key programming by any party in the value chain
- Unlimited number of keys
- Enrollment of keys can be done on-chip or off-chip
Designing towards PUF-based applications

- **R&D into PUF constructions and PUF-based key generators**
- **Integration of PUF-based key generators with crypto/security (e.g. key management, encryption, ...)**
- **Development of end-user applications using PUF-enabled crypto/security (e.g. secure cloud storage, ...)**
Summary

- **A silicon PUF** is a *process variation dependent circuit* → effectively a “**device fingerprint**”
  - **Delay-based** constructions: arbiter PUF, ring oscillator PUF, ...
  - **Memory-based** constructions: SRAM PUF, D flip-flop PUF, ...
  - “**Physically unclonable**”: process variations are beyond manufacturer’s control

- PUFs are typically *noisy* and *biased*, crypto keys are not...
  → PUF-based key generator: **PUF → KeyGen → Crypto Key**
  - Improve robustness with **error-correction** techniques → **helper data**
  - Improve unpredictability with **entropy accumulation**

- Designing PUF products is more than simply implementing PUFs!
Thank you!

Any questions?