# Analyzing the Fault Sensitivity of Secure Embedded Software

**Patrick Schaumont**

**Professor**
**Bradley Department of ECE**
**Virginia Tech**

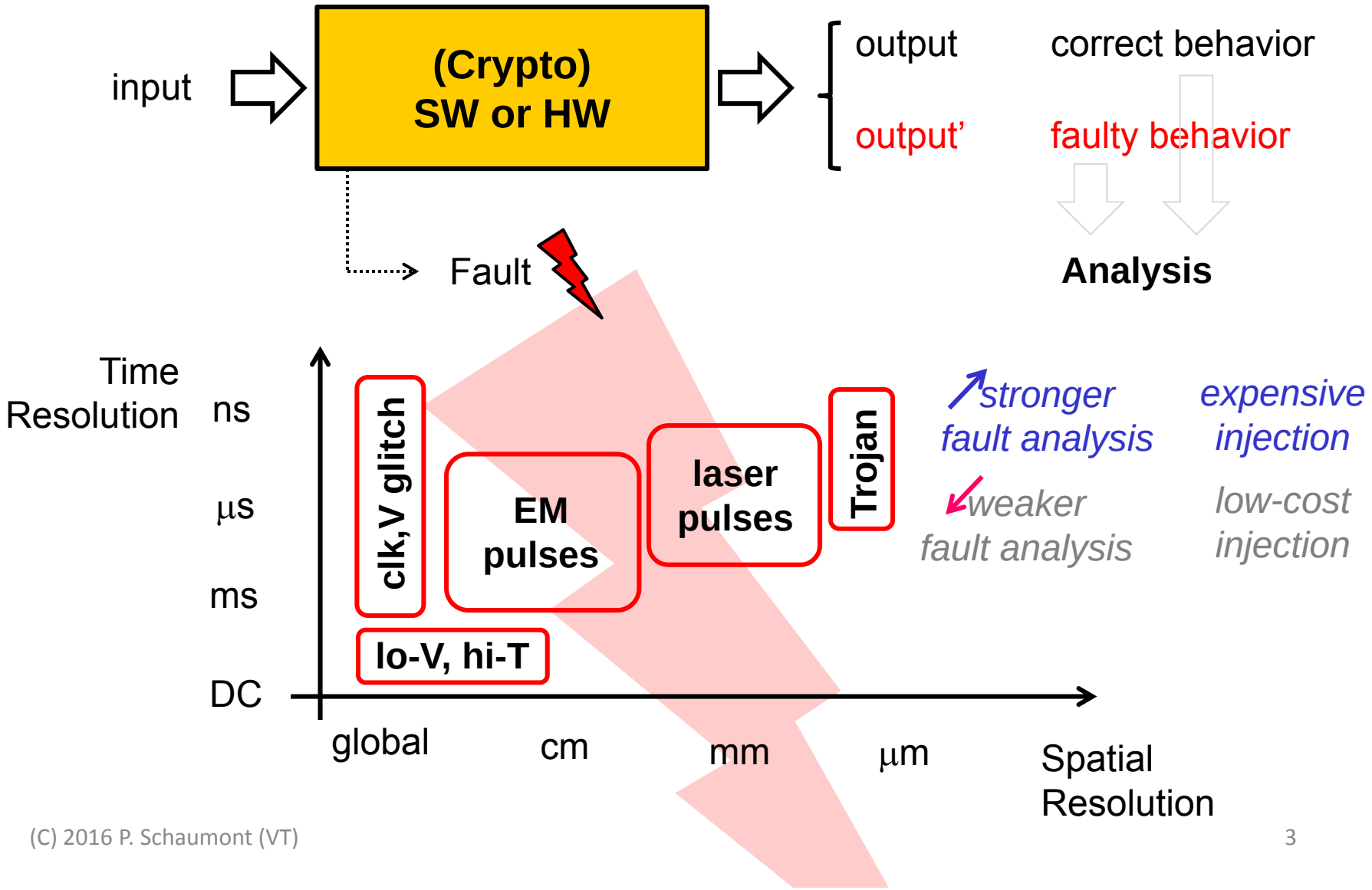
***Acknowledgments***
**National Science Foundation and SRC**

**Bilgiday Yuce, Nahid Farhady Ghalaty, Conor Patrick, Chinmay Despande, Marjan Ghodrati, Leyla Nazhandali**

# Outline

1. **Faults are a security liability**
2. **Faults as a side-channel - DFIA**
3. **Biased Fault Attacks on Software**
4. **Breaking Software Fault Countermeasures**
5. **Outlook**

# The Fault Attack Principle



# Why are Faults a Security Issue?

- **May enable external control of execution**
  - **Denial of service**
  - **Control of critical decisions**

```
if (! access_allowed)
   abort( );
```

**instruction_skip**

# Why are Faults a Security Issue?

- **May enable external control of execution**
  - **Denial of service**
  - **Control of critical decisions**

```
if (! access_allowed)
   abort( );
```

**instruction_skip**

- **May cause information leakage of secrets**

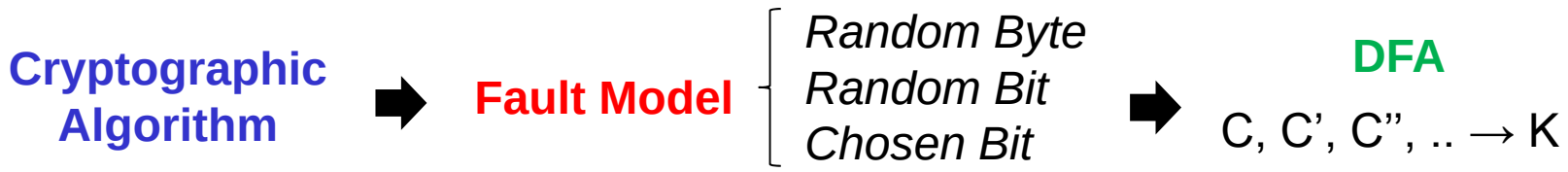
```
if (key_bit)
   r1 = r1 + 1;
else
   r0 = r0 + 1;

out = f(r1);
```

**fault in r1**

**key_bit *leaks indirectly via* out**

# Classic Differential Fault Analysis



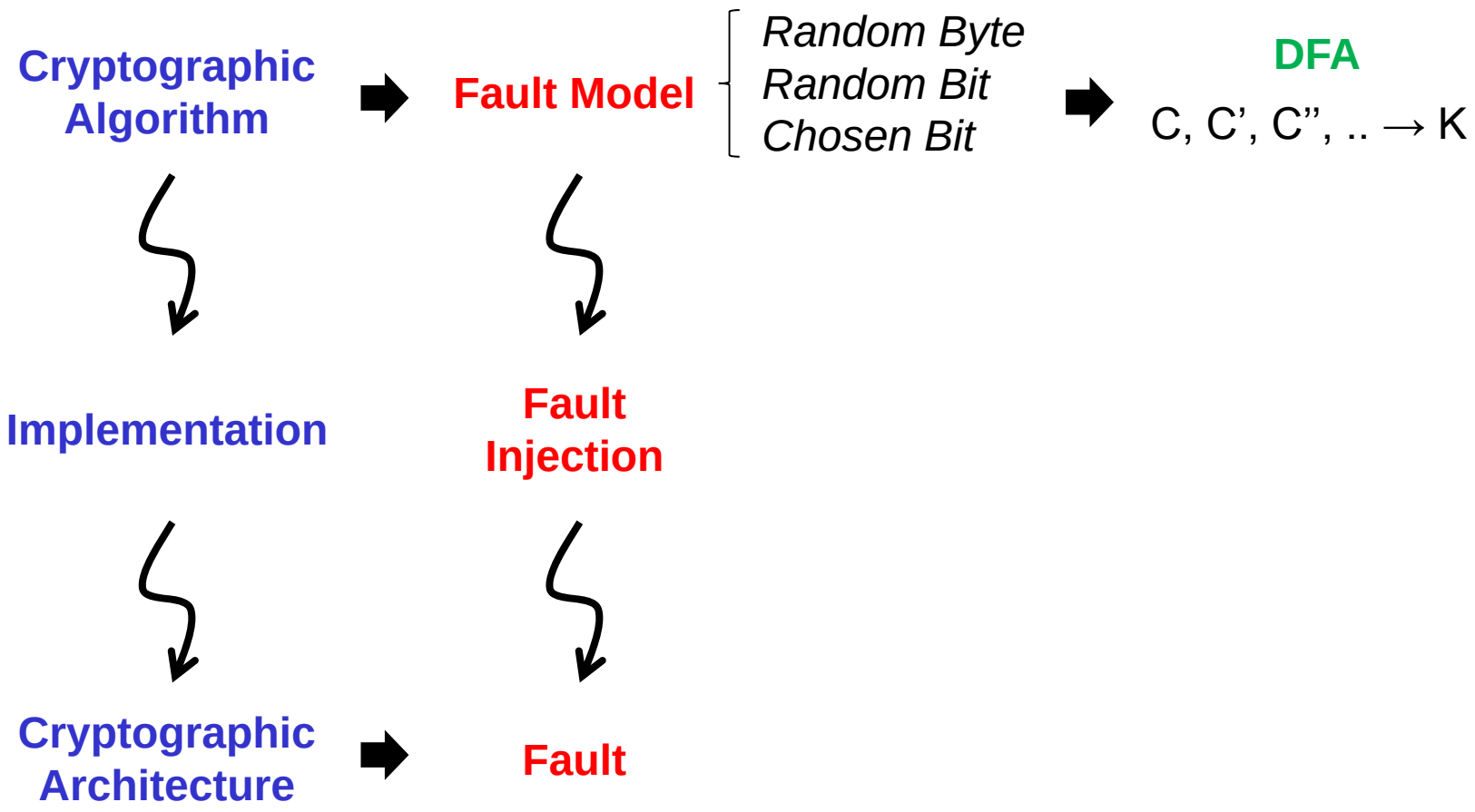

**[TM 2010] Single random byte fault at 8th round of AES-128: Key $2^{128}$ → $2^{12}$**
**[LGS+ 2010] Two seq. byte fault at 9th, 10th round of AES-192: Key $2^{128}$ → 1**

