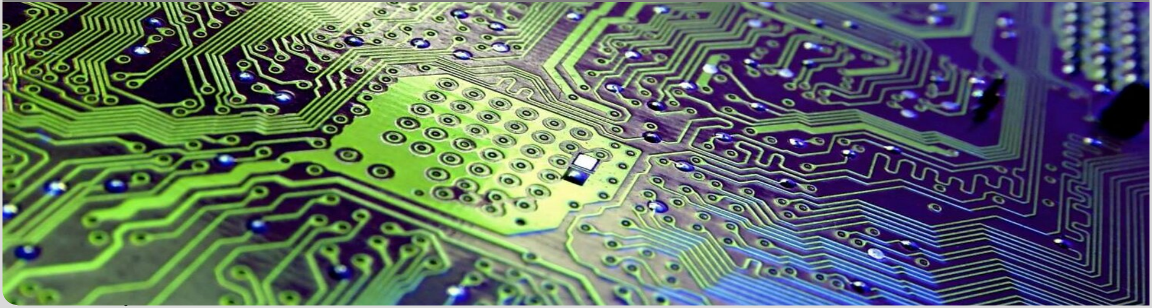
Karlsruher Institut für Technologie

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES

Jonas Krautter, Dennis R.E. Gnad, Mehdi B. Tahoori | 10.09.2018

INSTITUTE OF COMPUTER ENGINEERING – CHAIR OF DEPENDABLE NANO COMPUTING



Motivation

- More resources per FPGA ⇒ **Multi-user** environments:
  - Amazon, Microsoft and introduce FPGA usage in cloud computing
  - System-on-Chip (SoC) variants, tightly coupled FPGA based systems (Xilinx PYNQ, Intel Xeon FPGA, Intel/Altera-SoCs...)
  - Accelerators deployed to partitions through partial reconfiguration ⇒ **Multi-tenant** FPGAs

Motivation

- More resources per FPGA ⇒ **Multi-user** environments:
  - Amazon, Microsoft and introduce FPGA usage in cloud computing
  - System-on-Chip (SoC) variants, tightly coupled FPGA based systems (Xilinx PYNQ, Intel Xeon FPGA, Intel/Altera-SoCs...)
  - Accelerators deployed to partitions through partial reconfiguration ⇒ **Multi-tenant** FPGAs
- New attack scenarios:
  - Passive on-chip side-channels[1]
  - Denial-of-Service[2]
  - **This work: Fault attacks**
  - ...

[1] Schellenberg et al., ”An Inside Job: Remote Power Analysis Attacks on FPGAs”, DATE 2018

[2] Gnad et al., ”Voltage drop-based fault attacks on FPGAs using valid bitstreams”, FPL 2017

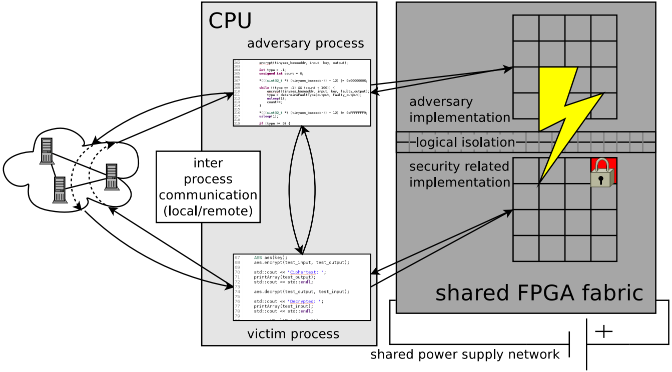
Motivation

- More resources per FPGA ⇒ **Multi-user** environments:
  - Amazon, Microsoft and introduce FPGA usage in cloud computing
  - System-on-Chip (SoC) variants, tightly coupled FPGA based systems (Xilinx PYNQ, Intel Xeon FPGA, Intel/Altera-SoCs...)
  - Accelerators deployed to partitions through partial reconfiguration ⇒ **Multi-tenant** FPGAs
- New attack scenarios:
  - Passive on-chip side-channels[1]
  - Denial-of-Service[2]
  - **This work: Fault attacks**
  - ...
- Proof-of-Concept work: Successful DFA on AES

[1] Schellenberg et al., ”An Inside Job: Remote Power Analysis Attacks on FPGAs”, DATE 2018

[2] Gnad et al., ”Voltage drop-based fault attacks on FPGAs using valid bitstreams”, FPL 2017

Threat model



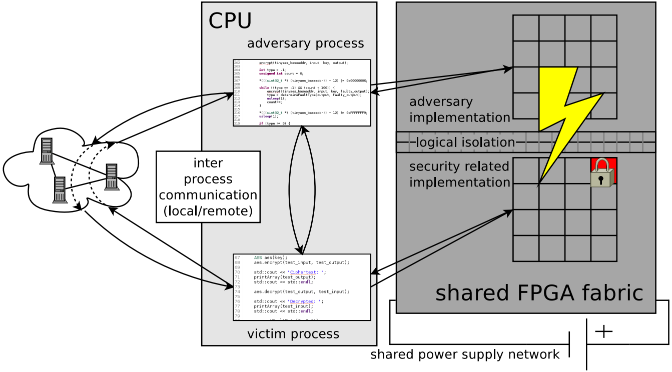
```
        encrypt(tinyaes_baseaddr, input, key, output);

        int type = -1;
        unsigned int count = 0;

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) |= 0x00000006;

        while ((type == -1) && (count < 100)) {
            encrypt(tinyaes_baseaddr, input, key, faulty_output);
            type = determineFaultType(output, faulty_output);
            usleep(1);
            count++;
        }

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) &= 0xFFFFFFF9;
        usleep(1);

        if (type >= 0) {
```

```
    AES aes(key);
    aes.encrypt(test_input, test_output);

    std::cout << "Ciphertext: ";
    printArray(test_output);
    std::cout << std::endl;

    aes.decrypt(test_output, test_input);

    std::cout << "Decrypted: ";
    printArray(test_input);
    std::cout << std::endl;

