IoT Hardware Security
Using Physically Unclonable Functions (PUFs) for Authentication and Secure Key Exchange

Professor Fayez Gebali

ECE Dept.
University of Victoria
https://ece.engr.uvic.ca/~fayez/
Outline

1. Introduction
2. PUF
3. Authentication
4. Exposure
5. Examples
6. Authentication Algorithms
Introduction
Motivation: New Security Challenges

1. Hardware attacks are harmful compared to software attacks
2. Computing devices are becoming distributed, unsupervised, and physically exposed
3. Protecting software is no longer enough
4. Attackers can physically tamper with devices
5. Layered defence strategy including hardware and software
6. Secret keys stored in NVRAMs are vulnerable to hardware attacks
Example of Infrastructure IoT: Telehealth [1, 2]
PUFs
What is a PUF

1. A simple circuit on the silicon chip
2. Output of PUF is unique due to random processing variations (RPV)
3. Output is noisy due to CMOS and thermal noise
4. Knowing circuit does not help knowing I/O relation
5. Provides tamper proofing and exposure resilience
Why Use PUF

1. Give hardware a **unique ID**

2. **Tamper proofing** hardware: Hardware root-of-trust (HRoT)

3. Facilitate **authentication**: simple, context-aware, or adaptive

4. Enable **secure key exchange**

5. Ensure **secure** symmetric and public key encryption
Adding Unique ID to Each Chip Through PUF? [3]
What is a PUF?

Challenge $c$ → PUF → Response $r$
**PUF Response: Hamming Distance**

Responses are noisy
Secure Processors
Hardware Root-of-Trust (HRoT)

Definition

1. A processor that is trusted in a cryptographic system (tamper proof)

2. Include a hardened (secure) hardware accelerators for cryptographic operations

3. Has secure key storage or generation
Cryptoprocessor (Elliptic Curve Accelerator)
PUF-Based Authentication
PUF-Based Authentication: By Device Fabricator

- Place results at trusted certification authority (CA)
PUF-Based Authentication: By Device Authenticator (Server)

From CA: 
(c, r)

- **FEC encode**
  - w

- **Key Geneate**
  - k

- **Hash**
  - h

- Only c, w and x are sent to client
- Allow adaptive/context-aware authentication
**PUF-Based Authentication: By Device (Client)**

- $r$, $k$, and $h$ are generated locally at client and server.
Session Key via PUF Response Considerations

1. PUF response is a means of secure key exchange
2. Response inherently has small entropy (why?)
3. Response is inherently noisy (unreliable)
4. Session key must have large entropy
5. Forward error correction (FEC) is typically used
Exposure Resilient Cryptography
Exposure Resilient Functions: All or Nothing Transform (AoNT) [4]

Assume the function “structure” is revealed, we have:
Exposure Resilient Functions: All or Nothing Transform (AoNT) [4]

Assume the function "structure" is revealed, we have:

**Definition**

Knowing the input of the function, the output can not be estimated
Exposure Resilient Functions: *All or Nothing Transform (AoNT)* [4]

Assume the function “structure” is revealed, we have:

**Definition**

Knowing the input of the function, the output can not be estimated

**Definition**

Knowing the output of the function, the input can not be estimated
### Exposure Resilient Functions: All or Nothing Transform (AoNT) [4]

Assume the function “structure” is revealed, we have:

#### Definition

Knowing the input of the function, the output can not be estimated

#### Definition

Knowing the output of the function, the input can not be estimated

#### Definition

Duplicating the function “structure” does not reproduce the input/output relation of the original structure
Exposure Resilient Security

1. In general, no guarantee of security if any fraction of secret key is exposed

2. Using PUF, security is guaranteed even when the input/challenge is publicly issued

3. Secret keys stored in memory could be identified as areas with high entropy
PUF Examples
PUF Examples

1. SRAM PUF
2. DRAM PUF
3. Ring Oscillator PUF
4. Arbiter PUF
5. Coating PUF
<table>
<thead>
<tr>
<th></th>
<th>PUF Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SRAM PUF</td>
</tr>
<tr>
<td>2</td>
<td>DRAM PUF</td>
</tr>
<tr>
<td>3</td>
<td>Ring Oscillator PUF</td>
</tr>
<tr>
<td>4</td>
<td>Arbiter PUF</td>
</tr>
<tr>
<td>5</td>
<td>Coating PUF</td>
</tr>
</tbody>
</table>
Static RAM (SRAM): Architecture
Static RAM (SRAM): 6-Transistor Cell Structure
Proposed Static RAM (SRAM): 8-Transistor Cell Structure
## PUF Examples

1. SRAM PUF
2. DRAM PUF
3. Ring Oscillator PUF
4. Arbiter PUF
5. Coating PUF
Dynamic RAM (DRAM): Architecture

Cells → Banks → Ranks → Channels → DRAM
Dynamic RAM (DRAM): Cell Structure
Dynamic RAM (DRAM): Cell Decay Model

