When the Decoder Has to Look Twice: Clock-Glitching a PUF Error Correction

CHES 2022

Jonas Ruchti, Michael Gruber, Michael Pehl

Chair of Security in Information Technology
School of Computation, Information and Technology
Technical University of Munich

2022-09-21
Overview: Fuzzy Commitment Scheme

Enrolment phase:
- Key $k$ to ECC encoder
- ECC encoder produces $c$ and $r$ (private)
- Helper data $w$
- ECC decoder produces $\tilde{c}$
- $\tilde{r}$

Reconstruction phase:
- ECC decoder produces $\tilde{k}$
- $\tilde{k}$ is the key

PUF

Key $k$ to ECC encoder
ECC encoder produces $c$ and $r$ (private)
Helper data $w$
ECC decoder produces $\tilde{c}$
$\tilde{r}$
$\tilde{k}$ is the key
Overview: Fuzzy Commitment Scheme

Enrolment phase
- Key $k$ to ECC encoder
- ECC encoder produces $c$ and $r$
- Helper data: $w$

Reconstruction phase
- ECC decoder produces $\tilde{r}$ and $\tilde{c}$
- Private helper data $\tilde{k}$ is reconstructed from $\tilde{r}$ and $\tilde{c}$
- Public key $\tilde{k}$ is used to verify and reconstruct the key
Context: Vulnerabilities of PUF Key Storage Schemes

- Physical attacks on PUF primitives: SCA, FIA
- Side-channel attacks on the error correction codes (ECCs) of PUF systems, e.g. Merli, Stumpf, and Sigl, Protecting PUF Error Correction by Codeword Masking, 2013.

This work: First fault injection analysis targeting ECC implementations for PUF key storage.
Outline

1. Attack
2. Experiment
3. Masking
4. PUF Noise
5. Realistic Attacker
6. Conclusion
Attack: Scenario

- Physical access to the device under attack
- Possibility to repeatedly trigger key reconstruction phases
- Pass/fail reconstruction phase result
- Serial transfer of the codeword to the ECC decoder
Attack: Scenario

- Physical access to the device under attack
- Possibility to repeatedly trigger key reconstruction phases
- Pass/fail reconstruction phase result
- Serial transfer of the codeword to the ECC decoder

Initially: Powerful attacker

- Noise-free PUF response
- Read/write access to helper data
- Profiling and attack one the same device

Later: More realistic attacker
Attack: Fault Model

Clock glitch at bit position $g$:
- Bit $\tilde{c}_{g+1}$ is replaced by bit $\tilde{c}_g$.
- Helper data modification $\Rightarrow$ ECC is at error correction limit.
- Extraction of bit differences using clock glitch:
  - Reconstruction succeeds $\Rightarrow$ Bits at $g$ and $g+1$ are the same.
  - Reconstruction fails $\Rightarrow$ Bits at $g$ and $g+1$ differ.

$n-1$ fault injections $\Rightarrow n-1$ codeword bit differences recoverable.
Clock glitch at bit position $g$:
- Bit $\tilde{c}_{g+1}$ is replaced by bit $\tilde{c}_g$
**Attack: Fault Model**

Clock glitch at bit position $g$:
- Bit $\tilde{c}_{g+1}$ is replaced by bit $\tilde{c}_g$
- Helper data modification $\Rightarrow$ ECC is at error correction limit

Clock glitching a PUF Error Correction

Ruchti, Gruber, Pehl

CHES 2022

Slide 6/16
Clock glitch at bit position $g$:

- Bit $\tilde{c}_{g+1}$ is replaced by bit $\tilde{c}_g$
- Helper data modification $\Rightarrow$ ECC is at error correction limit
- Extraction of bit differences using clock glitch:
  - Reconstruction succeeds $\Rightarrow$ Bits at $g$ and $g+1$ are the same
  - Reconstruction fails $\Rightarrow$ Bits at $g$ and $g+1$ differ
Attack: Fault Model

Clock glitch at bit position $g$:
- Bit $\tilde{c}_{g+1}$ is replaced by bit $\tilde{c}_g$
- Helper data modification $\Rightarrow$ ECC is at error correction limit
- Extraction of bit differences using clock glitch:
  - $\square$ Reconstruction succeeds $\Rightarrow$ Bits at $g$ and $g + 1$ are the same
  - $\square$ Reconstruction fails $\Rightarrow$ Bits at $g$ and $g + 1$ differ
- $n - 1$ fault injections $\Rightarrow$ $n - 1$ codeword bit differences recoverable
Experiment: Set-up

System under attack

- BCH decoder
- SIPO SRG
- UART interface, control logic
- Clock
- Trigger

Repetition decoder

(7, 1, 3)

(127, 64, 10)

BCH decoder
Experiment: Set-up

PC

UART interface, control logic

\( \tilde{\mathbf{c}} \)

\( \tilde{\mathbf{k}} \)

System under attack

\( \tilde{\mathbf{c}} \)

\( \tilde{\mathbf{k}} \)

PISO SRG

\( (7, 1, 3) \) Repetition decoder

SIPO SRG

\( (127, 64, 10) \) BCH decoder

Glitch generator

Configuration

Trigger

Clock

Artix-7 FPGA × 15
Experiment: Procedure

Profiling
- Estimate of observable data dependency
- One random key per board
- Random search to find best glitch parameters for each board

