Faults in Our Bus: Novel Bus Fault Attack to Break ARM TrustZone

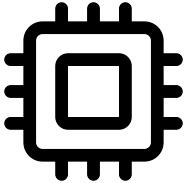
Nimish Mishra, Anirban Chakraborty, Debdeep Mukhopadhyay

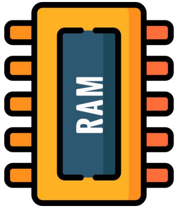
Secured Embedded Architecture Laboratory
Indian Institute of Technology, Kharagpur

Outline

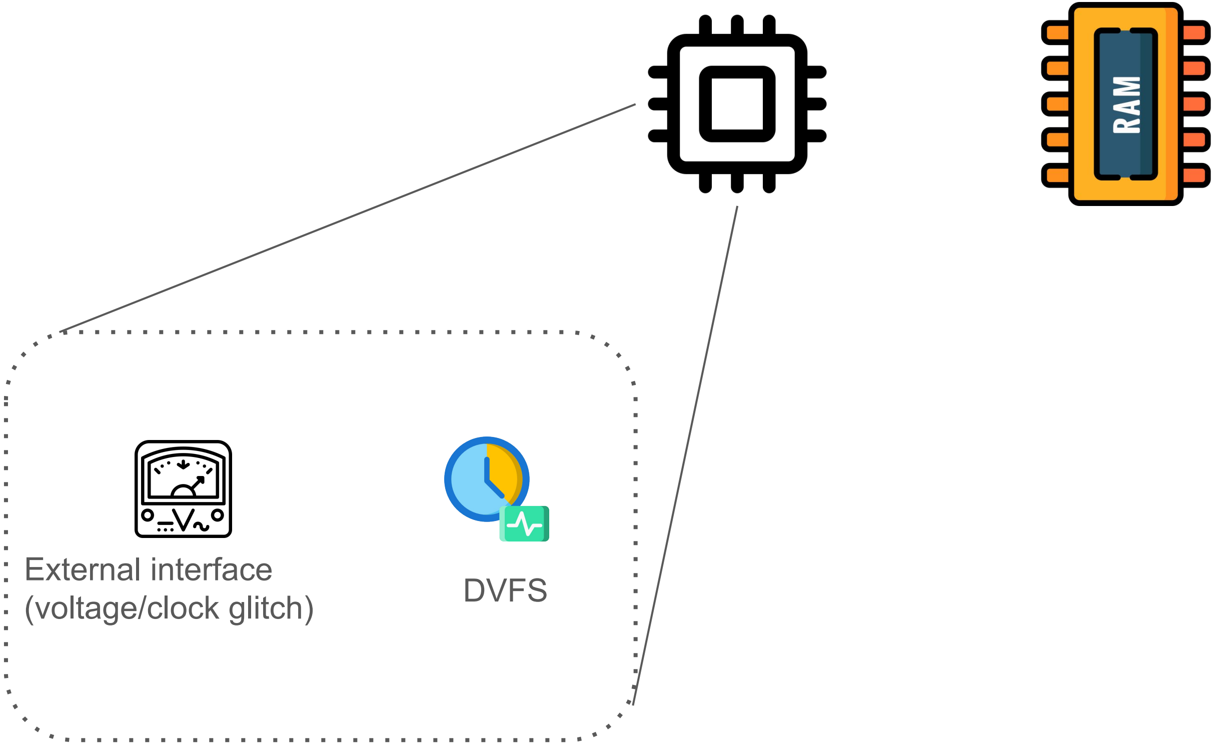
- Traditional Architectural Aspects for FI on SoCs   : Processor and Memory
- An *alternative* Architectural Aspect for FI on SoCs : System Bus
- End-to-End attack on Open Portable Trusted Execution Environment

Traditional Architectural Aspects for FI on SoCs : Fault Points





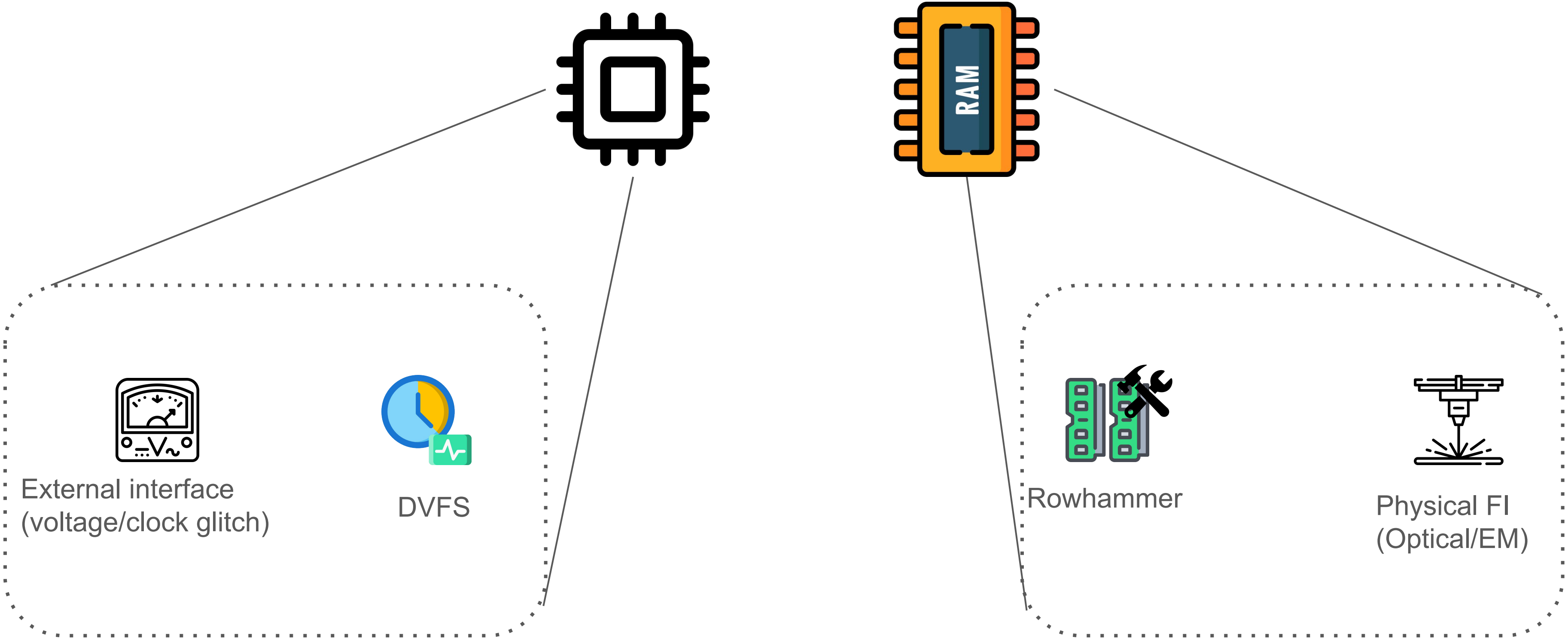
Traditional Architectural Aspects for FI on SoCs : Fault Points



External interface
(voltage/clock glitch)