**Current DFA methods are quite good**

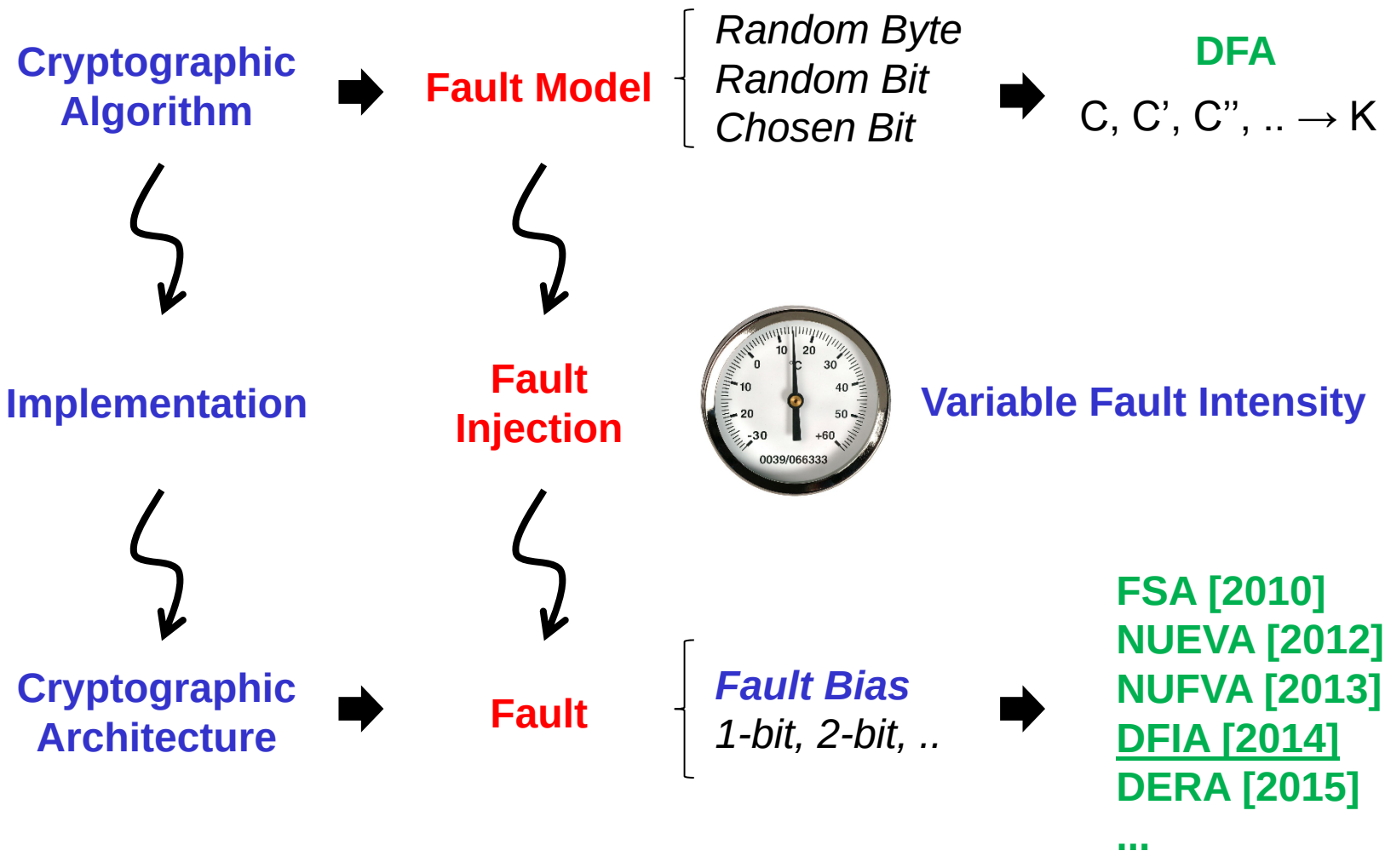
**IF**

**the fault model can be realized**

# Implementations and Actual Faults



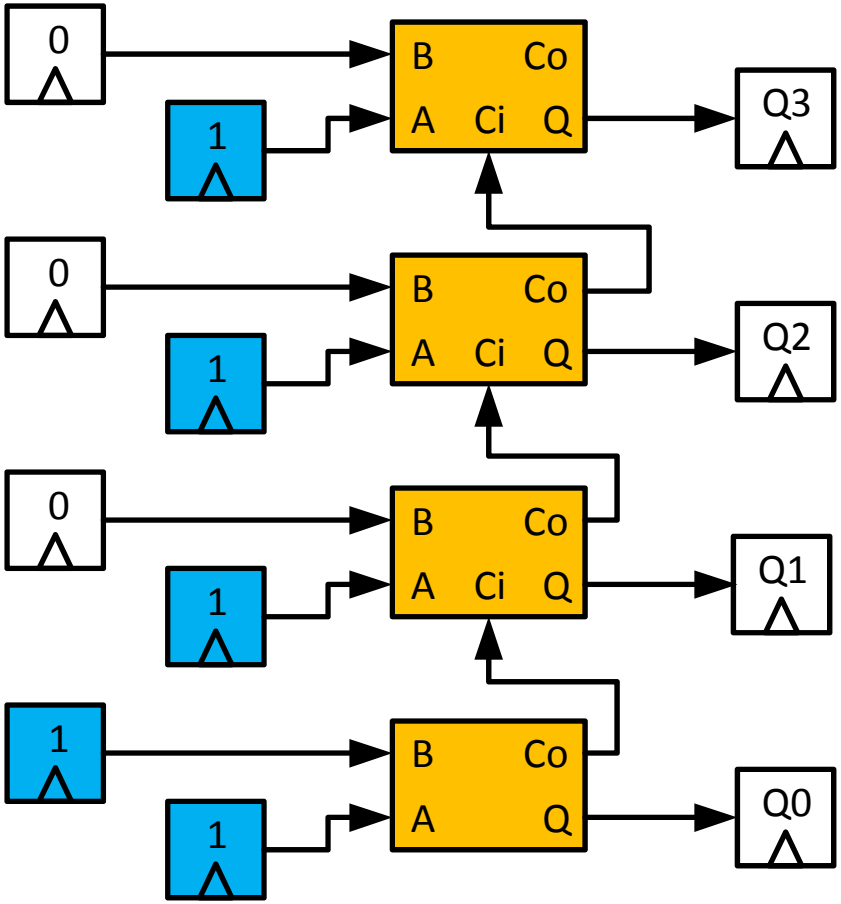
# Biased Fault Attacks



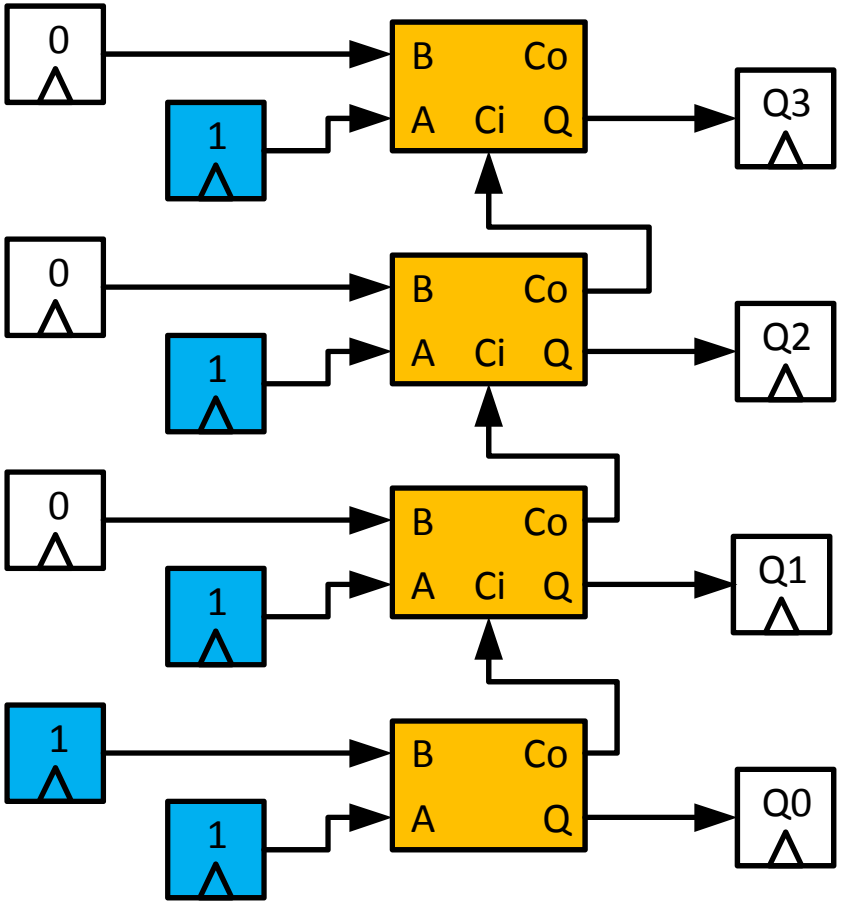
# Outline

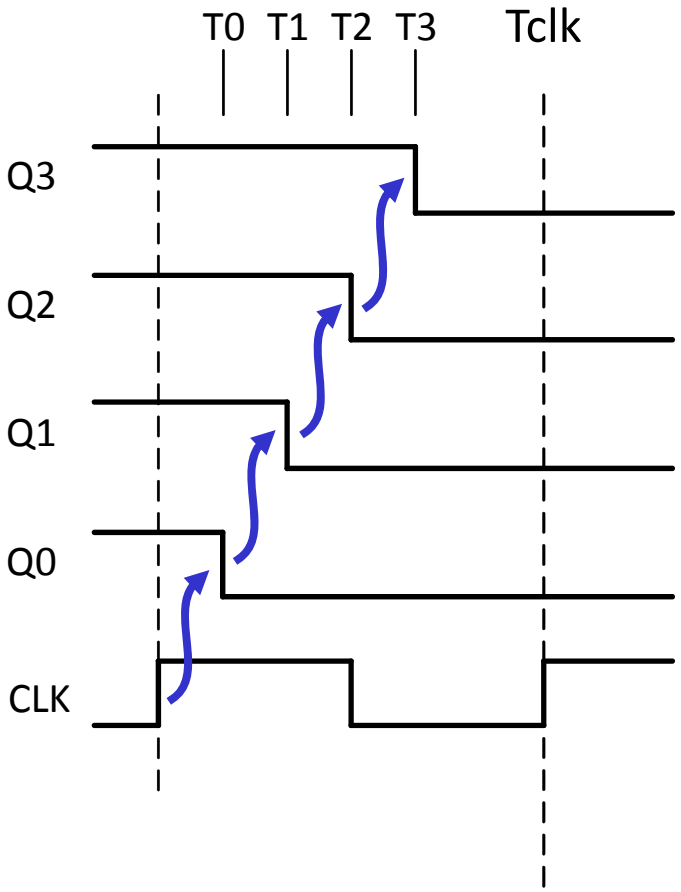
1. Faults are a security liability
2. Faults as a side-channel - DFIA
3. Biased Fault Attacks on Software
4. Breaking Software Fault Countermeasures
5. Outlook