Attack
- 15 FPGA boards
- 100 random keys attacked per board
- 250 trials with 127 clock glitches each
- Codeword extracted from average of trials
Experiment: Procedure

Profiling
- Estimate of observable data dependency
- One random key per board
- Random search to find best glitch parameters for each board

Attack
- 15 FPGA boards
- 100 random keys attacked per board
- 250 trials with 127 clock glitches each
- Codeword extracted from average of trials
Glitch Effects: Model Prediction

Experiment: 250 random codewords with a varying number of extra bit flips

![Graph showing the relationship between glitch position and reconstruction failures.](image)
Glitch Effects: Observed

Experiment: 250 random codewords with a varying number of extra bit flips

![Graph showing reconstruction failures (%) vs. Glitch position g for different numbers of additional bit flips.](image-url)
### Experiment: Results

After 250 trials: 14.8 codeword bit extraction errors on average

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Guess entropy (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>24.7 bit</td>
</tr>
</tbody>
</table>
Experiment: Results

After 250 trials: 14.8 codeword bit extraction errors on average

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Guess entropy (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>24.7 bit</td>
</tr>
<tr>
<td>Maximum-variance (MV)</td>
<td>10.4 bit</td>
</tr>
</tbody>
</table>

Number of trials

MV guesses

board 6, board 9, board 10, board 15
**Experiment: Results**

After 250 trials: 14.8 codeword bit extraction errors on average

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Guess entropy (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>24.7 bit</td>
</tr>
<tr>
<td>Maximum-variance (MV)</td>
<td>10.4 bit</td>
</tr>
<tr>
<td>Maximum extraction error probability</td>
<td>8.6 bit</td>
</tr>
</tbody>
</table>

![Graph showing MV guesses over trials for different boards](image-url)
Masking

- Masking is effective against SCA with a similar scenario\(^1\).
- Based on the fault model, masking *could* protect against the attack.

Masking

- Masking is effective against SCA with a similar scenario\(^1\)
- Based on the fault model, masking *could* protect against the attack

---

Masking

- Masking is effective against SCA with a similar scenario\(^1\)
- Based on the fault model, masking could protect against the attack

\[ \text{RNG} \]

\[ \text{BCH encoder} \]

\[ \text{Repetition decoder} \]

\[ \text{BCH decoder} \]

\[ \tilde{c} \]

\[ \tilde{k} \]

---

Experiment results

- Even better attack performance than for the unmasked case
- All 1500 tested codewords perfectly extractable
(a) Mask Applied at BCH Decoder Input

Experiment results
- Even better attack performance than for the unmasked case
- All 1500 tested codewords perfectly extractable

Why?

Clock

Mask

\[ \begin{align*}
  m_{i-1} & \quad m_i & \quad m_{i+1} & \quad m_{i+2} \\
  \tilde{c}_{i-1} & \quad \tilde{c}_i & \quad \tilde{c}_{i+1} & \quad \tilde{c}_{i+2}
\end{align*} \]
Some amount of protection, but...

- Best-attackable board: 19 bit left to guess on average
- Mask/codeword propagation delay matching can be impractical
PUF Noise

Model

- Static response offset (e.g. ageing): Can be extracted and compensated
- Measurement noise: Must be compensated by averaging more trials

Experiment

- PUF response now has i.i.d. measurement noise
- Attack is carried out with different BERs

Results

- Repetition decoder limits noise influence: For BER $\text{PUF} \leq 11\%$, fewer than 0.5 bit errors are left on average
- Attack progress is slower, but averaging can combat the remaining errors

$\Rightarrow$ Realistic error rates: Attack performance is nearly the same
PUF Noise

Model
- Static response offset (e.g. ageing): Can be extracted and compensated
- Measurement noise: Must be compensated by averaging more trials

Experiment
- PUF response now has i.i.d. measurement noise
- Attack is carried out with different BERs
PUF Noise

Model
- Static response offset (e.g. ageing): Can be extracted and compensated
- Measurement noise: Must be compensated by averaging more trials

Experiment
- PUF response now has i.i.d. measurement noise
- Attack is carried out with different BERs

Results
- Repetition decoder limits noise influence:
  For $\text{BER}_{\text{PUF}} \leq 11\%$, fewer than 0.5 bit errors are left on average
- Attack progress is slower, but averaging can combat the remaining errors
  \Rightarrow Realistic error rates: Attack performance is nearly the same
Realistic Attacker

- Profile one device, attack a different device
- No more helper data access/manipulation
- Attacker can *increase* PUF noise
Realistic Attacker

- Profile one device, attack a different device
- No more helper data access/manipulation
- Attacker can increase PUF noise

Results

- Attack is significantly slower (approx. factor 10)
- Attack performs well except for 3 boards
- Best-attackable board: 9.6 bit guesses left on average after 250 trials
Summary

Conclusion
- FIA on the ECC in PUF key storage can be feasible and more powerful than SCA
- Masking is difficult to get right and can even make matters worse
- Helper data manipulation detection does not always help
Summary

Conclusion
- FIA on the ECC in PUF key storage can be feasible and more powerful than SCA
- Masking is difficult to get right and can even make matters worse
- Helper data manipulation detection does not always help

In the full paper
- Profiling, extraction of static PUF response offsets
- HD manipulation and guessing strategies for more efficient key/codeword recovery
- Error-correcting partially extracted codewords
- Other secure sketches (e.g. syndrome construction)