DVFS

Traditional Architectural Aspects for FI on SoCs : Fault Points



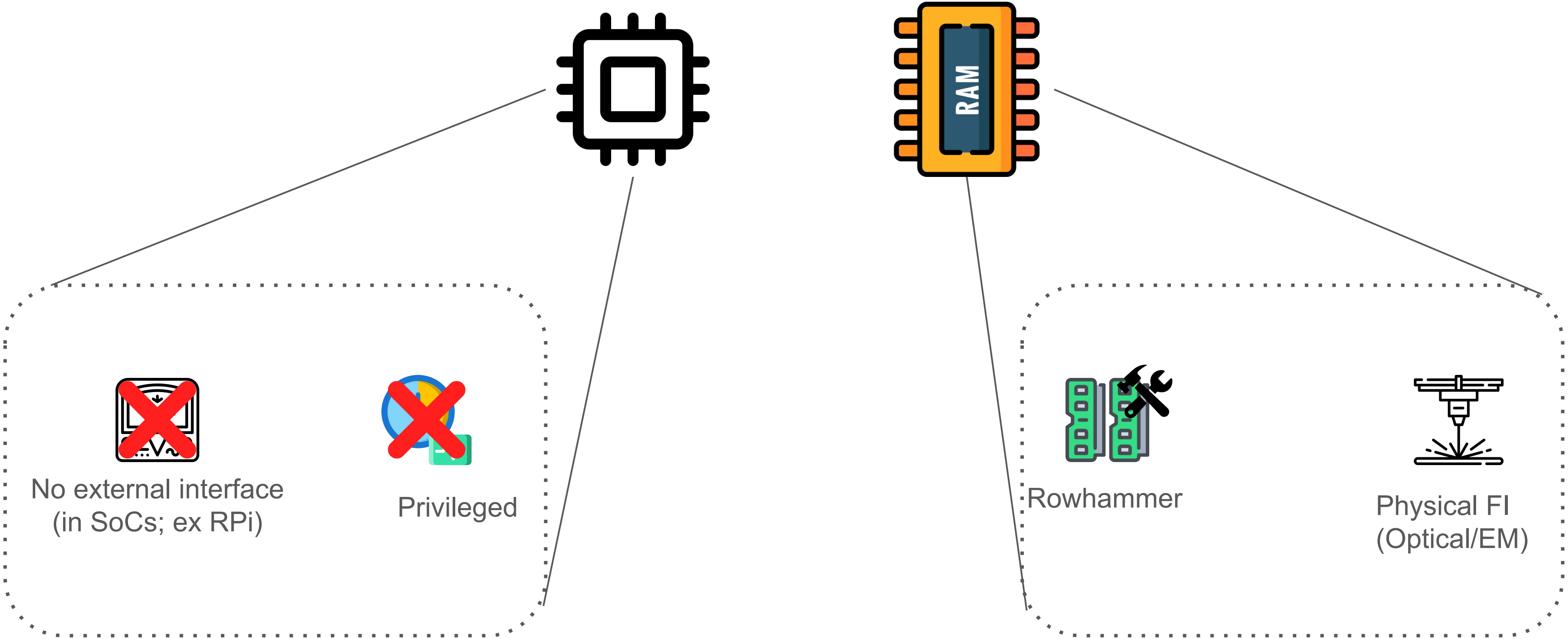
External interface (voltage/clock glitch)

DVFS

Rowhammer

Physical FI (Optical/EM)

Traditional Architectural Aspects for FI on SoCs : Defences



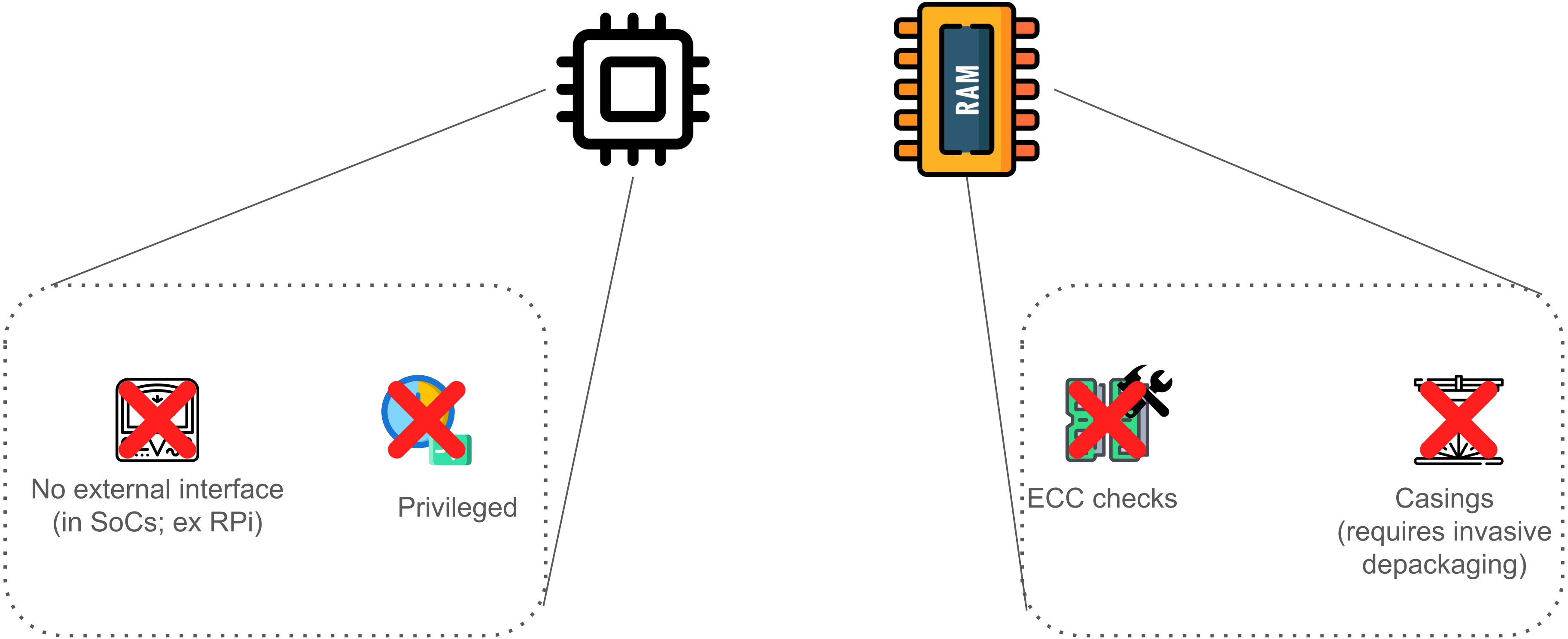
No external interface (in SoCs; ex RPi)

Privileged

Rowhammer

Physical FI (Optical/EM)

Traditional Architectural Aspects for FI on SoCs : Defences



No external interface
(in SoCs; ex RPi)