# Do Biased Faults Exist?

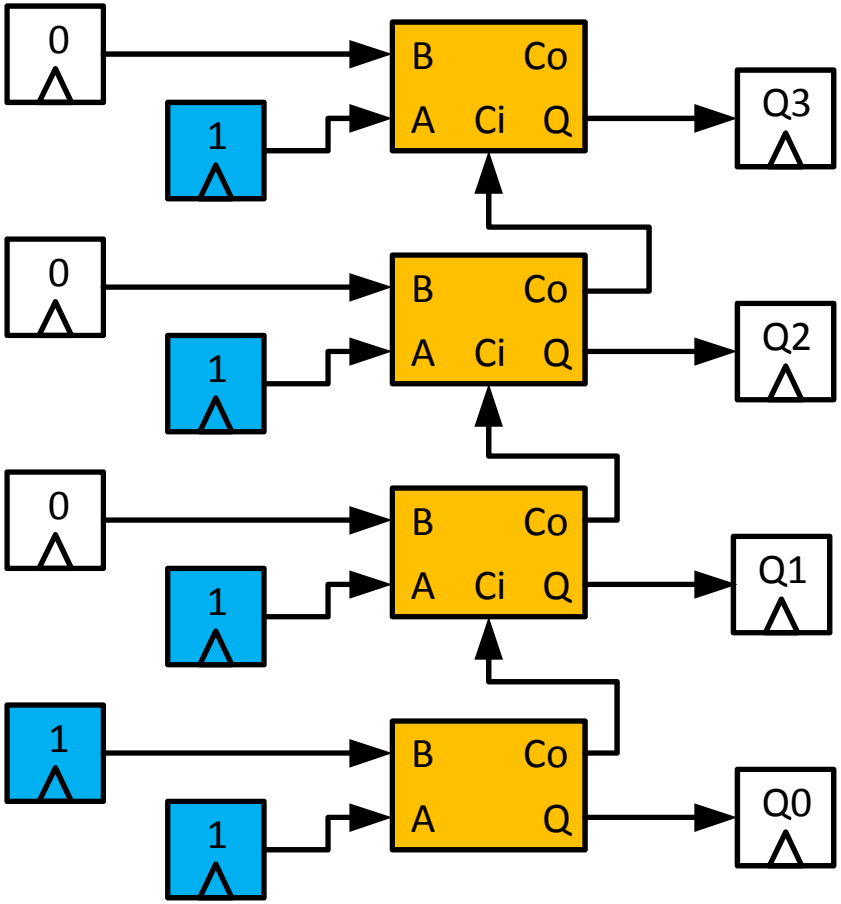


# Do Biased Faults Exist?

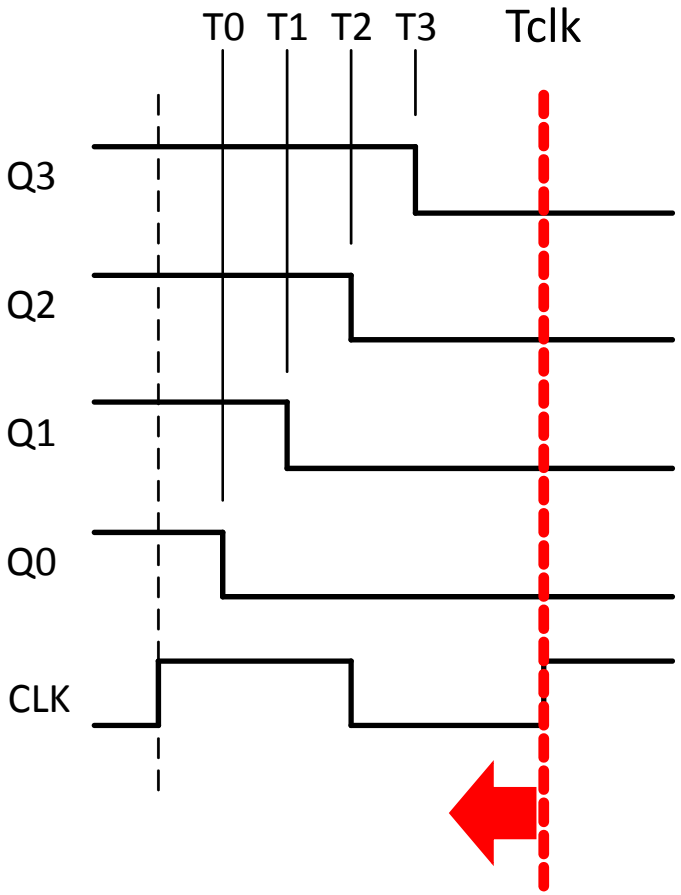


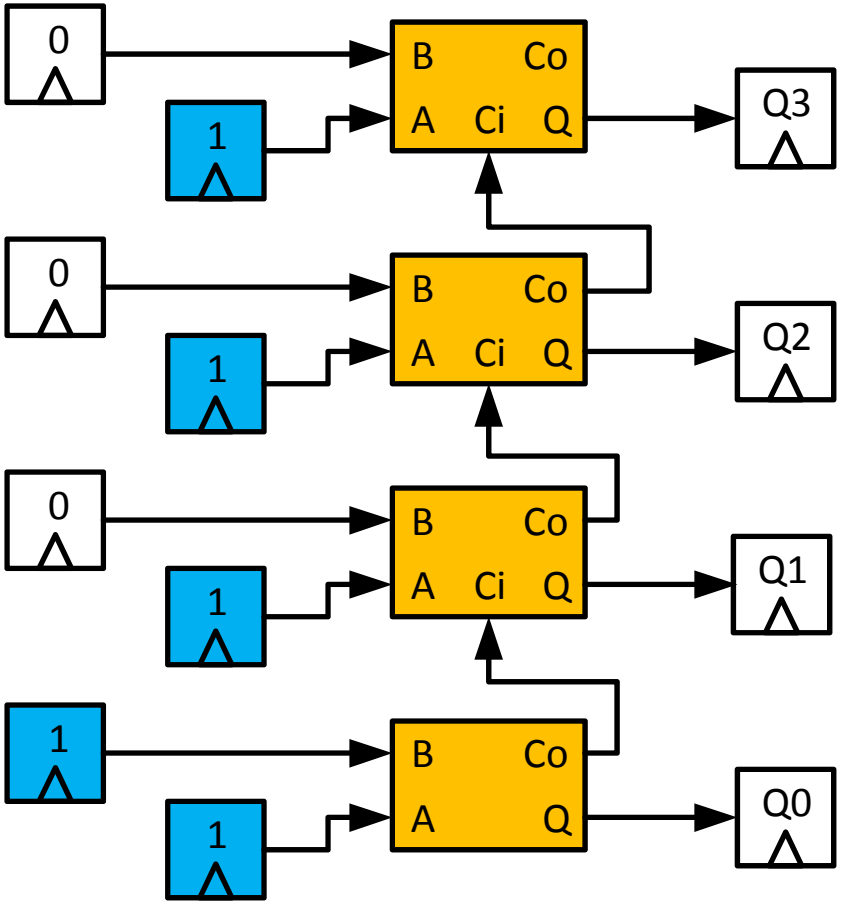
# Yes, Biased Faults Exist



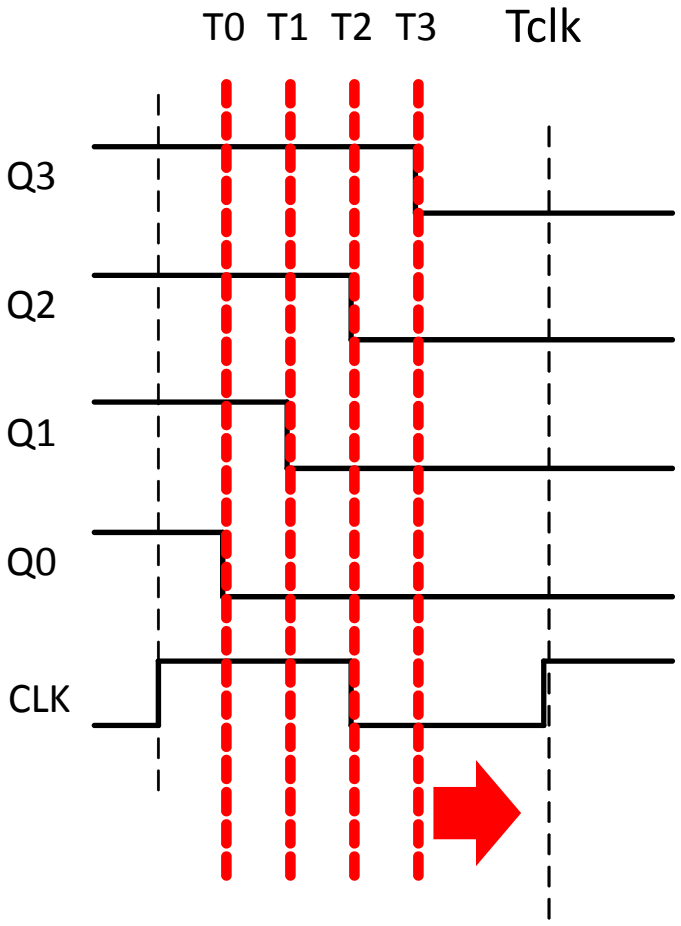

**Clock Glitching**



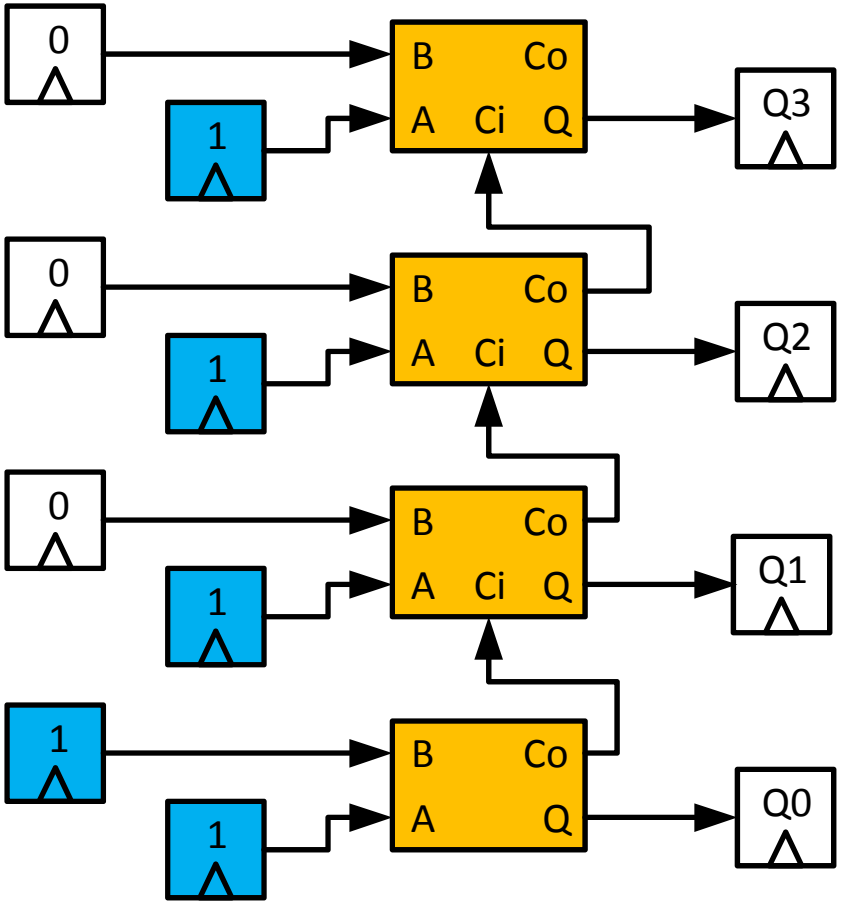