```

CPU
adversary process
inter process communication (local/remote)
victim process
adversary implementation
logical isolation
security related implementation
shared FPGA fabric
+
shared power supply network

- Shared FPGA fabric ⇒ **Shared Power Distribution Network (PDN)**

Threat model

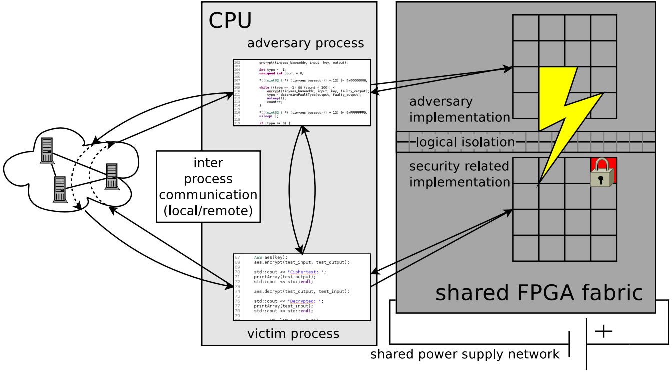

CPU
adversary process
inter process communication (local/remote)
victim process
adversary implementation
logical isolation
security related implementation
shared FPGA fabric
+
shared power supply network

```
        encrypt(tinyaes_baseaddr, input, key, output);

        int type = -1;
        unsigned int count = 0;

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) |= 0x00000006;

        while ((type == -1) && (count < 100)) {
            encrypt(tinyaes_baseaddr, input, key, faulty_output);
            type = determineFaultType(output, faulty_output);
            usleep(1);
            count++;
        }

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) &= 0xFFFFFFF9;
        usleep(1);

        if (type >= 0) {
```

```
    AES aes(key);
    aes.encrypt(test_input, test_output);

    std::cout << "Ciphertext: ";
    printArray(test_output);
    std::cout << std::endl;

    aes.decrypt(test_output, test_input);

    std::cout << "Decrypted: ";
    printArray(test_input);
    std::cout << std::endl;
```

- Shared FPGA fabric ⇒ **Shared Power Distribution Network (PDN)**
- Attacker and victim design **logically isolated**

Threat model

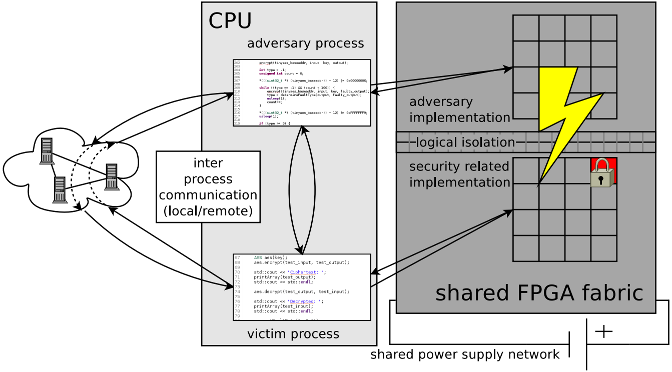

CPU
adversary process
inter process communication (local/remote)
victim process
adversary implementation
logical isolation
security related implementation
shared FPGA fabric
shared power supply network
+

```
        encrypt(tinyaes_baseaddr, input, key, output);

        int type = -1;
        unsigned int count = 0;

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) |= 0x00000006;

        while ((type == -1) && (count < 100)) {
            encrypt(tinyaes_baseaddr, input, key, faulty_output);
            type = determineFaultType(output, faulty_output);
            usleep(1);
            count++;
        }

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) &= 0xFFFFFFF9;
        usleep(1);

        if (type >= 0) {
```

```
    AES aes(key);
    aes.encrypt(test_input, test_output);

    std::cout << "Ciphertext: ";
    printArray(test_output);
    std::cout << std::endl;

    aes.decrypt(test_output, test_input);

    std::cout << "Decrypted: ";
    printArray(test_input);
    std::cout << std::endl;

```

- Shared FPGA fabric ⇒ **Shared Power Distribution Network (PDN)**
- Attacker and victim design **logically isolated**
- Victim software process has a public interface

Threat model


CPU
adversary process
inter process communication (local/remote)
victim process
adversary implementation
logical isolation
security related implementation
shared FPGA fabric
+
shared power supply network

```
        encrypt(tinyaes_baseaddr, input, key, output);

        int type = -1;
        unsigned int count = 0;

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) |= 0x00000006;

        while ((type == -1) && (count < 100)) {
            encrypt(tinyaes_baseaddr, input, key, faulty_output);
            type = determineFaultType(output, faulty_output);
            usleep(1);
            count++;
        }

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) &= 0xFFFFFFF9;
        usleep(1);

        if (type >= 0) {
```

```
    AES aes(key);
    aes.encrypt(test_input, test_output);

    std::cout << "Ciphertext: ";
    printArray(test_output);
    std::cout << std::endl;

    aes.decrypt(test_output, test_input);

    std::cout << "Decrypted: ";
    printArray(test_input);
    std::cout << std::endl;

```

- Shared FPGA fabric ⇒ **Shared Power Distribution Network (PDN)**
- Attacker and victim design **logically isolated**
- Victim software process has a public interface
- **Chosen-Plaintext Attack** scenario

