Reliable and Efficient PUF-Based Key Generation Using Pattern Matching

Authors:
Zdenek (Sid) Paral and Srinivas Devadas

Presented by:
Sugandha Gupta and Vishishtha Bothra
OUTLINE

- INTRODUCTION
  - MOTIVATION
  - SOLUTION
  - INSIGHT TO PUF & WORKING
  - PUF BASED KEY GENERATION
- PATTERN MATCHING KEY GENERATION (PMKG)
  - BASICS
  - PATTERN MATCHING
  - ARCHITECTURE
  - MAIN FEATURES
  - RELIABILITY
  - EVALUATION
- CONCLUSIONS
- REFERENCES
- DISCUSSION POINTS
Motivation

- Mobile and Embedded devices are used universally
- Important to protect critical application to prevent significant loss
- Need for Secure Computing

Current practices
- Place secret key in EEPROM/battery-backed SRAM
- Use hardware cryptographic operations
  - Digital signature
  - Encryption

- High Design Area
- High Power Consumption
- Vulnerable to invasive attacks
PUF - Physical Unclonable Function

- Silicon “biometric”
  - Unique
  - Robust
  - Unclonable
- Based on a complex physical system
- Secret is derived from the physical “variabilities” of the IC
  - Gate delay
  - Power on state of SRAM
  - Voltage threshold

✓ High Design Area
✓ Less Power
✓ Less Area
✗ Vulnerable to invasive attacks
✓ Invasive attacks are difficult to execute without modifying the physical characteristics

High Power Consumption
UNIVERSITY OF MICHIGAN

INSIGHT TO PUF

- Uniqueness
  - Same challenge input gives different responses

- Robustness
  - Stable over time

- Unclonability
  - Impractical and unreasonably expensive to clone

- Working
PUF BASED KEY GENERATION

- One of the major application of PUF
- Generates fixed size secret bits – seed, asymmetric key
- Caution!
  - The challenge response pair must be kept secret
  - Error correction needed for “noisy bits”
    - Helper data/ “syndrome”
  - Syndrome can further lead to leakage of data

Arbiter PUF
Two delay paths, produces output Y based on which is faster
BASICS OF KEY GENERATION

- Unconventional Approach
  - Challenge is kept secret instead of the response
- Provisioning
  - Externally provided secret seed
  - Generates public helper data and secret key
- Regeneration
  - Reproduce one of the earlier provisioned key using the helper data
**Pattern Matching**

- **Provisioning**
  - Secret Index (I)
  - Public Pattern (W)
  - Non-volatile Memory Block

- **Regeneration**
  - PUF
  - W from Non-Volatile Memory
  - Comparison Logic
  - Key Storage (Volatile Memory)

**Index**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
</table>

**Pattern**

<table>
<thead>
<tr>
<th></th>
<th>1011</th>
<th>1101</th>
<th>1111</th>
</tr>
</thead>
</table>
PMKG ARCHITECTURAL PARAMETERS

<table>
<thead>
<tr>
<th>Metric</th>
<th>Symbol</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key size</td>
<td>$K$</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>PUF count</td>
<td>$P$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Blender ratio</td>
<td>$B$</td>
<td>4</td>
<td>inp-bits/out-bit</td>
</tr>
<tr>
<td>Pattern width</td>
<td>$W$</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>Round length</td>
<td>$L$</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>Round count</td>
<td>$R$</td>
<td>16</td>
<td>$R \geq \frac{K}{N}$</td>
</tr>
<tr>
<td>Match threshold</td>
<td>$T$</td>
<td>80</td>
<td>Tolerance</td>
</tr>
<tr>
<td>Secret index size</td>
<td>$N$</td>
<td>10</td>
<td>$N = \log_2(L)$</td>
</tr>
<tr>
<td>Clocks per round</td>
<td>$CPR$</td>
<td>5,120</td>
<td>$CPR = \frac{(L+W) \times B}{P}$</td>
</tr>
<tr>
<td>Clocks total</td>
<td>$CT$</td>
<td>81,920</td>
<td>$CT = R \times CPR$</td>
</tr>
<tr>
<td>Entropy size</td>
<td>$E$</td>
<td>160</td>
<td>$K \leq E = R \times N$</td>
</tr>
<tr>
<td>Total pat. size</td>
<td>$S$</td>
<td>4,096</td>
<td>$S = W \times R$</td>
</tr>
</tbody>
</table>

**TABLE I**

ARCHITECTURAL PARAMETERS IN THE PATTERN MATCHING KEY GENERATOR
MAIN FEATURES OF PMKG

- Helper data/Syndrome cannot leak key
  - Index based key generation
- Forking in the challenge sequencer
  - Match signal is also used in the challenge sequencer
  - Subsequent challenge are less traceable after each iteration
  - Constant number of comparisons against helper pattern in each iterations
  - Avoids differential power and timing analysis attack
- Key Mixer
  - During provisioning – comparisons between running index counter and secret index of current round

SECURITY
Threshold has to be selected very wisely
- **Intra-PUF variability** – measure of reproducibility of responses
- **Inter-PUF variability** – measure of uniqueness of responses
- Leads to **Pattern Miss** and **Pattern Collision**
EVALUATION USING ASIC DATA

- Voltage variations in provisioning and regeneration at different temperatures
  - But can’t be quantified
- Threshold = 80 code bits
- Pattern Size W = 256 bits

<table>
<thead>
<tr>
<th>Settings W, T</th>
<th>Total Keygen</th>
<th>Trials</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1  2  3 4  5  6  7 8  9 10 11 12 13 14 15 16-19 20</td>
</tr>
<tr>
<td>96, 24</td>
<td>18535</td>
<td>14003 2500 865 355 231 126 85 66 38 37 23 17 13 7 39 94</td>
</tr>
<tr>
<td>128, 30</td>
<td>18535</td>
<td>18347 140 22 11 7 3 1 1 2 1</td>
</tr>
<tr>
<td>192, 54</td>
<td>18537</td>
<td>18534 3</td>
</tr>
<tr>
<td>256, 80</td>
<td>18540</td>
<td>18540</td>
</tr>
</tbody>
</table>

TABLE II
Key Re-Generation Results. Keys were provisioned at 25°C and re-generated between -25°C and +85°C.
CONCLUSIONS

▪ Viable method of PUF based Key Generation
  ▪ Key generation using index
  ▪ No storage of key directly
  ▪ Low clock latency
  ▪ Hardware requirements – minimal
  ▪ No error correction needed
    ▪ Select optimal Threshold (T) and Pattern Width (W)

▪ Improvements
  ▪ Can be made faster and secure by increasing number of PUF

▪ Future Work
  ▪ Use Delay PUF instead of Arbiter PUF
**REFERENCES**


- **Lecture Slides**

- **Web**
QUESTIONS
DISCUSSION Points

- Are the physical characteristics like process variation good enough to distinguish between ICs?
- Can PUF integrity be compromised with silicon wear out and extreme changes?
- Is setting threshold better or doing error correction?