# Yes, Biased Faults Exist



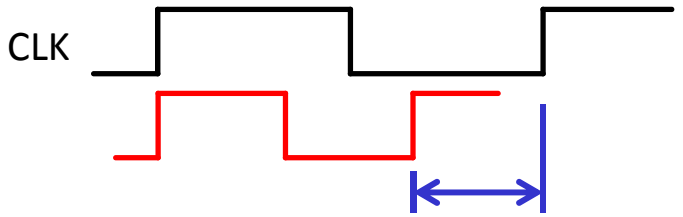

**Voltage Starving**



# Fault Intensity, Bias and Sensitivity




## Fault Intensity




## Fault Bias

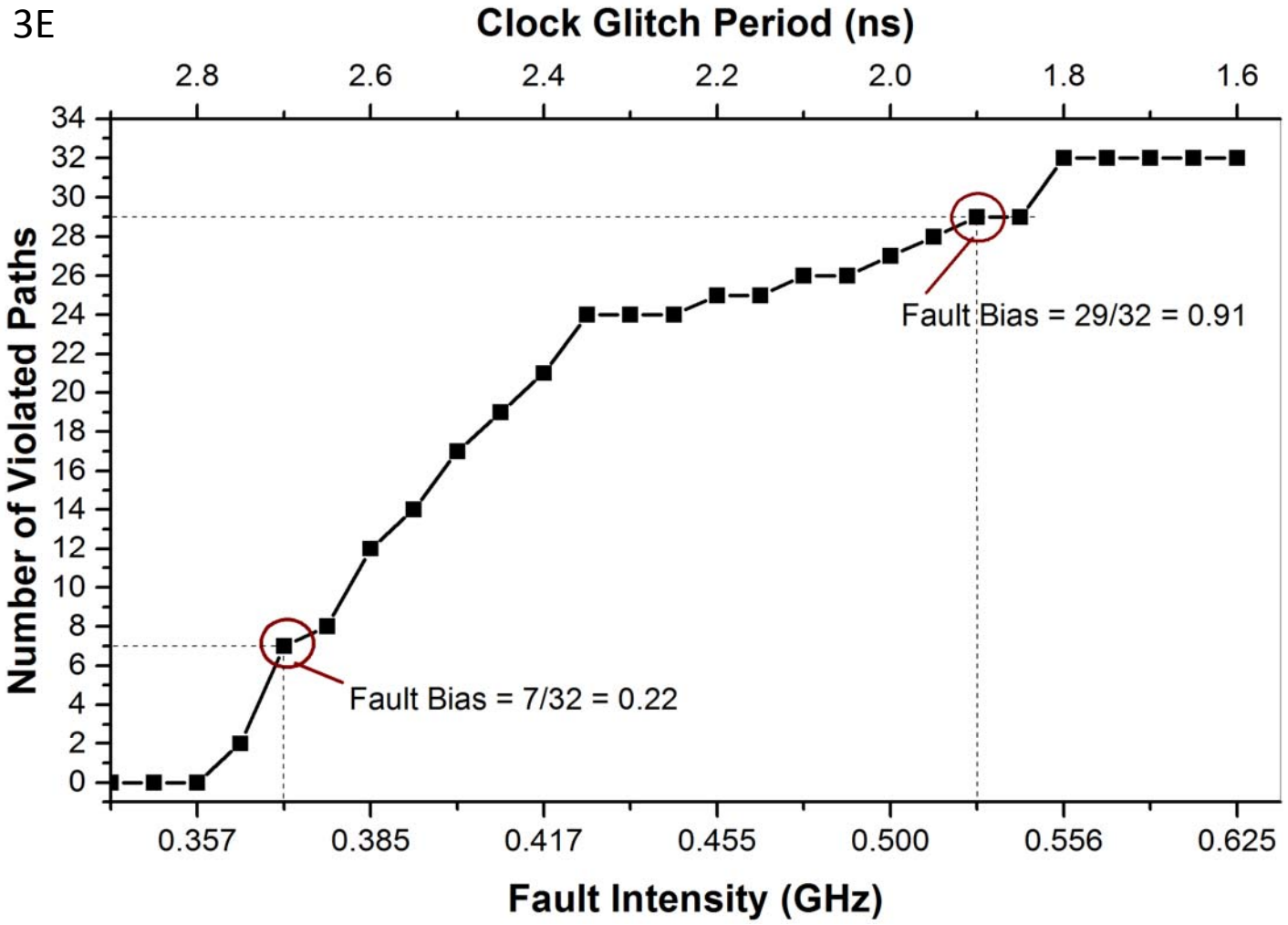
$$FB(i) = \frac{\#violated\_paths}{\#total\_paths}\Bigg|_{FI = i}$$