Outline

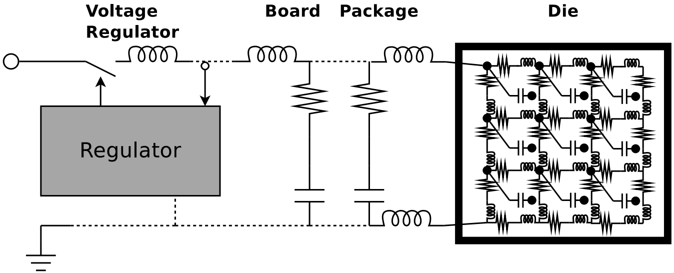
1. Background
2. Fault Injection and Analysis
3. Experimental Setup
4. Results
5. Discussion and Future Work
6. Conclusion

Outline

1. Background
2. Fault Injection and Analysis
3. Experimental Setup
4. Results
5. Discussion and Future Work
6. Conclusion

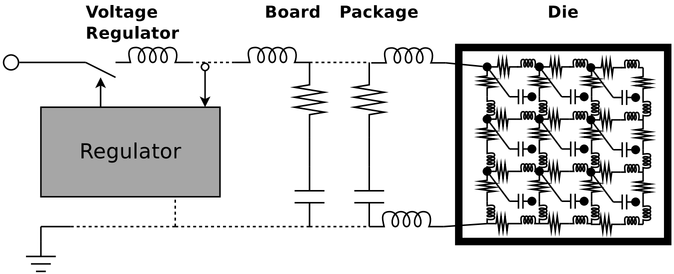
Power Distribution Network (PDN)

- Interconnections from the voltage regulator down to logic elements
- Model: RLC-mesh (Resistive, Inductive and Capacitive elements)



Power Distribution Network (PDN)

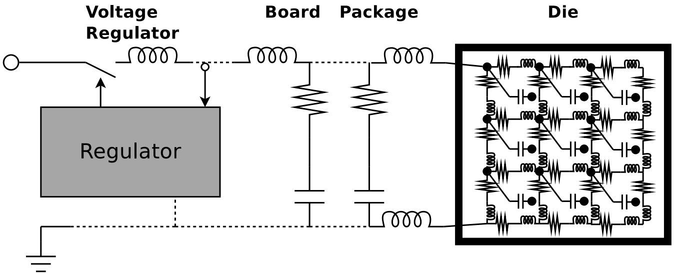
- Interconnections from the voltage regulator down to logic elements
- Model: RLC-mesh (Resistive, Inductive and Capacitive elements)



- Law of Inductance: $V_{\text{drop}} = I \cdot R + L \cdot \frac{dI}{dt}$

Power Distribution Network (PDN)

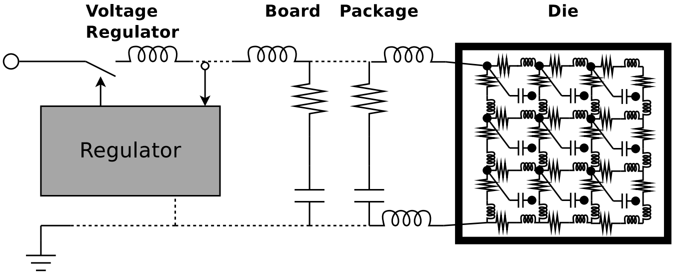
- Interconnections from the voltage regulator down to logic elements
- Model: RLC-mesh (Resistive, Inductive and Capacitive elements)



- Law of Inductance: $V_{\text{drop}} = I \cdot R + L \cdot \frac{dI}{dt}$
- High current variation ⇒ Power supply voltage variation

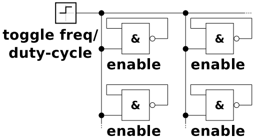
Power Distribution Network (PDN)

- Interconnections from the voltage regulator down to logic elements
- Model: RLC-mesh (Resistive, Inductive and Capacitive elements)



- Law of Inductance: $V_{\text{drop}} = I \cdot R + L \cdot \frac{dI}{dt}$
- High current variation ⇒ Power supply voltage variation
- Lower supply voltage ⇒ **Timing faults**

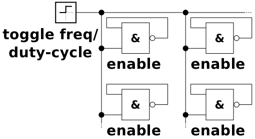
Malicious Logic



- Logic element to cause high current variation[2]: **Ring Oscillators (ROs)**

[2] Gnad et al., ”Voltage drop-based fault attacks on FPGAs using valid bitstreams”, FPL 2017

Malicious Logic



- Logic element to cause high current variation[2]: **Ring Oscillators (ROs)**

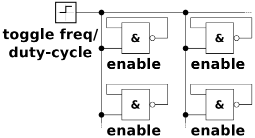
- Oscillation ⇒ Gate switching ⇒ Current variation ⇒ Voltage drop

[2] Gnad et al., "Voltage drop-based fault attacks on FPGAs using valid bitstreams", FPL 2017

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Malicious Logic



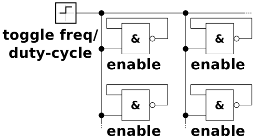
- Logic element to cause high current variation[2]: **Ring Oscillators (ROs)**
- Oscillation ⇒ Gate switching ⇒ Current variation ⇒ Voltage drop
- RO-grid must be toggled in a very specific way (freq, duty-cycle, delay)

[2] Gnad et al., ”Voltage drop-based fault attacks on FPGAs using valid bitstreams”, FPL 2017

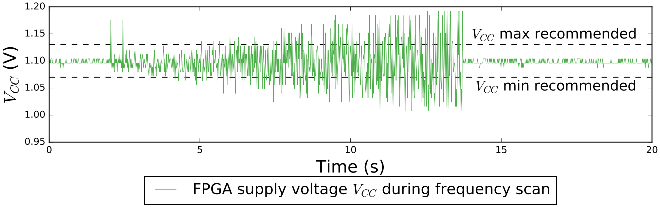
FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Malicious Logic

