Random Delay Insertion: Effective Countermeasure against DPA on FPGAs

Lu, Yingxi
Dr. Máire O’Neill
Prof. John McCanny
DPA and countermeasures

Random Delay Insertion (RDI) countermeasure

- Design parameters
- FPGA implementation
- Case study on AES

Attacks against RDI
Side Channel Attack

Reveal the security key stored in cryptographic implementations by monitoring physical characteristics, e.g. power, EM, timing.

Differential Power Analysis

One of the most effective SCA attacks. Analyzes the instantaneous power consumption of a security device.
WHAT IS DPA?

Probe 1

R = 1 ohm

Probe 2

Vcc

Clock Generator

Bus

Virtex II Pro FPGA

Ck

T: Trace set.

pt: Plaintext of round i.

ihr: Intermediate result under hypothesis of round i.
Key-space under attack reduces from $2^{256}$ to $2^8 \times 64$.

- **Device Under Attack**
- **Oscilloscope Board Controller**
- **Mix Signal Oscilloscope**
- **AES Software**
- **Sub-key Space**

Trace set: $T$

- **plaintext : Power**
- **plaintext : key**

**Pre-process** → **Correlation function** → **Test vectors**
To be effective, a cryptographic implementation should include multiple countermeasure techniques.

- Ephemeral keys, etc.
- Masking, etc.
- Random Delay Insertion is here.
Each countermeasure has its weakness.

Only applicable to specific applications.

- Ephemeral keys, etc.
- Masking, etc.
- Precharged Dual-Rail Logic, etc.

[Source: Mangard et al.]

10× power consumption.
Delays can be inserted before the cryptographic execution to change the position of operations in the time-dimension.
Delays can be inserted before the cryptographic execution to change the position of operations in the time-dimension.

- Length of delay corresponds to length of the clock cycle - can be easily identified.
- And therefore, the length of the delay is limited.
- Registers need to be included between the delay and the key-dependent operations.

The above disadvantages DO NOT apply on FPGA & ASIC. The delay is not implemented by instructions.

This work is the first RDI FPGA implementation.
$$P_{\text{total}} = (P_{\text{data-dependent}} + P_{\Delta}) + P_{\text{data-independent}}$$
\[ P_{\text{total}} = (P_{\text{data-dependent}} + P_{\Delta}) + P_{\text{data-independent}} \]

\[ C(H, P_{\text{total}}) = \frac{E(H \cdot (P_d + P_{\Delta} + P_{\text{ind}})) - E(H) \cdot E((P_d + P_{\Delta} + P_{\text{ind}}))}{\sqrt{\text{Var}(H) \cdot \text{Var}(P_d + P_{\Delta} + P_{\text{ind}})}} \]

\[ \therefore P_d \perp P_{\Delta} \perp P_{\text{ind}}, H \perp P_{\Delta}, H \perp P_{\text{ind}} \]

Given \( P_d \) and given large \( \Delta \)

\[ \sqrt{1 + \frac{\text{Var}(P_{\Delta})}{\text{Var}(P_d)}} \approx \sqrt{rac{\text{Var}(P_{\Delta})}{\text{Var}(P_d)}} \propto \sqrt{\text{Var}(P_{\Delta})} \]

Following linear assumption, \( P_{\Delta} : \Delta \)

\[ C(H, P_{\text{total}}) \propto \frac{1}{\sqrt{k \cdot \Delta_{\text{max}}}} = \frac{\sqrt{\delta}}{\Delta_{\text{max}}} \]
\[ P_{\text{total}} = (P_{\text{data-dependent}} + P_{\Delta}) + P_{\text{data-independent}} \]

\[ C(H, P_{\text{total}}) = E(H \cdot (P_d + P_{\Delta} + P_{\text{ind}})) - E(H) \cdot E((P_d + P_{\Delta} + P_{\text{ind}})) \]