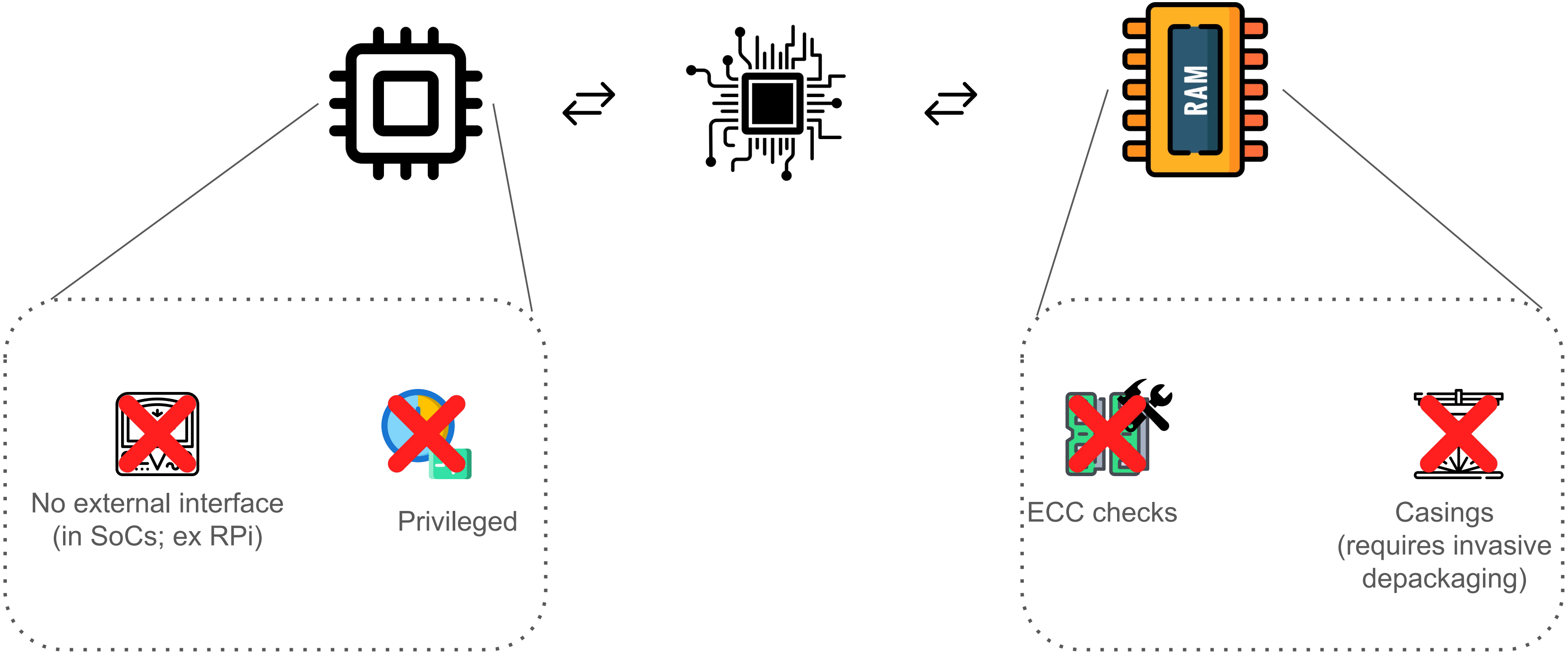
Privileged

ECC checks

Casings
(requires invasive
depackaging)

Are there other **architectural aspects** which can be **used for faults**, for which **no known defences** are deployed yet?

Alternative Architectural Aspect for FI on SoCs : System Bus



No external interface
(in SoCs; ex RPi)

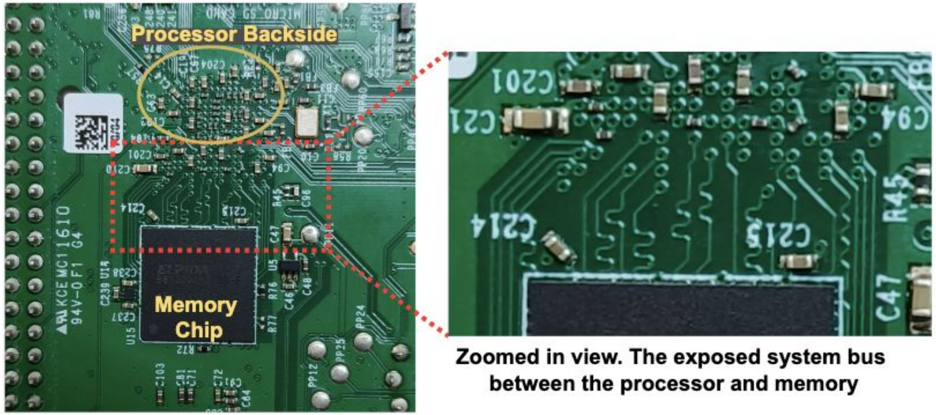
Privileged

ECC checks

Casings
(requires invasive
depackaging)

Alternative Architectural Aspect for FI on SoCs : System Bus

- Uncased and exposed
- Involved mainly with **load/store** instructions
- **Prior works**
  (1) simulation of bus faults
  (2) external voltage glitches on PlayStation consoles to **skip** memory cycles



Zoomed in view. The exposed system bus between the processor and memory

Fig: Exposed bus connections in RPi3

Bus Faults : Attack Principle

load dest_reg, [mem_addr]

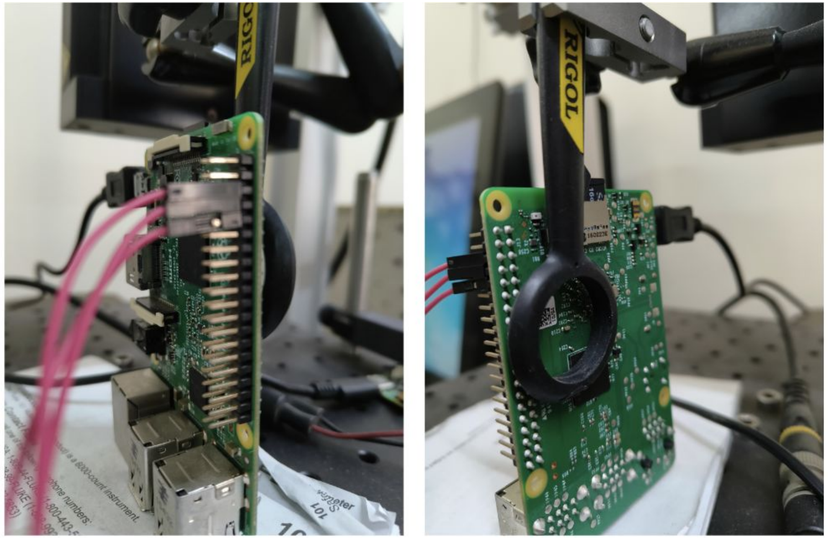
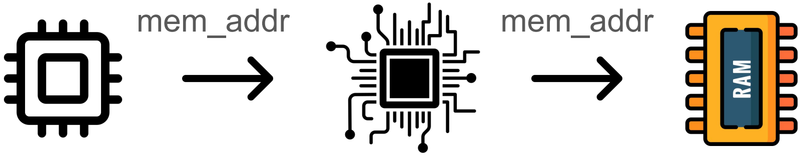


Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3

Bus Faults : Attack Principle

①



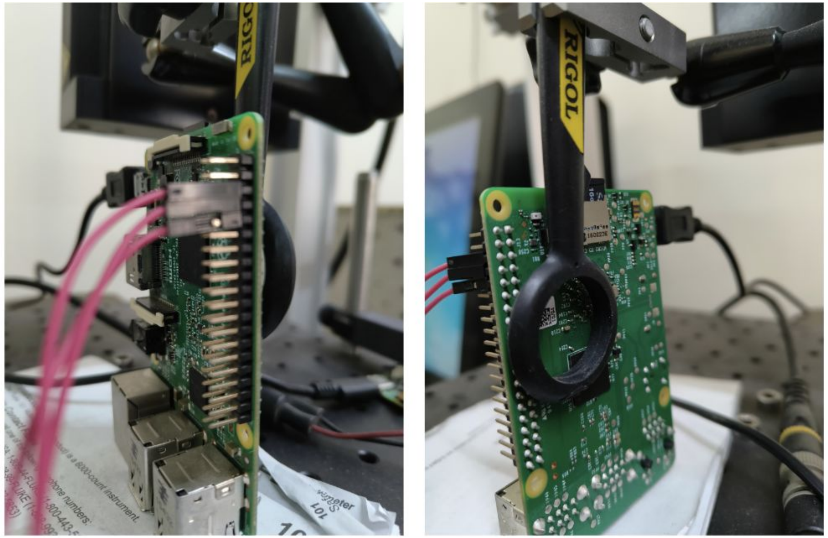
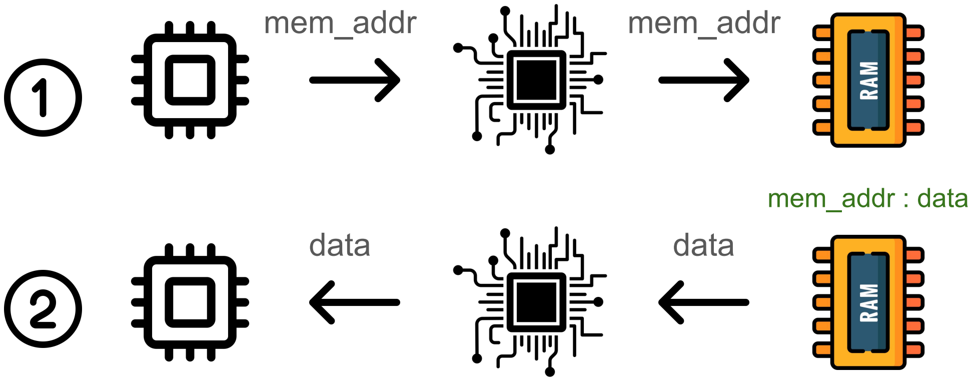


Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3

load dest_reg, [mem_addr]

Bus Faults : Attack Principle



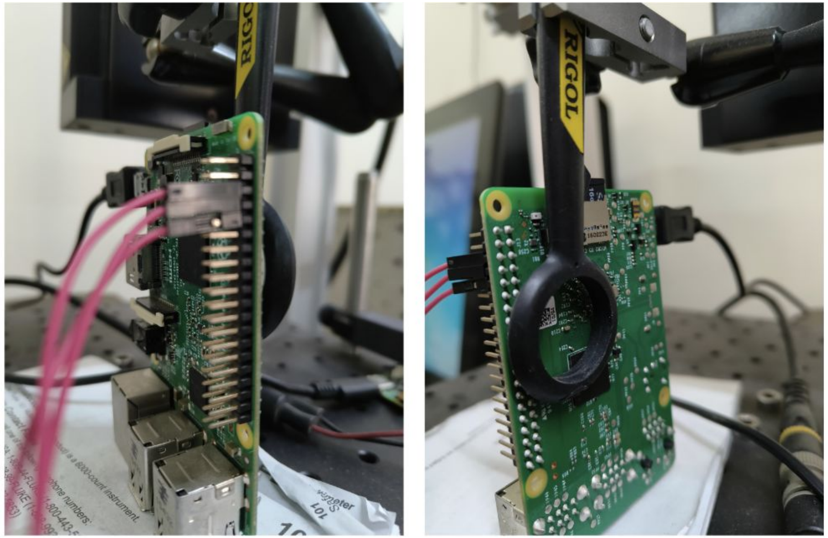
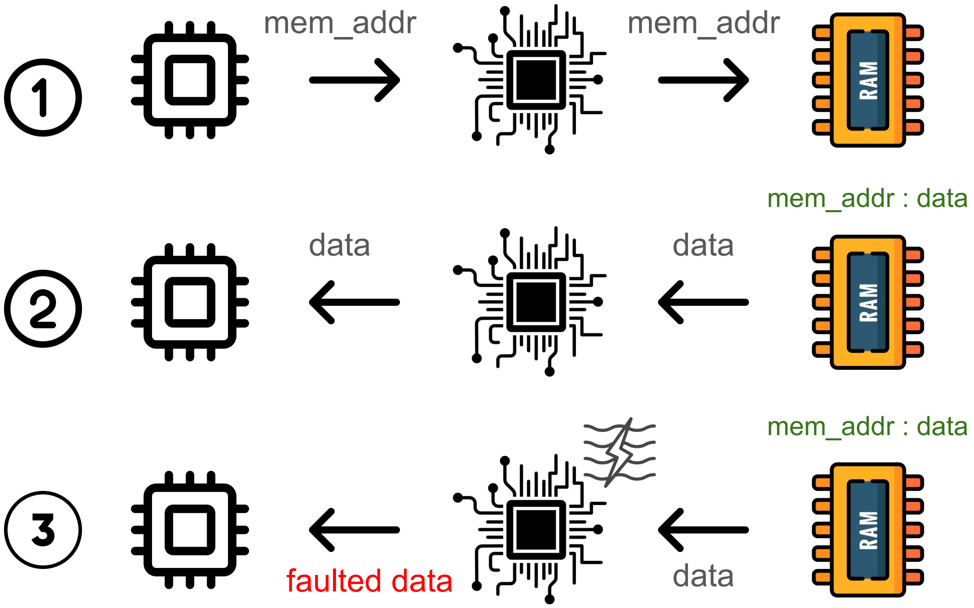


Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3

```
load dest_reg, [mem_addr]
```

Bus Faults : Attack Principle



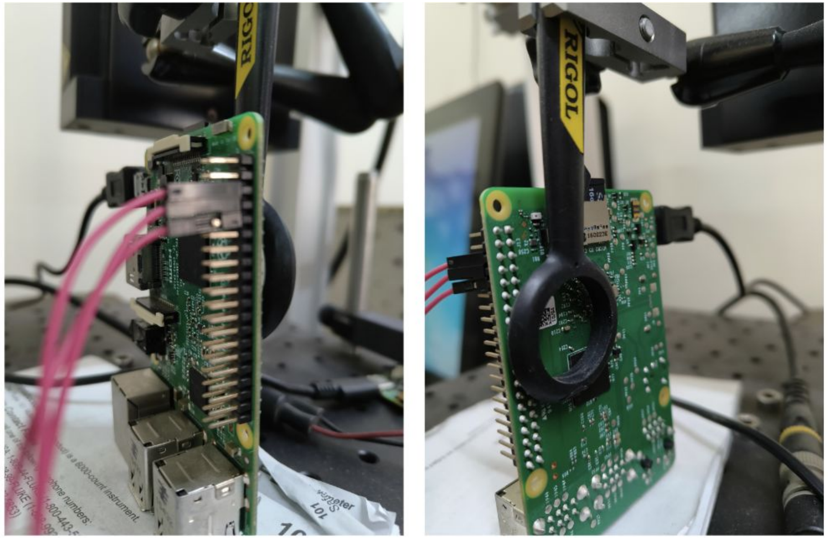
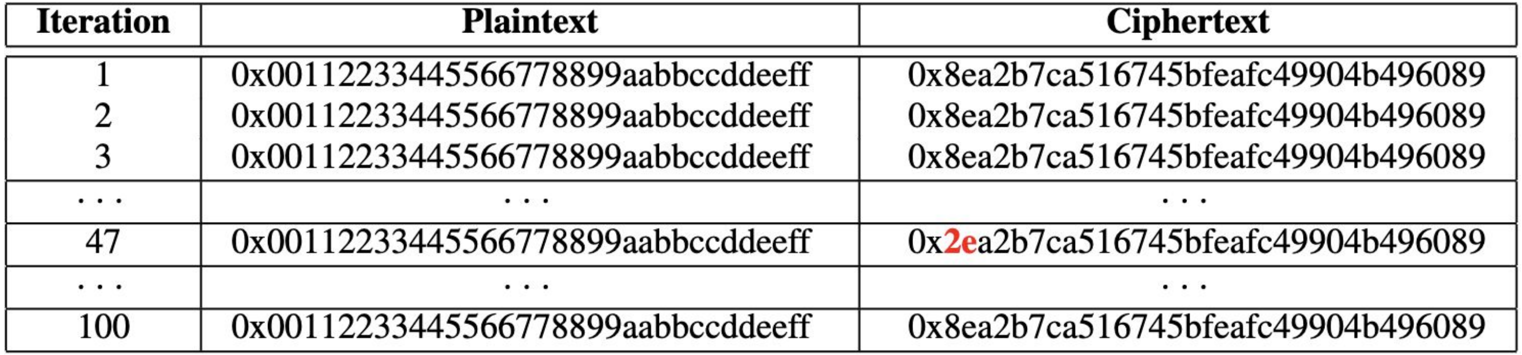


Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3

load dest_reg, [mem_addr]

Bus Faults : Differential Fault Attack on AES



Iteration	Plaintext	Ciphertext
1 2 3	0x00112233445566778899aabbccddeeff 0x00112233445566778899aabbccddeeff 0x00112233445566778899aabbccddeeff	0x8ea2b7ca516745bfeafc49904b496089 0x8ea2b7ca516745bfeafc49904b496089 0x8ea2b7ca516745bfeafc49904b496089
...	...	...
47	0x00112233445566778899aabbccddeeff	0x2ea2b7ca516745bfeafc49904b496089
...	...	...
100	0x00112233445566778899aabbccddeeff	0x8ea2b7ca516745bfeafc49904b496089

- Table based implementation (AESNI absent on SoCs)
- Fault injection in Round 8
- Key entropy reduction to 2^8 [1]

[1] Tunstall, M., Mukhopadhyay, D., & Ali, S. (2011). Differential fault analysis of the advanced encryption standard using a single fault.

Bus Faults : Comparison with FI on Memory

- **Probe position** does not influence memory chip
- **load** instruction fetches **correct data** once probe is removed (**transient** fault)

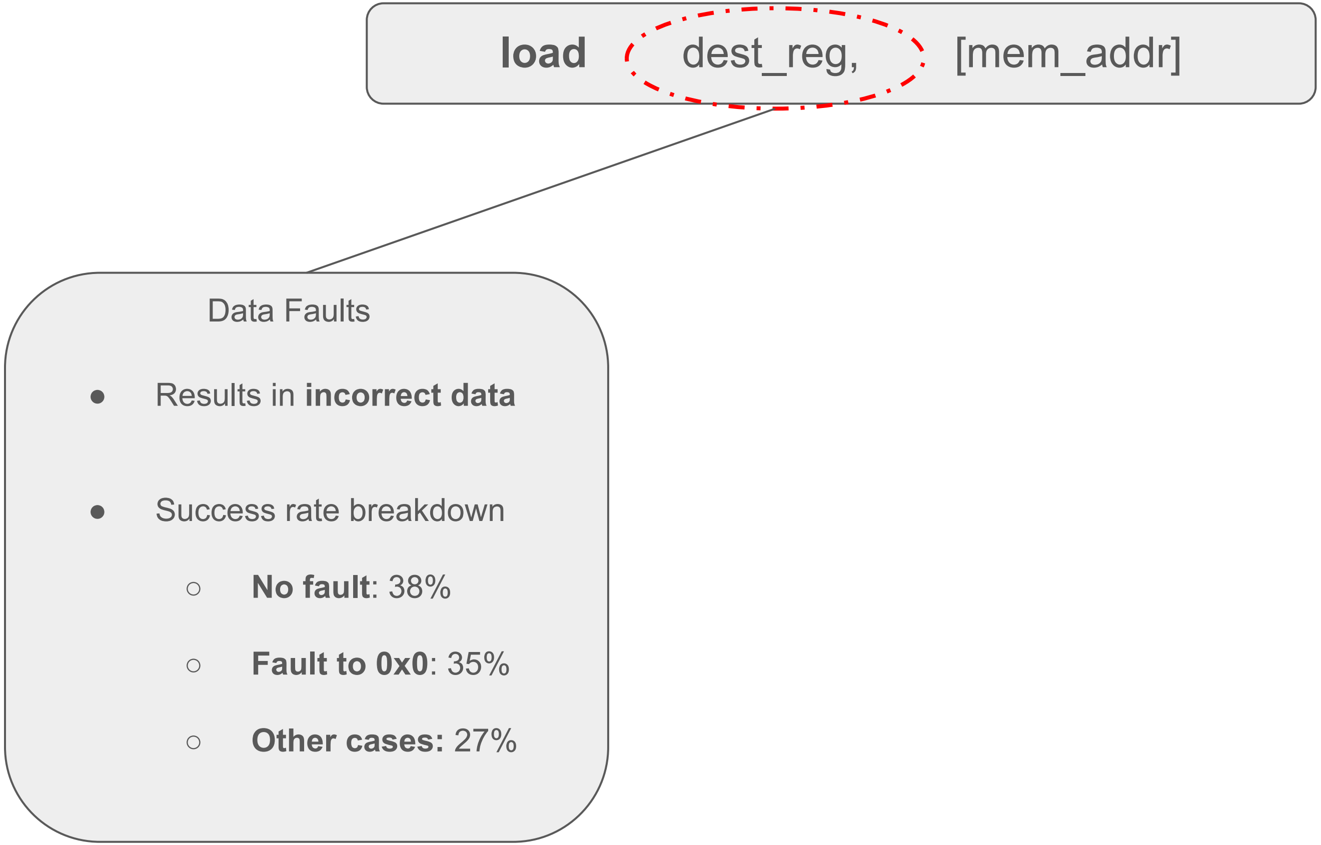
Bus Faults : Comparison with FI on Processor

- **Probe position** does not influence process
- No depackaging performed on target systems
- **[Empirical Observation]** DFA on AES not reproducible with probe position over the packaged processor

Bus Faults : Characterization and Success Rate

```
load      dest_reg,      [mem_addr]
```