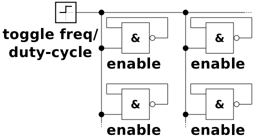


- Logic element to cause high current variation[2]: **Ring Oscillators (ROs)**
- Oscillation ⇒ Gate switching ⇒ Current variation ⇒ Voltage drop
- RO-grid must be toggled in a very specific way (freq, duty-cycle, delay)
- ⇒ **Calibration** of fault injection parameters required



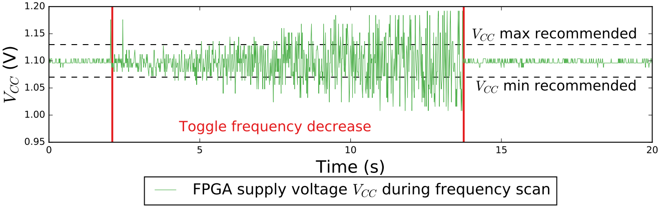
[2] Gnad et al., "Voltage drop-based fault attacks on FPGAs using valid bitstreams", FPL 2017

Malicious Logic



- Logic element to cause high current variation[2]: **Ring Oscillators (ROs)**

- Oscillation ⇒ Gate switching ⇒ Current variation ⇒ Voltage drop
- RO-grid must be toggled in a very specific way (freq, duty-cycle, delay)
- ⇒ **Calibration** of fault injection parameters required



[2] Gnad et al., ”Voltage drop-based fault attacks on FPGAs using valid bitstreams”, FPL 2017

Outline

1. Background
2. Fault Injection and Analysis
3. Experimental Setup
4. Results
5. Discussion and Future Work
6. Conclusion

Fault Injection and Analysis

- Differential Fault Analysis on AES[3]

[3] Piret et al., ”A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad”, CHES 2003

Fault Injection and Analysis

- Differential Fault Analysis on AES[3]
- Original scheme: Single-byte faults before 8th round

  ⇒ All output bytes faulty

[3] Piret et al., ”A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad”, CHES 2003

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

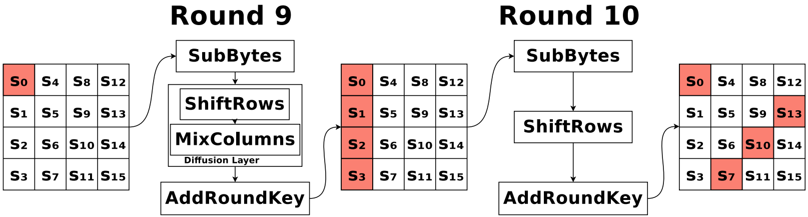
J. Krautter, D.R.E. Gnad and M.B. Tahoori

Fault Injection and Analysis

- Differential Fault Analysis on AES[3]
- Original scheme: Single-byte faults before 8th round

  ⇒ All output bytes faulty
- Injection requires high precision

  ⇒ Fault injection before 9th round



Round 9

🟥 S_0	S_4	S_8	S_{12}
S_1	S_5	S_9	S_{13}
S_2	S_6	S_{10}	S_{14}
S_3	S_7	S_{11}	S_{15}

SubBytes

ShiftRows

MixColumns

Diffusion Layer

AddRoundKey

🟥 S_0	S_4	S_8	S_{12}
🟥 S_1	S_5	S_9	S_{13}
🟥 S_2	S_6	S_{10}	S_{14}
🟥 S_3	S_7	S_{11}	S_{15}

Round 10

SubBytes

ShiftRows

AddRoundKey

🟥 S_0	S_4	S_8	S_{12}
S_1	S_5	S_9	🟥 S_{13}
S_2	S_6	🟥 S_{10}	S_{14}
S_3	🟥 S_7	S_{11}	S_{15}

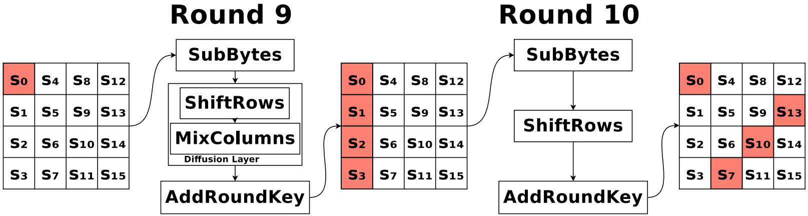
[3] Piret et al., "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad", CHES 2003

Fault Injection and Analysis

- Differential Fault Analysis on AES[3]
- Original scheme: Single-byte faults before 8th round

  ⇒ All output bytes faulty
- Injection requires high precision

  ⇒ Fault injection before 9th round

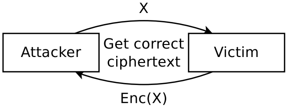


- Successful injection can be **verified**

[3] Piret et al., ”A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad”, CHES 2003

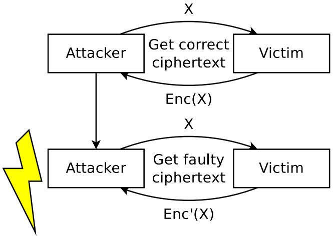
Fault Injection and Analysis

- Attacker issues encryption request to get correct ciphertext



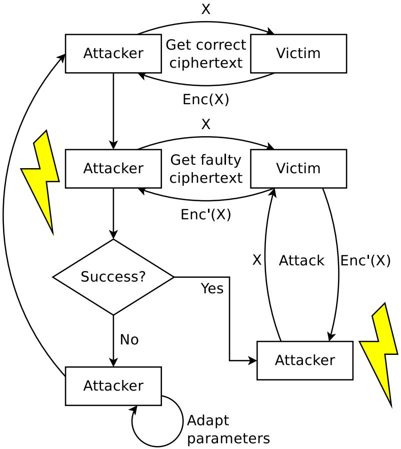
Fault Injection and Analysis

- Attacker issues encryption request to get correct ciphertext
- Attacker issues encryption requests while activating RO grid