## Fault Sensitivity

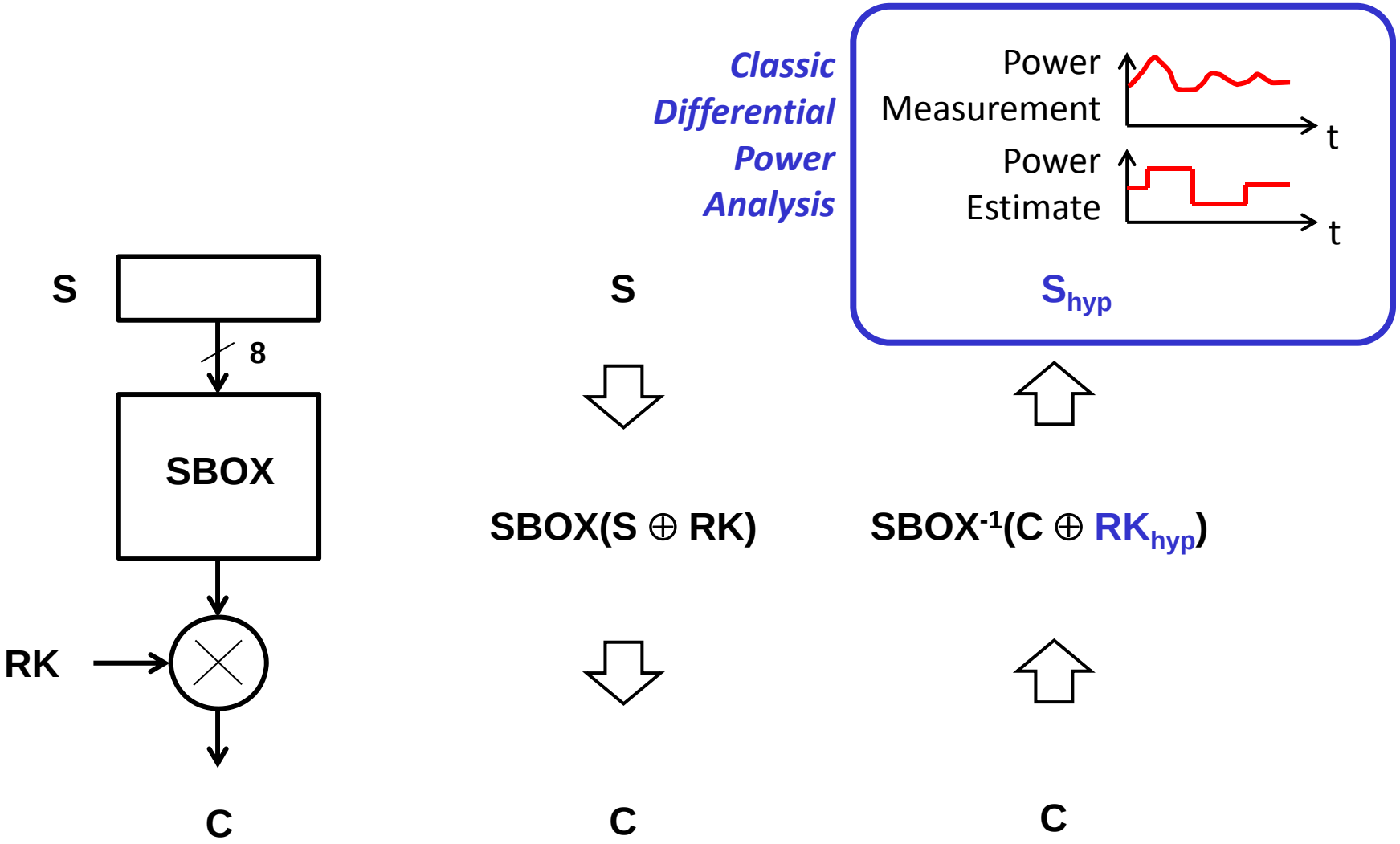
FS = i for which FB(i) < ε

# Fault Bias as function of Fault Intensity

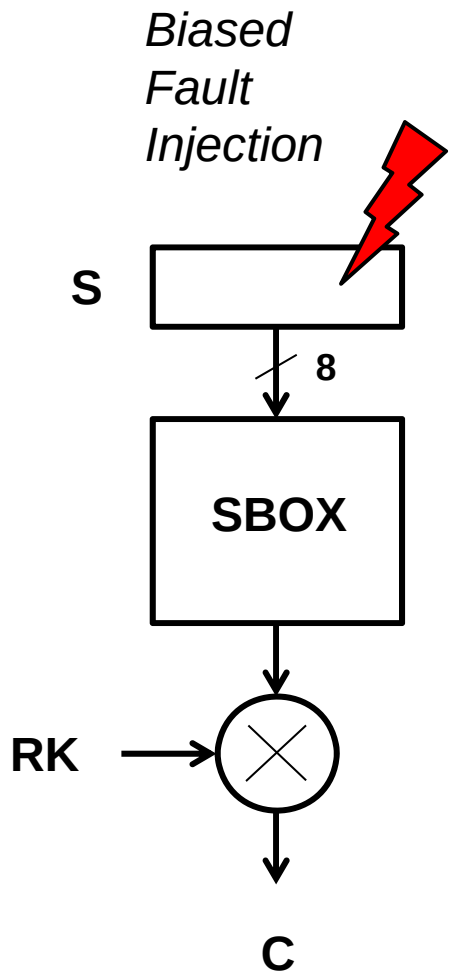
32-bit ripple carry adder
Spartan 3E

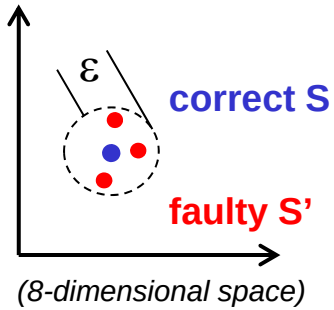


# Biased Faults as a Side Channel



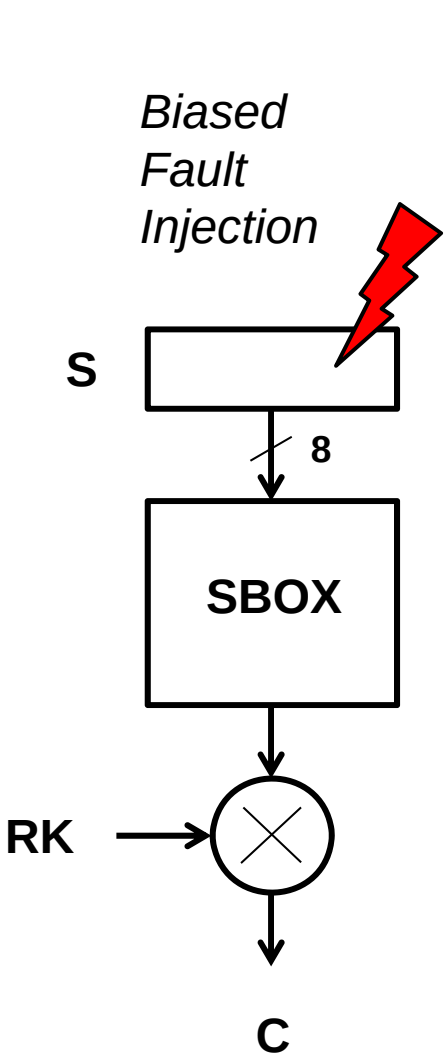
# Biased Faults as a Side Channel



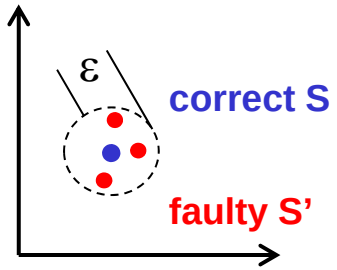




C'

# Biased Faults as a Side Channel



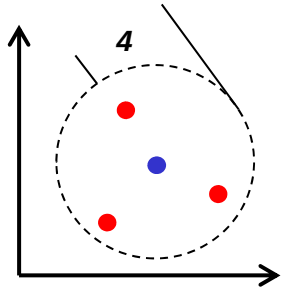

**Under Correct Key Hypothesis**




$SBOX^{-1}(C' \oplus RK_{hyp})$

C'

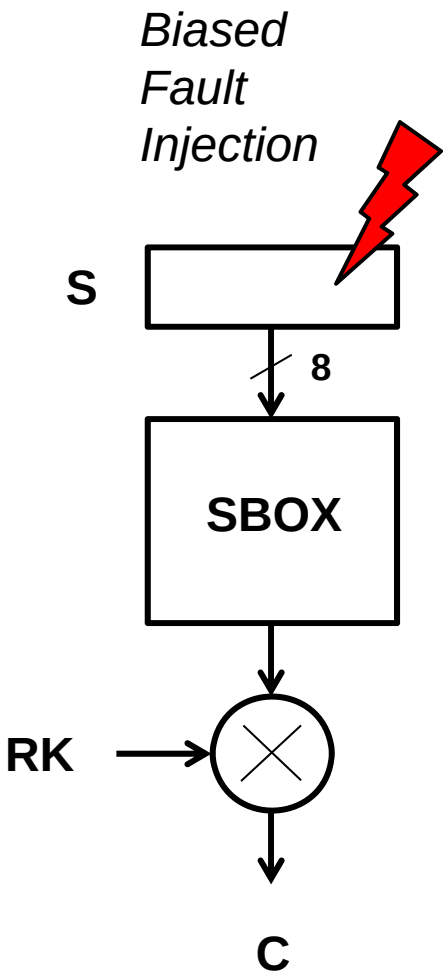
**Under Wrong Key Hypothesis**




$SBOX^{-1}(C' \oplus RK_{hyp})$

C'

# Differential Fault Intensity Analysis




***Differential Fault Intensity Analysis***

1. Inject Faults at different Fault Intensities
   $HW(S \oplus S') < \varepsilon$
2. Collect Fault Ciphertext C'
3. For all Key hypothesis $RK_{hyp}$ compute
   $S_{i,RK} = SBOX^{-1}(C' \oplus RK_{hyp})$
4. Select RK for which

   **$RK = ArgMin(\sum_i \sum_j HD(S_{i,RK}, S_{j,RK}))$**

# DFIA versus DFA

**DFA**

- **makes a precise assumption on the injected fault**
- **needs a system of equations to resolve key guess**

**DFIA**

- **makes an approximate model of the injected fault**
- **uses max likelihood testing to resolve key guess**

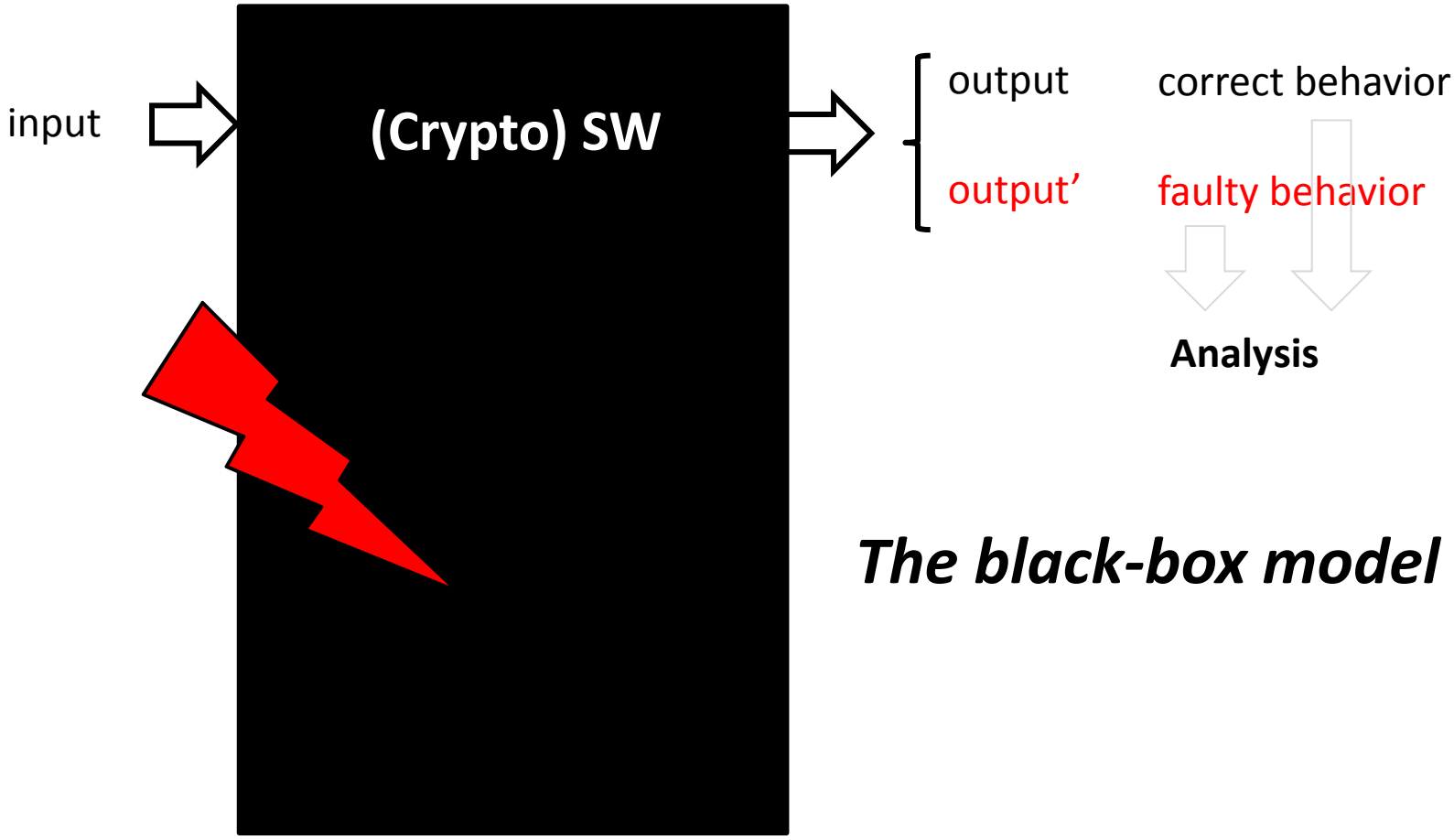
**DFIA relaxes the fault model requirements and is more suitable when fault injection is hard to control**