Bus Faults : Characterization and Success Rate

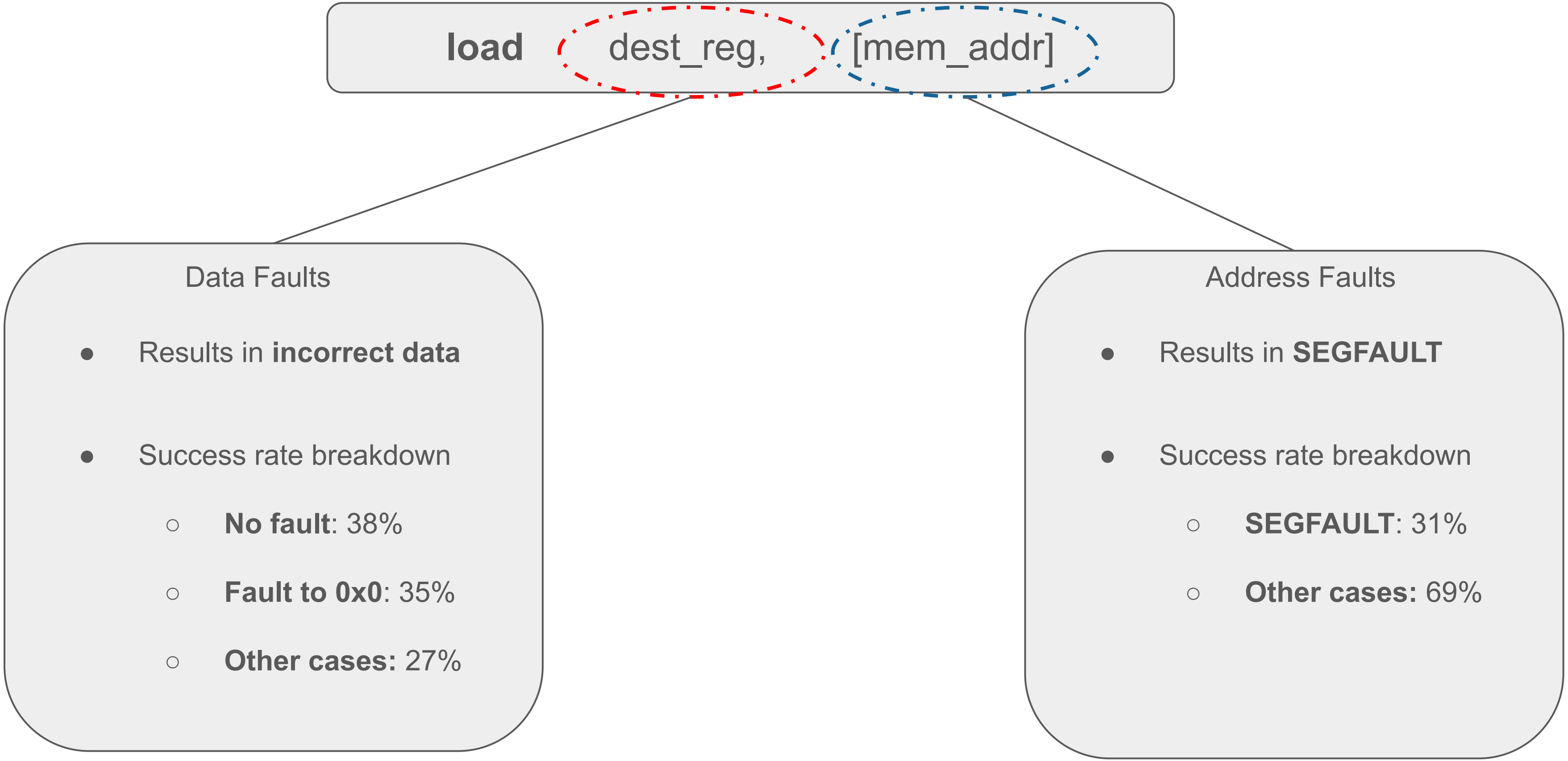


load dest_reg, [mem_addr]

Data Faults

- Results in **incorrect data**
- Success rate breakdown
  - **No fault**: 38%
  - **Fault to 0x0**: 35%
  - **Other cases:** 27%

Bus Faults : Characterization and Success Rate



load dest_reg, [mem_addr]

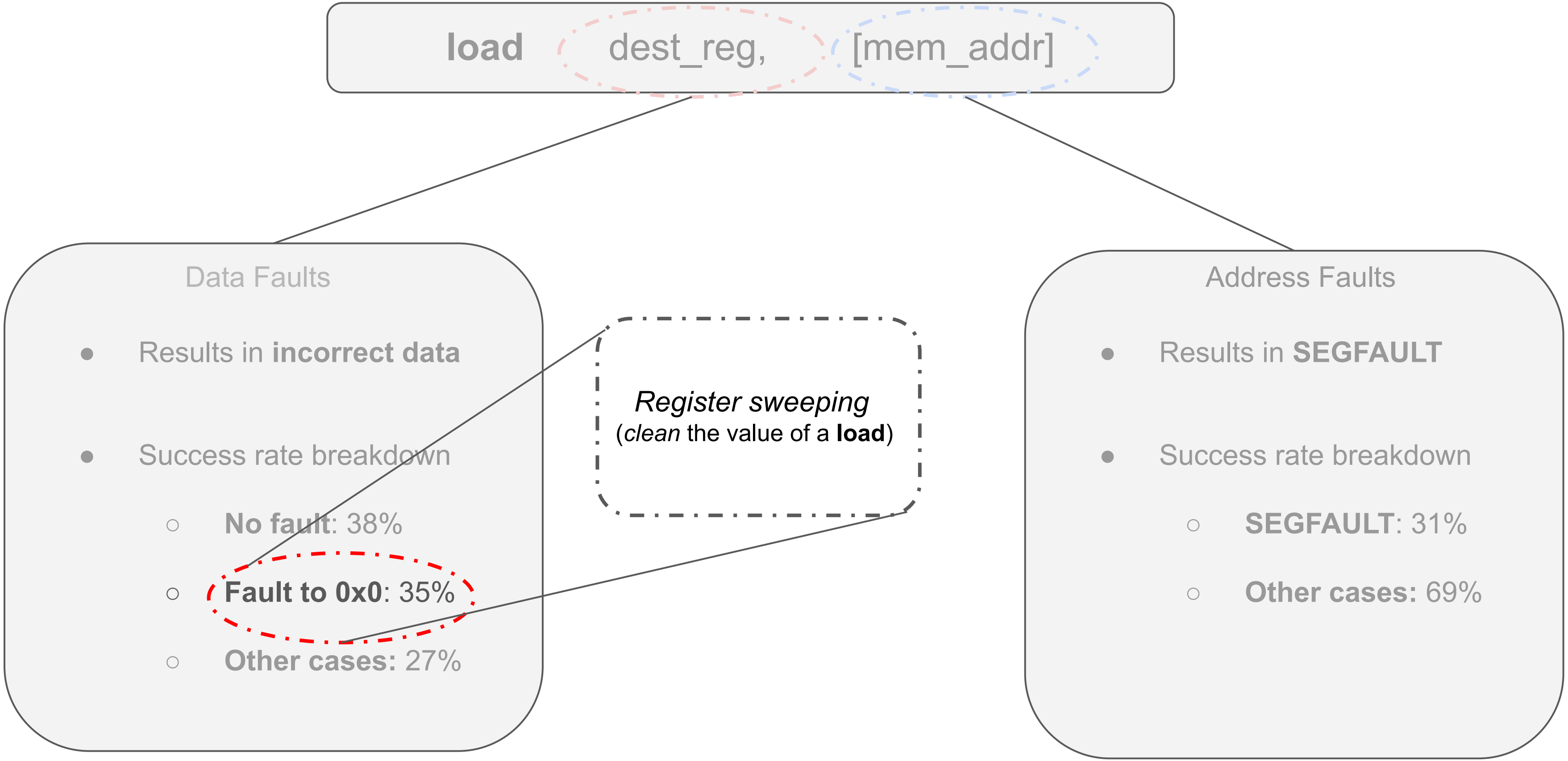
Data Faults

- Results in **incorrect data**
- Success rate breakdown
  - **No fault**: 38%
  - **Fault to 0x0**: 35%
  - **Other cases:** 27%

Address Faults

- Results in **SEGFAULT**
- Success rate breakdown
  - **SEGFAULT**: 31%
  - **Other cases:** 69%

Bus Faults : Characterization and Success Rate



load dest_reg, [mem_addr]

Data Faults

- Results in **incorrect data**
- Success rate breakdown
  - **No fault**: 38%
  - **Fault to 0x0**: 35%
  - **Other cases:** 27%