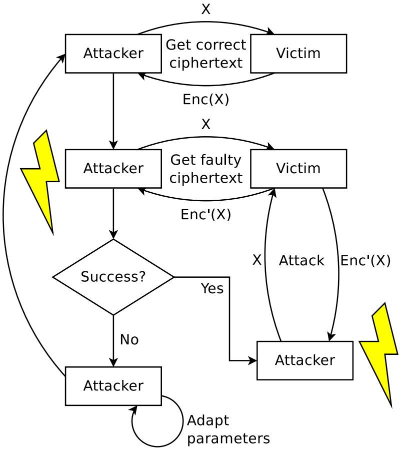
Fault Injection and Analysis

- Attacker issues encryption request to get correct ciphertext
- Attacker issues encryption requests while activating RO grid
- Fault injection is **calibrated** until desired faults appear



Fault Injection and Analysis

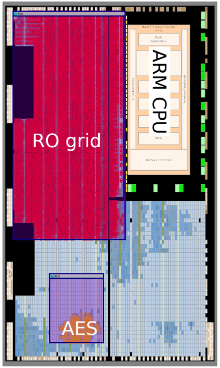
- Attacker issues encryption request to get correct ciphertext
- Attacker issues encryption requests while activating RO grid
- Fault injection is **calibrated** until desired faults appear
- Calibration is done only **once** for a specific board



Outline

1. Background
2. Fault Injection and Analysis
3. Experimental Setup
4. Results
5. Discussion and Future Work
6. Conclusion

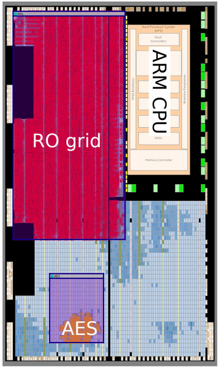
Experimental Setup



- FPGA boards: 3× Terasic DE1-SoC, 1× Terasic DE0-Nano-SoC
  - 3 boards of the same type
  - 2 different boards

    ⇒ Show generality of attack
- Cyclone V FPGA and ARM Cortex-A9 on one chip
- Linux environment on ARM Cortex-A9

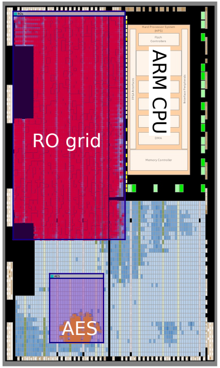
Experimental Setup



- FPGA boards: 3× Terasic DE1-SoC, 1× Terasic DE0-Nano-SoC
  - 3 boards of the same type
  - 2 different boards

    ⇒ Show generality of attack
- Cyclone V FPGA and ARM Cortex-A9 on one chip
- Linux environment on ARM Cortex-A9
- Entire threat model in one SoC:
  - Attacker and victim software on ARM core
  - Respective IP cores on FPGA fabric

Experimental Setup



- FPGA boards: 3× Terasic DE1-SoC, 1× Terasic DE0-Nano-SoC
  - 3 boards of the same type
  - 2 different boards

    ⇒ Show generality of attack
- Cyclone V FPGA and ARM Cortex-A9 on one chip
- Linux environment on ARM Cortex-A9
- Entire threat model in one SoC:
  - Attacker and victim software on ARM core
  - Respective IP cores on FPGA fabric
- Fault injection on SoC, Key recovery on PC

Outline

1 Background

2 Fault Injection and Analysis

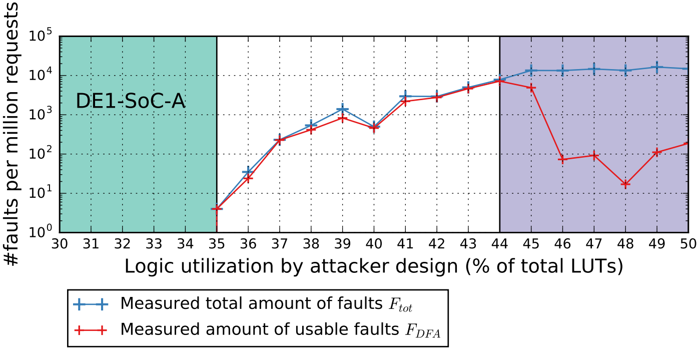
3 Experimental Setup

4 Results

5 Discussion and Future Work

6 Conclusion

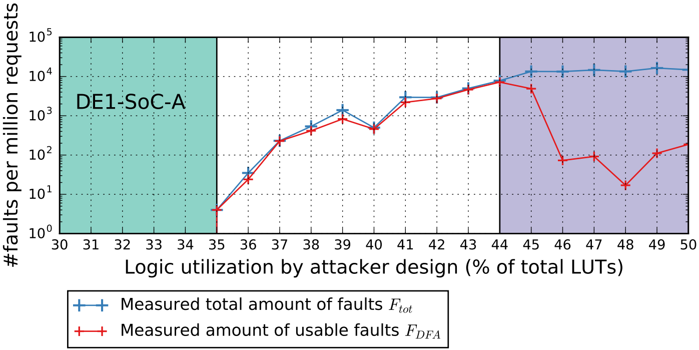
Fault Injection Rate vs #RO



- Experiments on DE1-SoC, design **fully constrained**

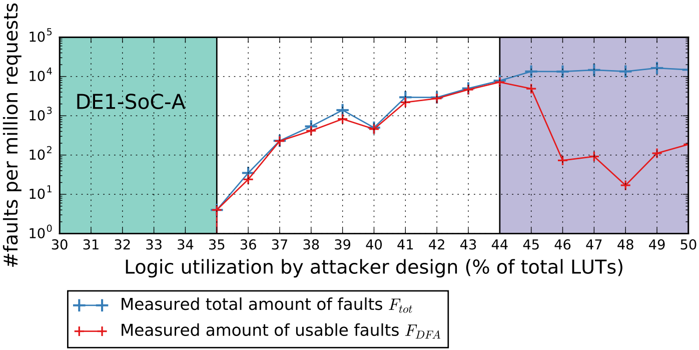
J. Krautter, D.R.E. Gnad and M.B. Tahoori