*Relevant publications [DATE14] [FDTC14] [COSADE15] [IEEE ESL16]*

# Outline

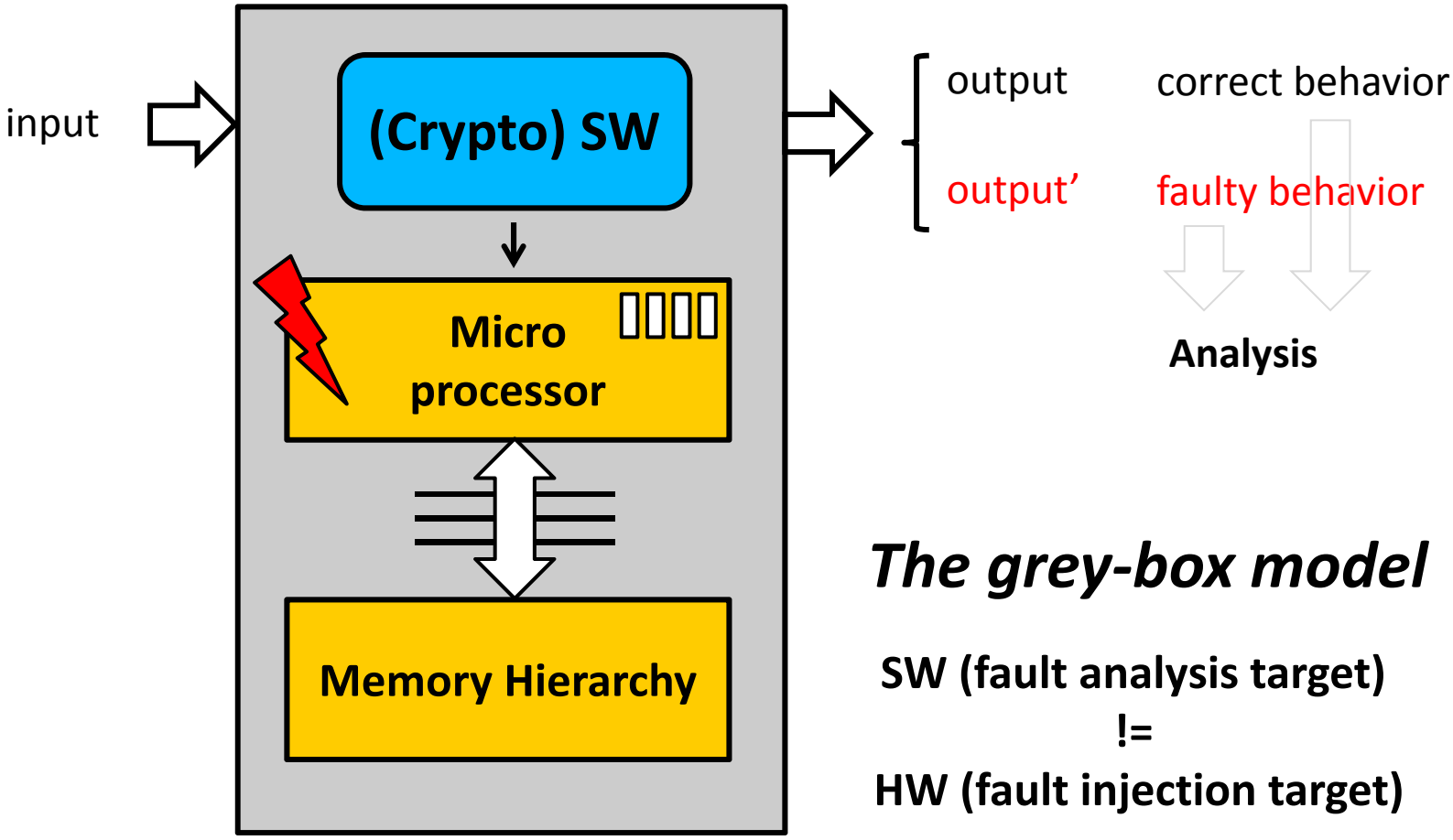
1. Faults are a security liability
2. Faults as a side-channel - DFIA
3. Biased Fault Attacks on Software
4. Breaking Software Fault Countermeasures
5. Outlook

# Fault Attacks on Software




***The black-box model***

# Fault Attacks on Software

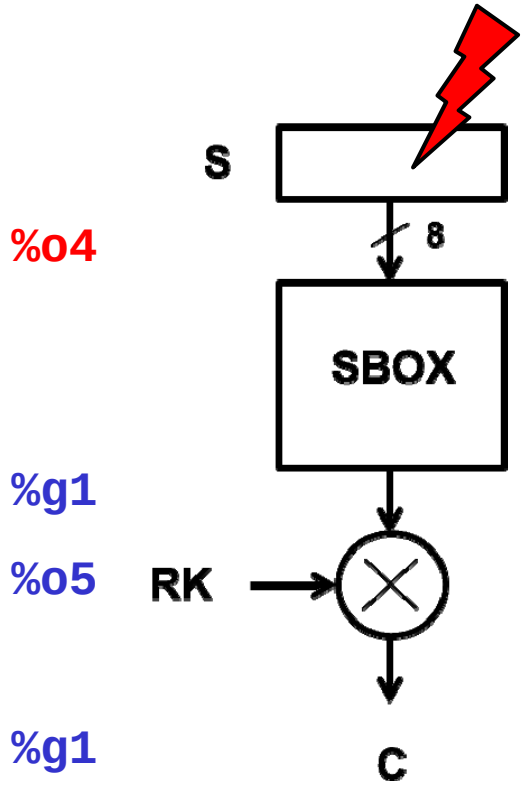



***The grey-box model***

**SW (fault analysis target)**
**!=**
**HW (fault injection target)**

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```



# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | ST | XOR | | LD3 | LD2 |
| | | | | | | |
| | | | | | | |

# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | ST | XOR | | LD3 | LD2 |
| | | | ST | XOR | | LD3 |
| | | | | | | |

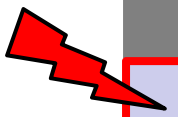
# In-order RISC Pipeline Example

```
ld      [%o3 + 0xb], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | ST | XOR | | LD3 | LD2 |
| | | | ST | XOR | | LD3 |
| | | | | ST | XOR | |

# Fault Injection (FI) Observations

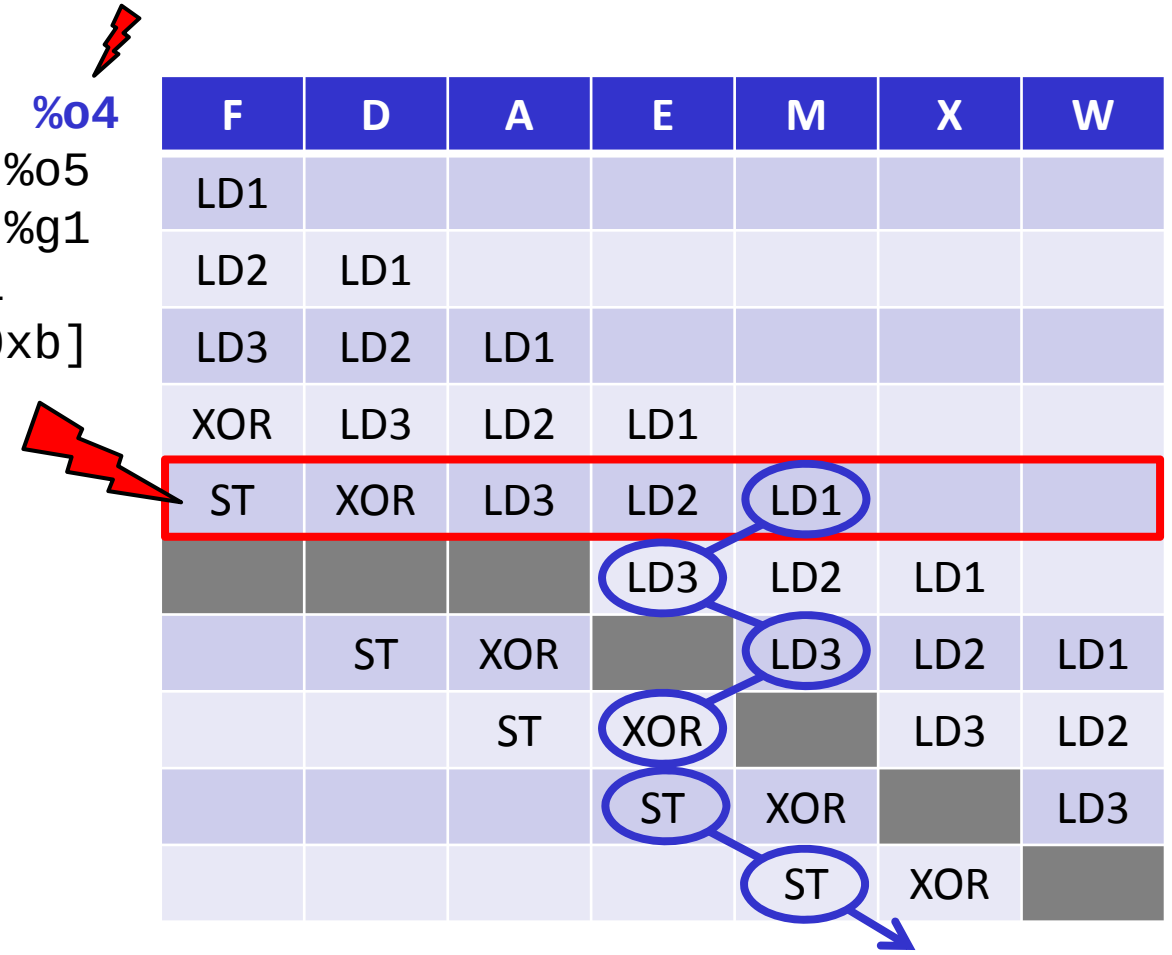
1. **FI affects multiple instructions**
2. **Pipeline hazards affect sensitivity**
3. **FI effect depends on pipeline stage**
4. **Fault sensitivity depends on**
   - **Instruction**
   - **Pipeline Stage**



| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | ST | XOR | | LD3 | LD2 |
| | | | ST | XOR | | LD3 |
| | | | | ST | XOR | |

# DFIA on SBOX access