- Capacitor decays linearly (not RC delay anymore)
PUF Examples

1. SRAM PUF
2. DRAM PUF
3. Ring Oscillator PUF
4. Arbiter PUF
5. Coating PUF
Ring Oscillator (RO) PUF

\[(0,0) \quad (0,1) \quad (0,S-1)\]
\[f(0)\]
\[\text{upcounter}\]
\[c(0)\]
\[en\] \[rst\]

\[(1,0) \quad (1,1) \quad (1,S-1)\]
\[f(1)\]
\[\text{upcounter}\]
\[c(1)\]
\[en\] \[rst\]

\[\vdots\]

\[(W-1,0) \quad (W-1,1)\]
\[(W-1,S-1)\]
\[f(W-1)\]
\[\text{upcounter}\]
\[c(W-1)\]
\[en\] \[rst\]
Proposed Configurable Galois Ring Oscillator (RO) PUF

(a)

(b)
## PUF Examples

1. SRAM PUF
2. DRAM PUF
3. Ring Oscillator PUF
4. Arbiter PUF
5. Coating PUF
Aribter PUF
PUF Examples

1. SRAM PUF
2. DRAM PUF
3. Ring Oscillator PUF
4. Arbiter PUF
5. Coating PUF
Coating PUF
Coating PUF

$TiN/ TiO_2$ Array

Aluminium wire

Passivation layer

$SiO_2$ Insulation layer

Integrated Circuit
Algorithms
## Forward Error Correction Codes (FEC) in Telecommunications versus PUF Authentication

<table>
<thead>
<tr>
<th>Telecommunications</th>
<th>PUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender generates and sends message to receiver</td>
<td>Sender receives message. Receiver generate noisy message</td>
</tr>
<tr>
<td>Sender sends message and redundant data</td>
<td>Sender sends redundant data only</td>
</tr>
<tr>
<td>Noise generated at channel</td>
<td>Noise generated at receiver</td>
</tr>
<tr>
<td>Resend when errors are not removed</td>
<td>Fail authentication when errors are not removed</td>
</tr>
</tbody>
</table>
Proposed Authentication Algorithms

1. Alg # 1: Single Challenge

2. Alg # 2: Repeated Challenge

3. Alg # 3: Repeated Challenge with Bit Selection
Authentication Algorithms

1. Alg # 1: Single Challenge
2. Alg # 2: Repeated Challenge
3. Alg # 3: Repeated Challenge with Bit Selection
## Authentication Algorithm #1: Single Challenge

<table>
<thead>
<tr>
<th>Server</th>
<th>Channel</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select $c$, $x$</td>
<td>Generate $K$, $h$, $w$</td>
<td>$r' = \text{PUF}(c)$</td>
</tr>
<tr>
<td></td>
<td>$(c,x,w)$</td>
<td>$r_1 = \text{FEC_decode}(r', w)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$K = \text{Hash}(r_1, x)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$h^* = \text{Hash}(r_1, x)$</td>
</tr>
<tr>
<td></td>
<td>Verify $h^* = h$</td>
<td></td>
</tr>
</tbody>
</table>
Authentication Algorithms

1. Alg # 1: Single Challenge

2. Alg # 2: Repeated Challenge

3. Alg # 3: Repeated Challenge with Bit Selection
### Authentication Algorithm #2: Repeated Challenge

**Server**

- Select $c, x, N$

**Channel**

- Generate $r, r_2 = \text{avg}(r_2)$

- $K, h$

- $(c, x, N) \rightarrow r' = PUF(c, N)$

**Client**

- $r_2 = \text{avg}(r', N), K$

- Verify $h^* = h$

- $h^* = \text{Hash}(r_2, x)$
Authentication Algorithms

1. Alg # 1: Single Challenge

2. Alg # 2: Repeated Challenge

3. Alg # 3: Repeated Challenge with Bit Selection
## Authentication Algorithm # 3: Repeated Challenge with Bit Selection

<table>
<thead>
<tr>
<th>Server</th>
<th>Channel</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select ( c, x, N, A )</td>
<td>((c, x, N)) \rightarrow ( r' = PUF(c, N) )</td>
<td>( r_{3a} = \text{avg}(r_3, N) )</td>
</tr>
<tr>
<td>Generate ( K, h )</td>
<td></td>
<td>( a = \text{Select_vector}(r_{3a}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( r_3 = \text{Bit_select}(r_{3a}, a) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( K = \text{Hash}(r_3, x) )</td>
</tr>
<tr>
<td>Verify ( h^* = h )</td>
<td>( h^* \leftarrow )</td>
<td>( h^* = \text{Hash}(r_3, x) )</td>
</tr>
</tbody>
</table>