Fault Injection Rate vs #RO



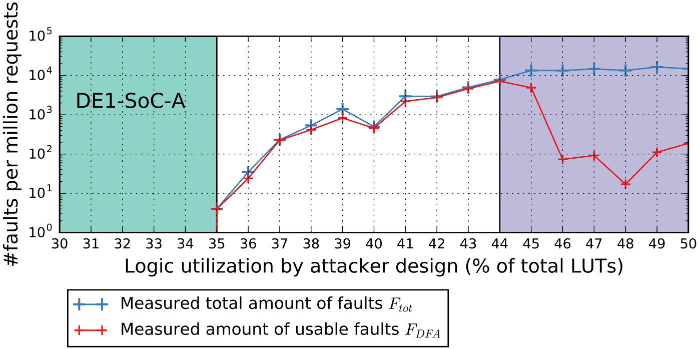
- Experiments on DE1-SoC, design **fully constrained**
- Evaluate **usable** (for DFA) faults and **total** amount of faults

Fault Injection Rate vs #RO



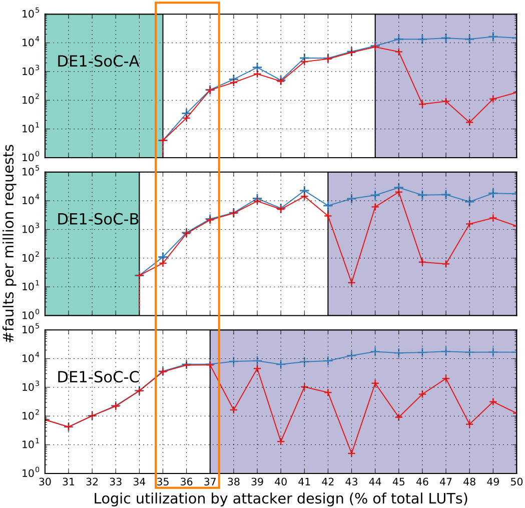
- Experiments on DE1-SoC, design **fully constrained**
- Evaluate **usable** (for DFA) faults and **total** amount of faults
- Injection rate increases with amount of ROs

Fault Injection Rate vs #RO



- Experiments on DE1-SoC, design **fully constrained**
- Evaluate **usable** (for DFA) faults and **total** amount of faults
- Injection rate increases with amount of ROs
- Injection accuracy decreases after a certain amount

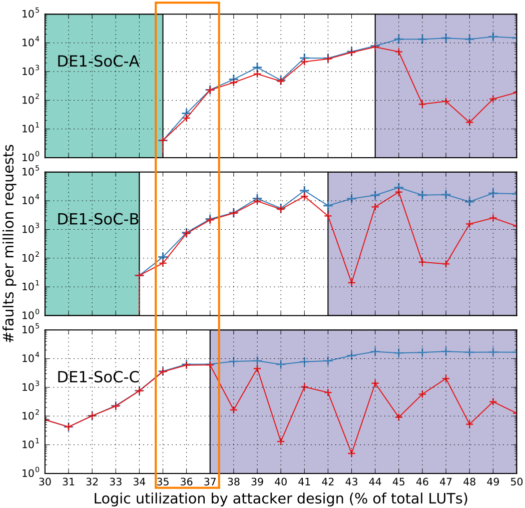
Fault Injection Rate vs #RO



- Extended experiments: 3 different boards

J. Krautter, D.R.E. Gnad and M.B. Tahoori

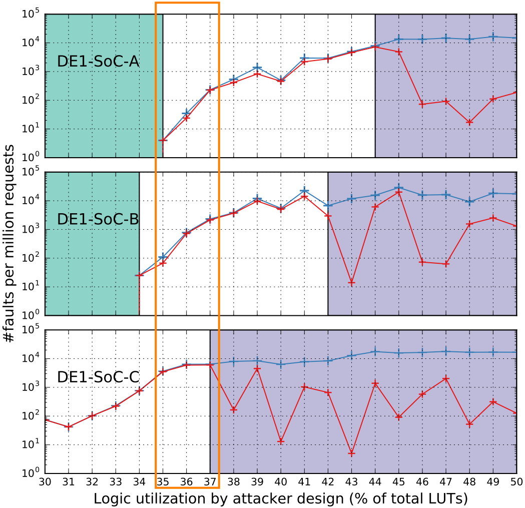
Fault Injection Rate vs #RO



- Extended experiments: 3 different boards
- All boards vulnerable, Calibration finds params

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Fault Injection Rate vs #RO

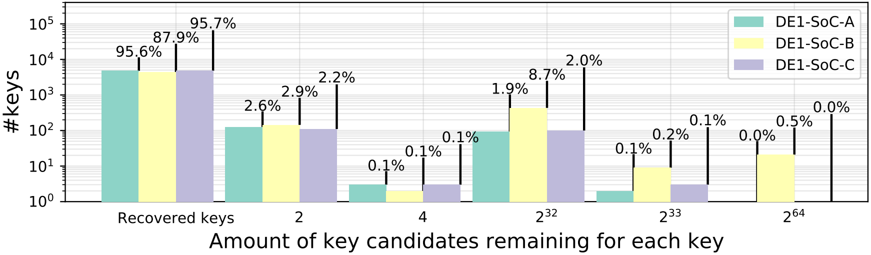


- Extended experiments: 3 different boards
- All boards vulnerable, Calibration finds params
- Process variation ⇒ Different optimal #RO