```
ld      [%o3 + 0xb0], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

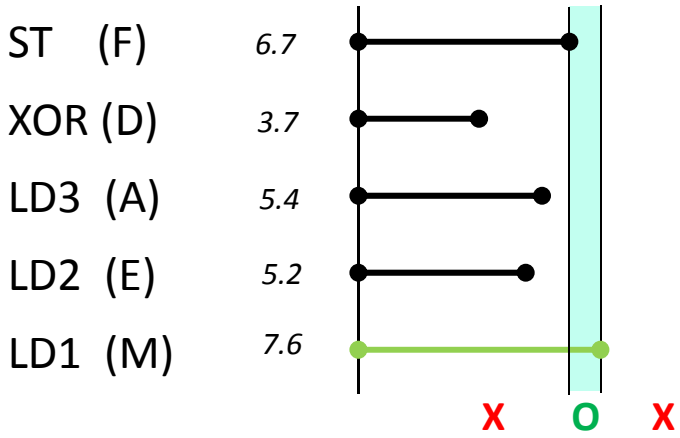


# DFIA on SBOX access

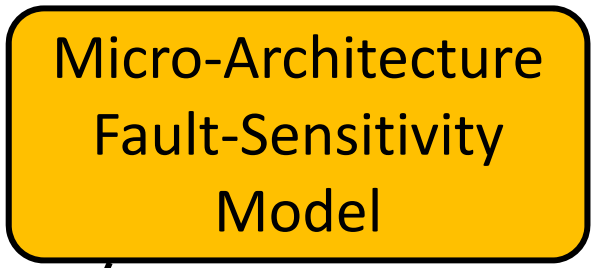
```
ld      [%o3 + 0xb0], %o4
ldub    [%o0 + 0xb], %o5
ldub    [%o4 + 0xb], %g1
xor     %g1, %o5, %g1
stb     %g1, [%o3 + 0xb]
```

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| LD3 | LD2 | LD1 | | | | |
| XOR | LD3 | LD2 | LD1 | | | |
| ST | XOR | LD3 | LD2 | LD1 | | |
| | | | LD3 | LD2 | LD1 | |
| | ST | XOR | | LD3 | LD2 | LD1 |
| | | ST | XOR | | LD3 | LD2 |
| | | | ST | XOR | | LD3 |
| | | | | ST | XOR | |

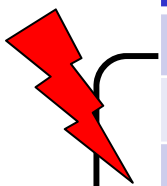
**FI Parameters**



# Micro-Architecture Fault Sensitivity Model

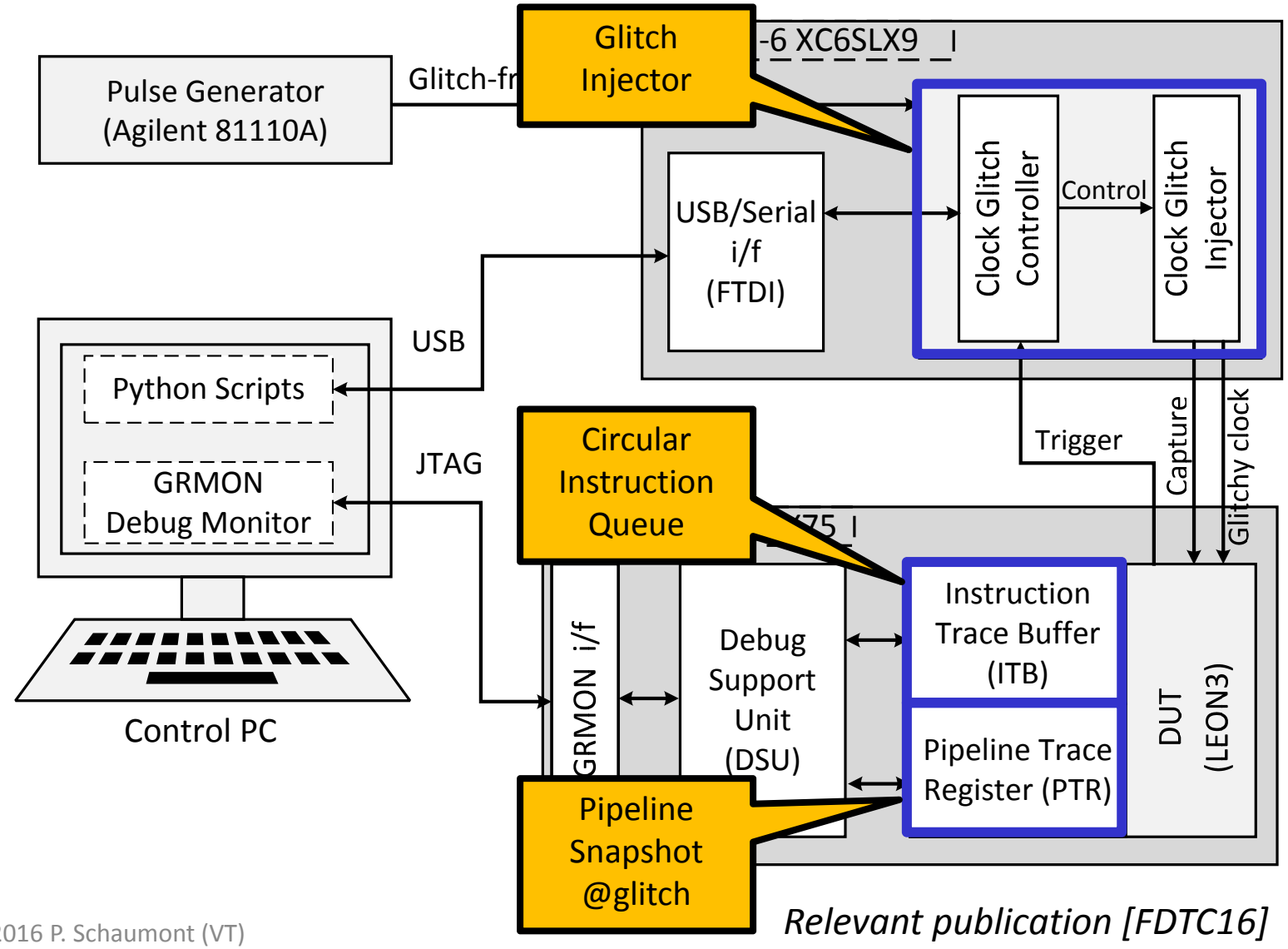




Timing = f(cycle, instruction)



| | F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|---|
| STB | 6.7 | 3.2 | 9.1 | 7 | 5.4 | 4.9 | 0 |
| LD | 5.7 | 3.4 | 5.4 | 5.2 | 7.6 | 2.8 | 5.6 |
| LDUB | 7 | 3.6 | 6.5 | 5.1 | 7.2 | 2.6 | 4.5 |
| XOR | 5.5 | 3.7 | 6.5 | 5.1 | 0 | 3.3 | 4.9 |

# Measurement & Verification Setup




*Relevant publication [FDTC16]*

# Using the μArch Fault Sensitivity Model

**DFIA attack with/without FS Model**

- **with/without fault sensitivity model**
- **on AES-SBOX and TBOX software**

**→ 10x reduction of fault injection space**

*@160ps resolution*

| | Design | Glitch Span Fault Intensity | # Cycles | # Fault Locations |
|---|---|---|---|---|
| Black-Box (wo model) | AES-SBOX | 3 – 15.8 | 13 | 1040 |
| | AES-TBOX | 3 – 15.8 | 16 | 1280 |
| with model | AES-SBOX | 4.7 – 6.9 | 6 | 90 |
| | AES-TBOX | 5.4 – 6.6 | 9 | 81 |

*Relevant publication [FDTC15]*

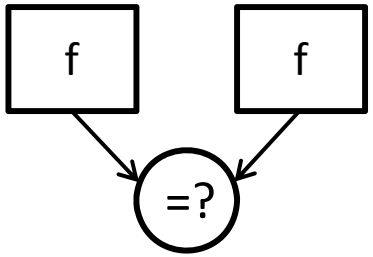
# Outline

1. Faults are a security liability
2. Faults as a side-channel - DFIA
3. Biased Fault Attacks on Software
4. Breaking Software Fault Countermeasures
5. Outlook

# Software Fault-Attack Countermeasures

## Software Countermeasure Fault Injection

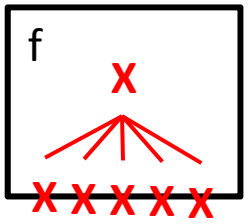
### Detection based




Information Redundancy
Algorithmic Redundancy

**at instruction-level:**
- **instruction duplication**
- **instruction triplication**
- **parity-invariant**

### Infection based




Currently, mostly broken ..