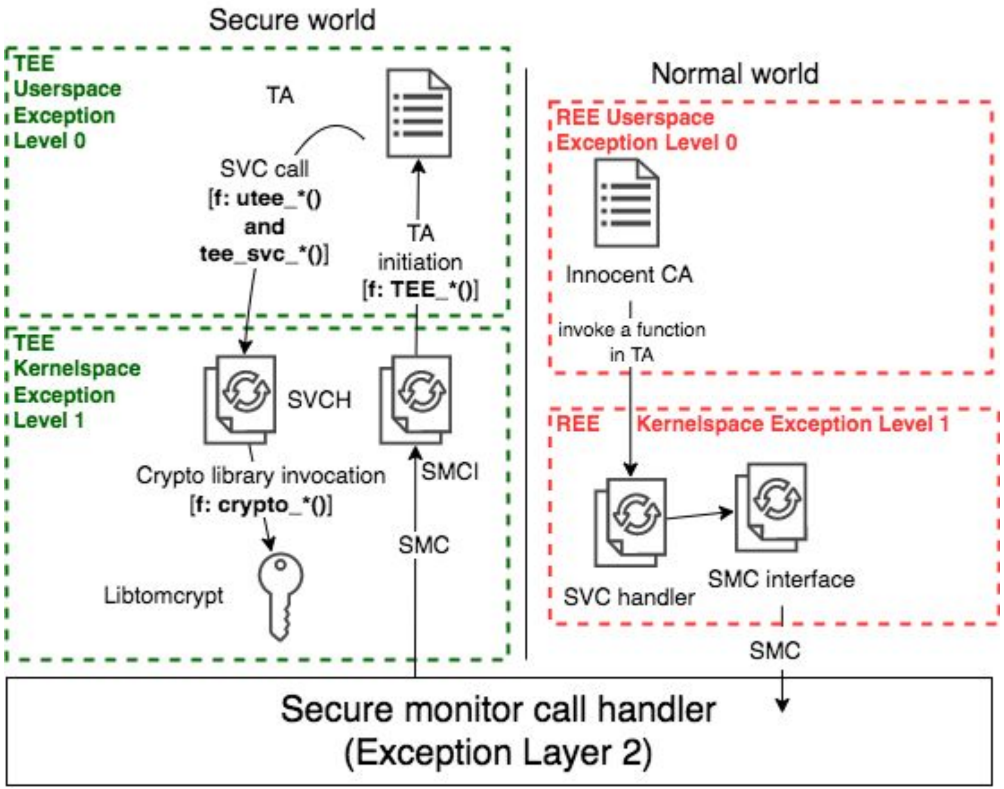
Register sweeping
(*clean* the value of a **load**)

Address Faults

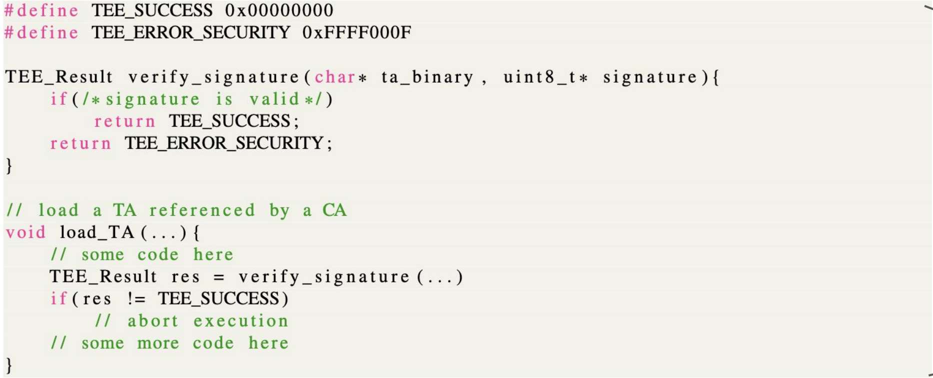
- Results in **SEGFAULT**
- Success rate breakdown
  - **SEGFAULT**: 31%
  - **Other cases:** 69%

Register sweeping to mount an end-to-end attack on Open Portable Trusted Execution Environment (OP-TEE)

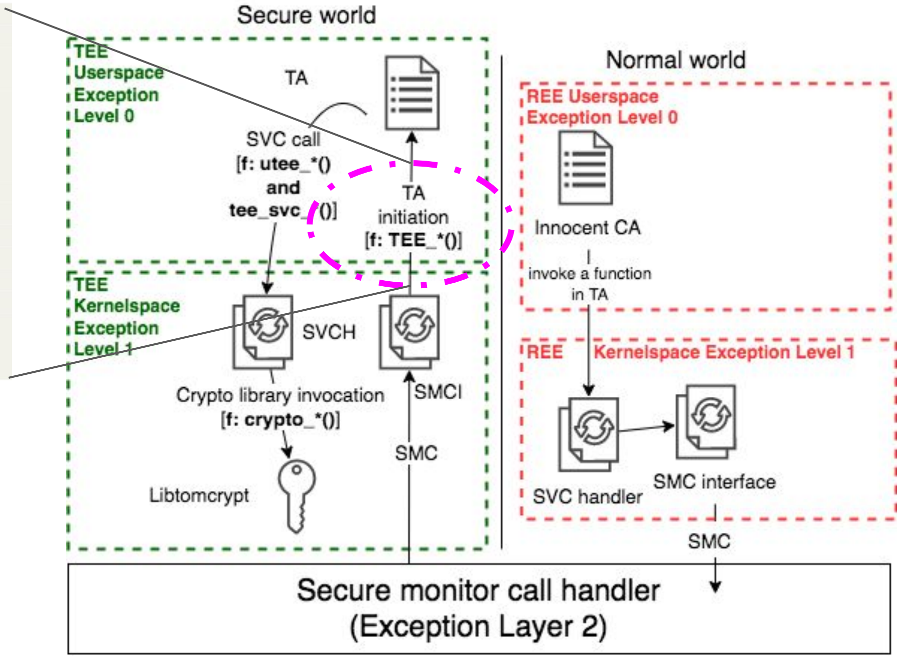
Attack on TEE : Architecture


Secure world
TEE Userspace Exception Level 0
TA
SVC call [f: utee_*() and tee_svc_*()]
TA initiation [f: TEE_*()]
TEE Kernelspace Exception Level 1
SVCH
Crypto library invocation [f: crypto_*()]
SMCI
SMC
Libtomcrypt
Normal world
REE Userspace Exception Level 0
Innocent CA
invoke a function in TA
REE Kernelspace Exception Level 1
SVC handler
SMC interface
SMC
Secure monitor call handler (Exception Layer 2)