- \( \Delta \): discrete variable, \( 0 < \Delta < \Delta_{\text{max}} \).
- \( \delta \): the deviation of \( \Delta \).
- \( k \): natural number, \( \Delta_{\text{max}} = k \times \delta \).
- Lower \( C(H, P_{\text{total}}) \) to achieve higher security - requires large \( \Delta_{\text{max}} \) and \( k \), but small \( \delta \).
- Trade-off between security and performance / cost.

\[
\begin{align*}
C(H, P_{\text{total}}) & \propto \frac{1}{\sqrt{1 + \frac{1}{\text{SNR}}} \sqrt{1 + \frac{\text{Var}(P_{\Delta})}{\text{Var}(P_d)}}} \\
& \propto \frac{1}{\sqrt{k \cdot \Delta_{\text{max}}}} = \frac{\sqrt{\delta}}{\Delta_{\text{max}}} 
\end{align*}
\]
Step 1. Measure the power traces from the original cryptographic implementation.
Step 2. Modify the power traces with random delay up to $\Delta_{\text{max}}$ and down to $\delta$.
Step 3. Perform DPA on the shifted traces to check whether it is successful.
- **Split Random Delay Insertion (Split-RDI)**

- **Attackers can counteract RDI using realignment algorithms to identify the delay.**

- **We split and apply the delay to multiple stages to protect our delay from such realignment attacks.**
Delay logic [Bucci et al.]

True Random Number Generator [Schellekens et al.]

Split Random Delay Insertion (Split RDI)
Now let’s recall the parameters.

To lower the correlation, we should:

- Increase $\Delta_{\text{max}}$. But this increases the delay and reduces the speed.
- Decrease $\delta$. Significantly lowers correlation. But limited by $k$.
- Increase $k$. But significantly increases power and area, a larger $k$ implies more random bits.
Step 1. Measure the power traces from the original cryptographic implementation.

Step 2. Modify the power traces with random delay up to $\Delta_{\text{max}}$ and down to $\delta$.

Step 3. Perform DPA on the shifted traces to check whether it is successful.
DPA results of shifted power traces measured from an original AES FPGA implementation.

- \( k = 12 \) means 4 bits TRNG is required.

\[ \delta = t_s: \text{RDI is effective when } \Delta_{\text{max}} > 12t_s. \]
DPA results of shifted power traces measured from an original AES FPGA implementation.

Correct Key Hypothesis

\[ \delta = 8t_s \]

Wrong Key Hypothesis

\[ \delta = 10t_s \]

Effective \( \Delta_{\text{max}} \) increases considerably with larger \( \delta \)!

\[ \delta = 8t_s: \Delta_{\text{max}} > 23t_s, \quad \delta = 10t_s: \Delta_{\text{max}} > 33t_s \]
We finally implemented the RDI with the following parameters:

\[ \delta = 2t_s, \Delta_{\text{max}} = 16t_s, k = 8. \]

(3 bits TRNG is required.)

RDI results in a low-cost DPA countermeasure (figures including the TRNG).
We evaluated the above AES FPGA implementation with RDI applied under three DPA attack techniques.

- Original DPA attack.
- DPA attack with Sliding Window post-computation.
- DPA attack with realignment pre-computation.
A recently developed DPA attack in the frequency domain is also performed on the proposed countermeasure.

\[ x(n) \rightarrow X(\kappa) = \sum_{n=0}^{N-1} x(n) \cdot \omega_N^{n \cdot \kappa} \]
\[ x(n - l) \rightarrow X(\kappa) \cdot \omega_N^{n \cdot l} \]

However, the effectiveness of such an attack depends on the DFT window size \( N \).
Random Delay Insertion technique is an effective countermeasure against DPA when implemented on FPGAs.

The trade off between performance and design cost has been discussed. And RDI results in a low-cost DPA countermeasure.

A Split-RDI with carefully chosen delay can be used to prevent realignment, Sliding Window DPA and the attack in the frequency domain.

Thank you!