# Instruction Duplication Countermeasure

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```

- **Duplicated execution of instructions, compare**
- **Breaking countermeasure requires back-to-back fault injection - considered difficult**
- **Micro-architecture Fault Sensitivity Model can pin down the weak spot of this countermeasure**

# Analyzing Instruction Duplication

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```

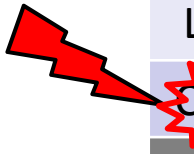
**data hazard**

**branch interlock hazard**

| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| CMP | LD2 | LD1 | | | | |
| | | LD2 | LD1 | | | |
| BNE | CMP | | LD2 | LD1 | | |
| | | CMP | | LD2 | LD1 | |
| | | | CMP | | LD2 | LD1 |
| | BNE | | | CMP | | LD2 |
| | | BNE | | | CMP | |
| | | | BNE | | | CMP |

# Attack Scenarios

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```



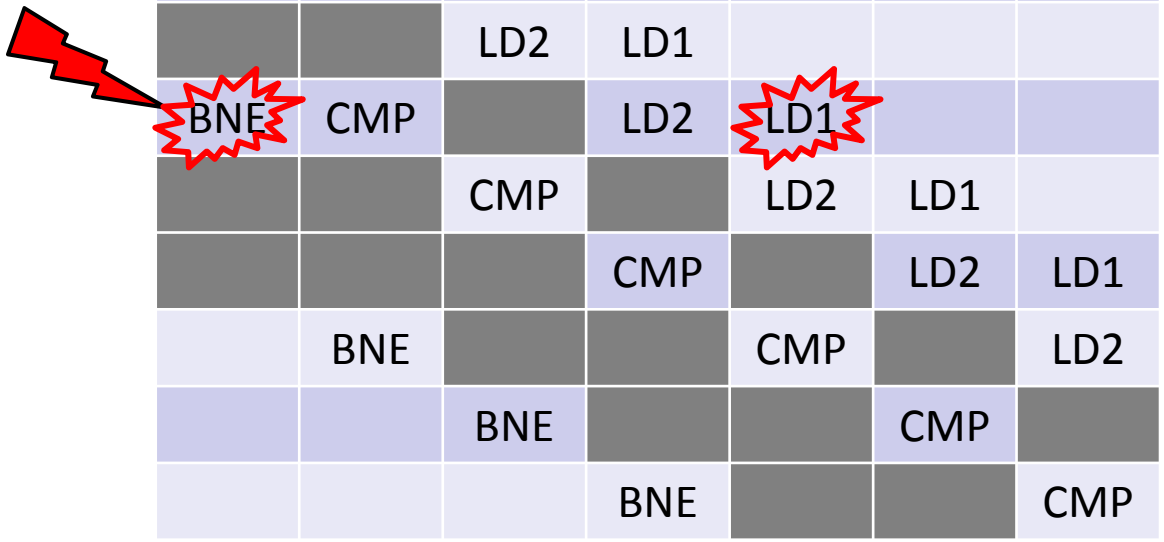
| F | D | A | E | M | X | W |
|---|---|---|---|---|---|---|
| LD1 | | | | | | |
| LD2 | LD1 | | | | | |
| CMP | LD2 | LD1 | | | | |
| | | LD2 | LD1 | | | |
| BNE | CMP | | LD2 | LD1 | | |
| | | CMP | | LD2 | LD1 | |
| | | | CMP | | LD2 | LD1 |
| | BNE | | | CMP | | LD2 |
| | | BNE | | | CMP | |
| | | | BNE | | | CMP |

**Scenario 1 (single-glitch):**

1. Instruction Fault in CMP
   CMP → NOP
2. Computation Fault in LD1
   (Biased fault)

# Attack Scenarios

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```




**Scenario 2 (single glitch):**

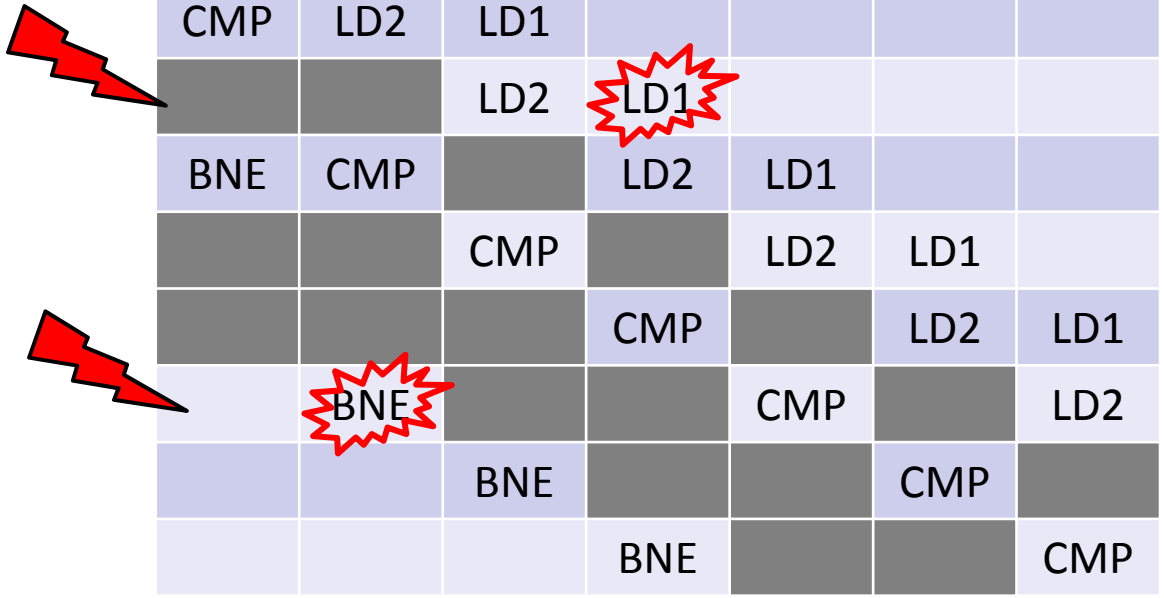
1. Instruction Fault in BNE
   BNE → NOP
2. Computation Fault in LD1
   (Biased fault)

# Attack Scenarios

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```

**Scenario 3 (multi-glitch):**

1. Computation Fault in LD1
   (Biased fault)
2. Instruction Fault in BNE
   BNE → NOP



# Verification on Prototype

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```

| | Glitch FI (ns) | Impacted Instruction | Fault Effect |
|---|---|---|---|
| Scenario 1 | 25 - 33 | LD1 (A)<br>CMP (D) | Faulty %g2<br>CMP → SRL |
| Scenario 2 | 32 - 38 | LD1 (M)<br>BNE (F) | Faulty %g2<br>BNE → NOP |
| Scenario 3 | 38<br>34 - 36 | LD1 (E)<br>BNE (D) | Faulty %g2<br>BNE → NOP |

→ All scenarios break duplication countermeasure

*Relevant publication [FDTC16]*

# Outline

1. Faults are a security liability
2. Faults as a side-channel - DFIA
3. Biased Fault Attacks on Software
4. Breaking Software Fault Countermeasures
5. Outlook

# Summary

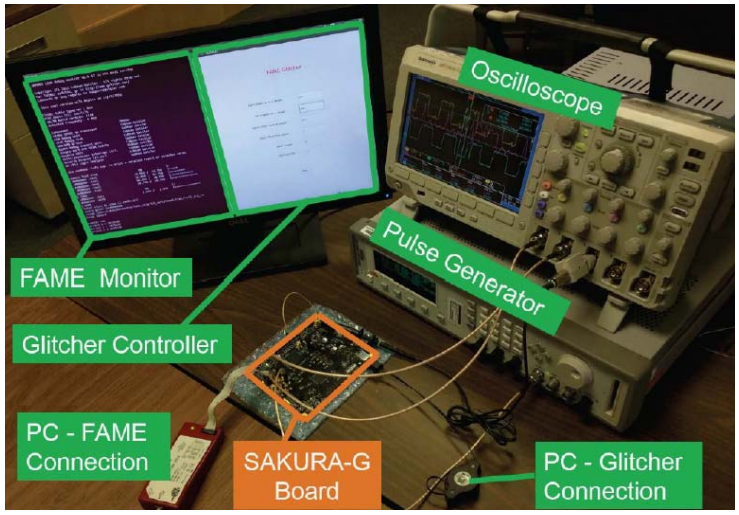
- **Fault Models not just cryptographer's imagination**
  - **Fault effects have physical causes and can be understood by a computer engineer**
  - **Insight into the fault effect leads to better fault attack**

- **Existing processors:**
  **Can software countermeasures be improved?**
  - **Yes: improve redundancy using bitslicing**

*Relevant publication [SAC16]*

# Summary

- **New Processors:**
  **Can we integrate countermeasures into the μArch?**
  - **FAME – Fault-attack Aware Microprocessor Extension**
  - **Fault Detection with Hardware Sensors;**
    **Micro-architectural support for state recovery;**
    **Software Trap handler to implement fault response**
  - **Design Details**
    **SRC e-Workshop 2/26**
    **HASP 2016 paper**
  - **Planned Tape-out 9/16**
  - **Design Report**
    **SRC Review 9/27/16**



# References


| | |
|---|---|
| **[DATE14]** | Nahid Farhady Ghalaty, Aydin Aysu, and Patrick Schaumont, "*Analyzing and Eliminating the Causes of Fault Sensitivity Analysis*", Design, Automation & Test in Europe (DATE 2014), Dresden, Germany, March 2014 (6 pages).doi: 10.7873/DATE.2014.217. |
| **[FDTC14]** | Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont, "*Differential Fault Intensity Analysis*," 11th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2014), 49-58, Busan, Korea, September 2014. doi: 10.1109/FDTC.2014.15. |
| **[COSADE15]** | Nahid Farhady Galathy, Bilgiday Yuce and Patick Schaumont, "*Differential Fault Intensity Analysis on PRESENT and LED Block Ciphers*," 6th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2015), 174-188, Berlin, Germany, April 2015.doi: 10.1007/978-3-319-21476-4_12. |
| **[IEEE ESL 16]** | Nahid Farhady Ghalaty, Bilgiday Yuce, Patrick Schaumont, "*Analyzing the Efficiency of Biased-Fault Based Attacks*," IEEE Embedded Systems Letters, 8(2):33-36, 2016. doi: 10.1109/LES.2016.2524652. |
| **[FDTC15]** | Bilgiday Yuce, Nahid Farhady Galathy and Patrick Schaumont, "*Improving Fault Attacks on Embedded Software using RISC Pipeline Characterization*," 12th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC-2015), St Malo, France, September 2015. doi: 10.1109/FDTC.2015.16. |
| **[FDTC16]** | Bilgiday Yuce, Nahid Farhady Ghalaty, Harika Santapuri, Chinmay Deshpande, Conor Patrick, Patrick Schaumont, "*Software Fault Resistance is Futile: Effective Single-glitch Attacks*," Fault Diagnosis and Tolerance in Cryptography (FDTC 2016), Santa Barbara, CA, August 2016. |
| **[SAC16]** | Conor Patrick, Bilgiday Yuce, Nahid Farhady Ghalaty, Patrick Schaumont, "*Lightweight Fault Attack Resistance in Software Using Intra-Instruction Redundancy*," Selected Areas in Cryptography (SAC 2016), St. John's, Canada, August 2016. |
| **[HASP16]** | Bilgiday Yuce, Nahid Farhady Ghalaty, Chinmay Deshpande, Conor Patrick, Leyla Nazhandali and Patrick Schaumont, "*FAME: Fault-attack Aware Microprocessor Extensions for Hardware Fault Detection and Software Fault Response*," ACM Hardware and Architectural Support for Security and Privacy (HASP) 2016, Seoul, Korea, June 2016. doi: 10.1145/2948618.2948626. |

Thank you for your attention!
I'll be happy to answer your questions.

Patrick Schaumont
schaum@vt.edu