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Key Recovery on 5000 random keys

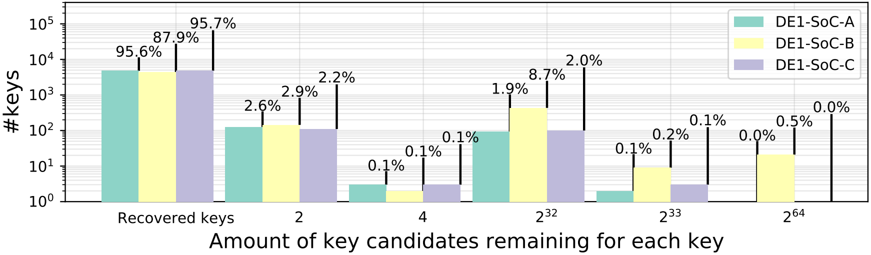


- Experiments on DE1-SoC with best fault injection configuration

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

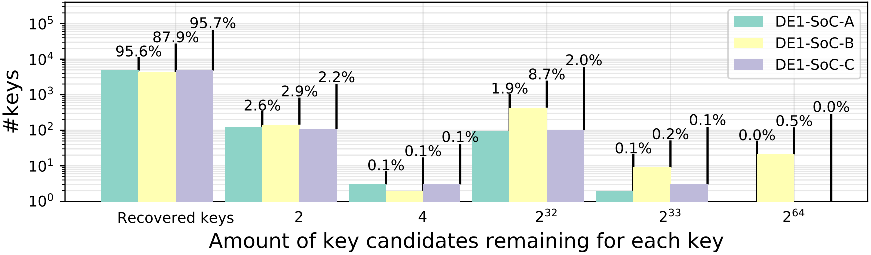
J. Krautter, D.R.E. Gnad and M.B. Tahoori

Key Recovery on 5000 random keys



- Experiments on DE1-SoC with best fault injection configuration
- Majority of 5000 keys can be recovered

Key Recovery on 5000 random keys



- Experiments on DE1-SoC with best fault injection configuration
- Majority of 5000 keys can be recovered
- Unrecovered keys due to **multi-byte faults**

Outline

Karlsruher Institut für Technologie

1. Background
2. Fault Injection and Analysis
3. Experimental Setup
4. Results
5. Discussion and Future Work
6. Conclusion

Discussion and Future Work

- Attack on fully constrained design on DE1-SoC with $< 50\%$ resources

Discussion and Future Work

Karlsruher Institut für Technologie

- Attack on fully constrained design on DE1-SoC with $<$ 50% resources
- Smaller DE0-Nano-SoC: Fully constrained design not vulnerable

  $\Rightarrow$ Not all devices are equally vulnerable

Discussion and Future Work

- Attack on fully constrained design on DE1-SoC with $< 50\%$ resources
- Smaller DE0-Nano-SoC: Fully constrained design not vulnerable

  $\Rightarrow$ Not all devices are equally vulnerable
- Alternatives to using ROs may exist

Discussion and Future Work

- Attack on fully constrained design on DE1-SoC with $<$ 50% resources
- Smaller DE0-Nano-SoC: Fully constrained design not vulnerable

  $\Rightarrow$ Not all devices are equally vulnerable
- Alternatives to using ROs may exist
- Attack may be extended to hard cores (ARM SoC)

FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Discussion and Future Work

- Attack on fully constrained design on DE1-SoC with $< 50\%$ resources
- Smaller DE0-Nano-SoC: Fully constrained design not vulnerable

  $\Rightarrow$ Not all devices are equally vulnerable
- Alternatives to using ROs may exist
- Attack may be extended to hard cores (ARM SoC)
- Possible **mitigation**:
  - Internal sensors
  - Bitstream checking
  - Voltage islands

Outline

1 Background

2 Fault Injection and Analysis

3 Experimental Setup

4 Results

5 Discussion and Future Work

6 Conclusion

Conclusion

- High precision fault injection on shared FPGAs is possible

Conclusion

- High precision fault injection on shared FPGAs is possible
- Logical isolation is not enough to prevent manipulation

Conclusion

- High precision fault injection on shared FPGAs is possible
- Logical isolation is not enough to prevent manipulation
- Threat model must be considered for FPGA multi-user environments

Conclusion

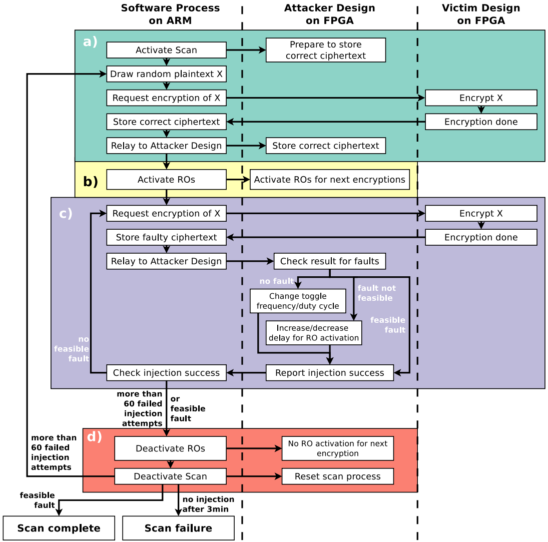
- High precision fault injection on shared FPGAs is possible
- Logical isolation is not enough to prevent manipulation
- Threat model must be considered for FPGA multi-user environments
- Mitigation may require new/modified hardware

Thank you for your attention!

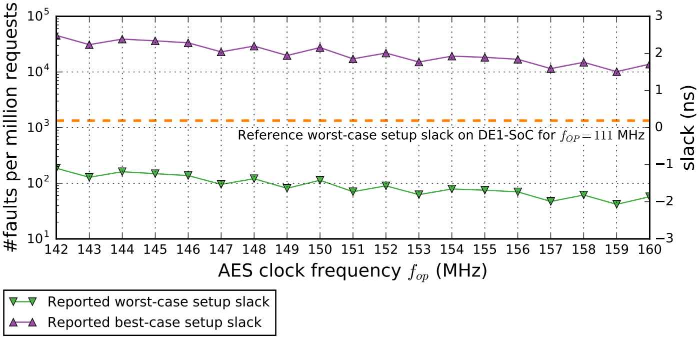
FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Additional Slides – Complete Scan Flow



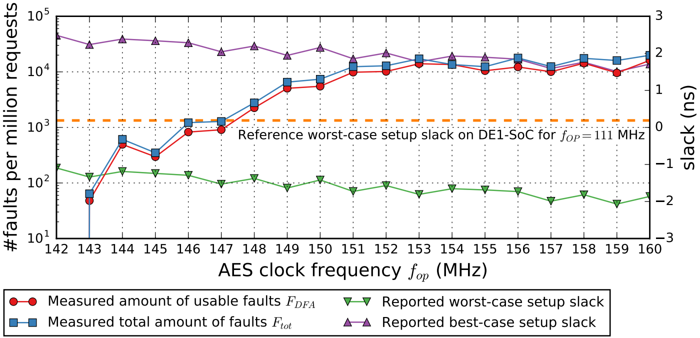
Additional Slides – Slack Dependent Analysis



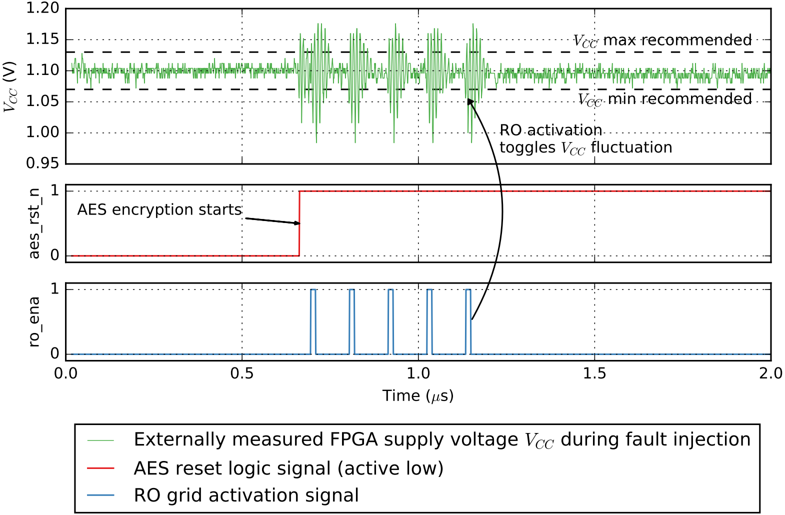
FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

J. Krautter, D.R.E. Gnad and M.B. Tahoori

Additional Slides – Slack Dependent Analysis



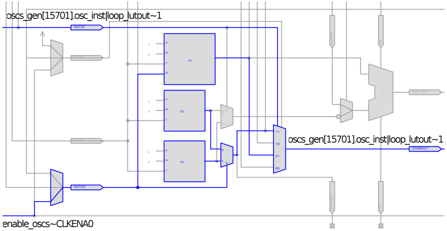
Additional Slides – Injection Process

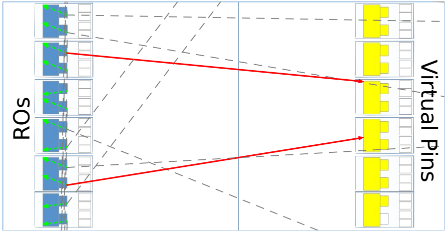


FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs

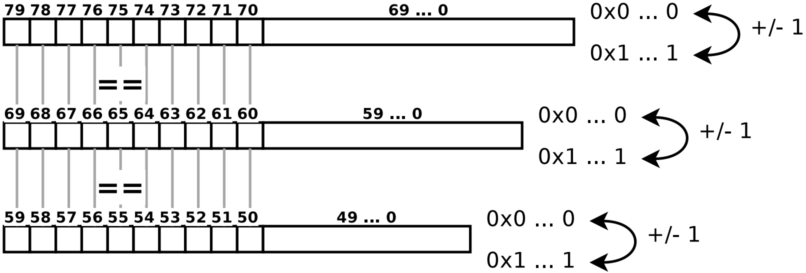
J. Krautter, D.R.E. Gnad and M.B. Tahoori

Additional Slides – RO Floorplan


oscs_gen[15701].osc_inst|loop_lutout~1
oscs_gen[15701].osc_inst|loop_lutout~1
enable_oscs~CLKENA0



Additional Slides – Adder Test Design


79 78 77 76 75 74 73 72 71 70
69 ... 0
0x0 ... 0
0x1 ... 1
+/- 1
==
69 68 67 66 65 64 63 62 61 60
59 ... 0
0x0 ... 0
0x1 ... 1
+/- 1
==
59 58 57 56 55 54 53 52 51 50
49 ... 0
0x0 ... 0
0x1 ... 1
+/- 1