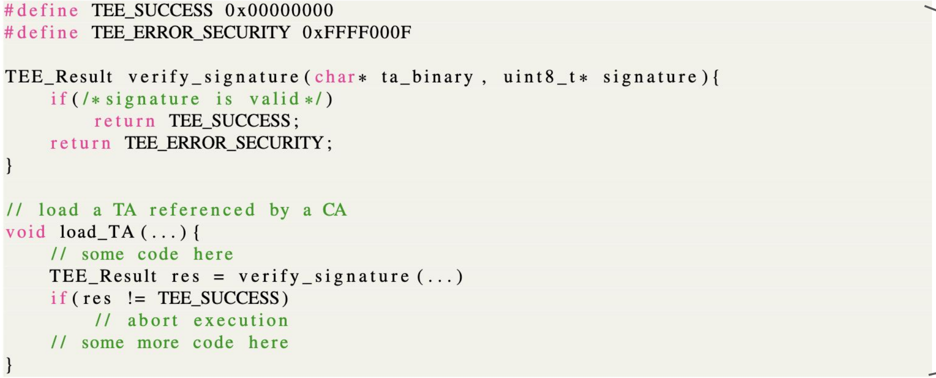
Attack on TEE : Attack Point



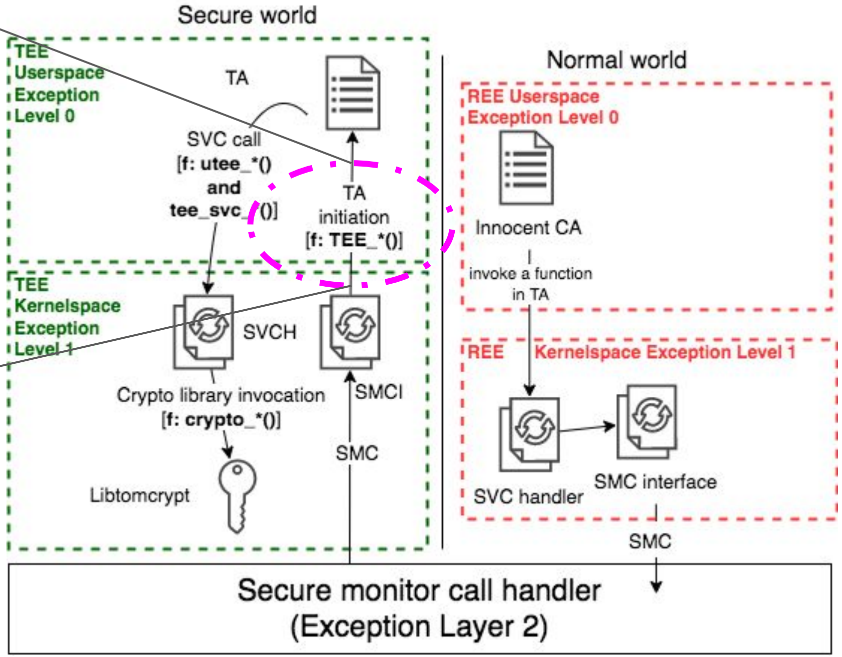
```
#define TEE_SUCCESS 0x00000000
#define TEE_ERROR_SECURITY 0xFFFF000F

TEE_Result verify_signature(char* ta_binary, uint8_t* signature){
    if(/*signature is valid*/)
        return TEE_SUCCESS;
    return TEE_ERROR_SECURITY;
}

// load a TA referenced by a CA
void load_TA(...){
    // some code here
    TEE_Result res = verify_signature(...)
    if(res != TEE_SUCCESS)
        // abort execution
    // some more code here
}
```



Attack on TEE : Attack Point



```
#define TEE_SUCCESS 0x00000000
#define TEE_ERROR_SECURITY 0xFFFF000F

TEE_Result verify_signature(char* ta_binary, uint8_t* signature){
    if(/*signature is valid*/)
        return TEE_SUCCESS;
    return TEE_ERROR_SECURITY;
}

// load a TA referenced by a CA
void load_TA(...){
    // some code here
    TEE_Result res = verify_signature(...)
    if(res != TEE_SUCCESS)
        // abort execution
    // some more code here
}
```


Secure world
TEE Userspace Exception Level 0
TA
SVC call [f: utee_*() and tee_svc_*()]
TA initiation [f: TEE_*()]
TEE Kernelspace Exception Level 1
SVCH
Crypto library invocation [f: crypto_*()]
SMCI
SMC
Libtomcrypt
Normal world
REE Userspace Exception Level 0
Innocent CA
invoke a function in TA
REE Kernelspace Exception Level 1
SVC handler
SMC interface
SMC
Secure monitor call handler (Exception Layer 2)

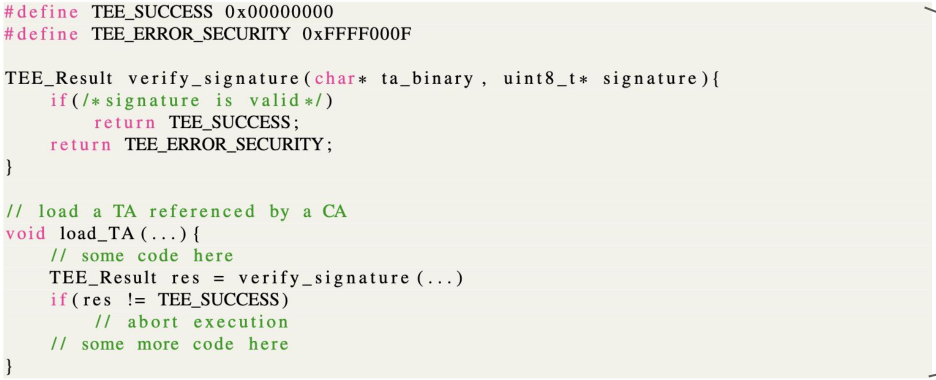
External glitch

DVFS

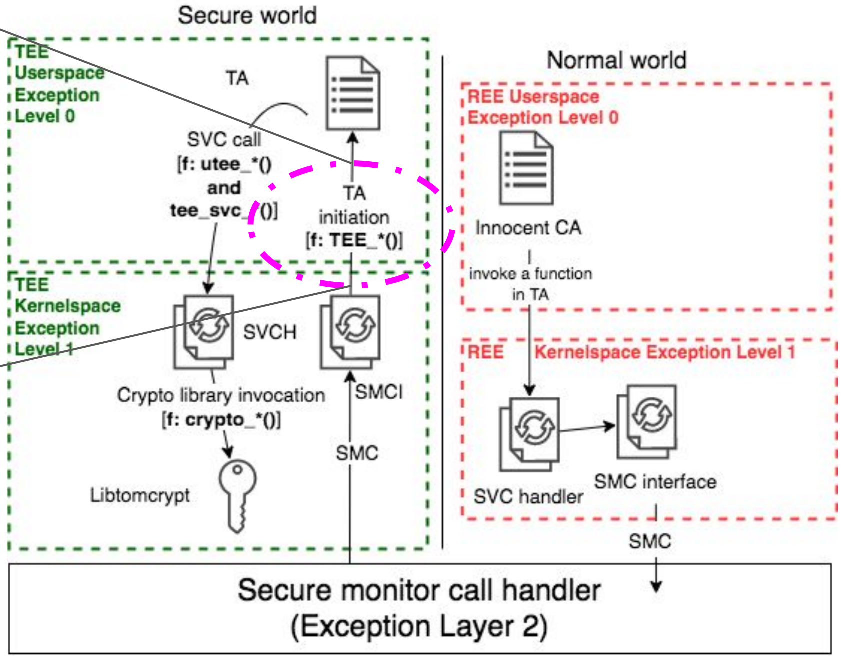
Rowhammer

Steal signing key

Attack on TEE : Attack Point



```
#define TEE_SUCCESS 0x00000000
#define TEE_ERROR_SECURITY 0xFFFF000F

TEE_Result verify_signature(char* ta_binary, uint8_t* signature){
    if(/*signature is valid*/)
        return TEE_SUCCESS;
    return TEE_ERROR_SECURITY;
}

// load a TA referenced by a CA
void load_TA(...){
    // some code here
    TEE_Result res = verify_signature(...)
    if(res != TEE_SUCCESS)
        // abort execution
    // some more code here
}
```



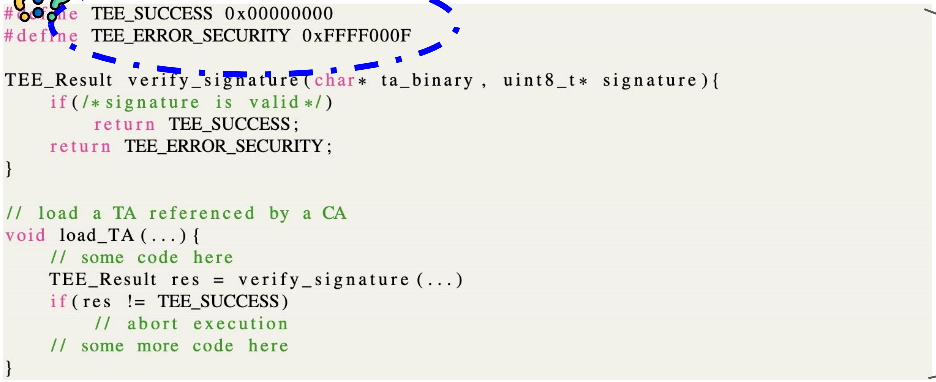
Not Available

Not Available

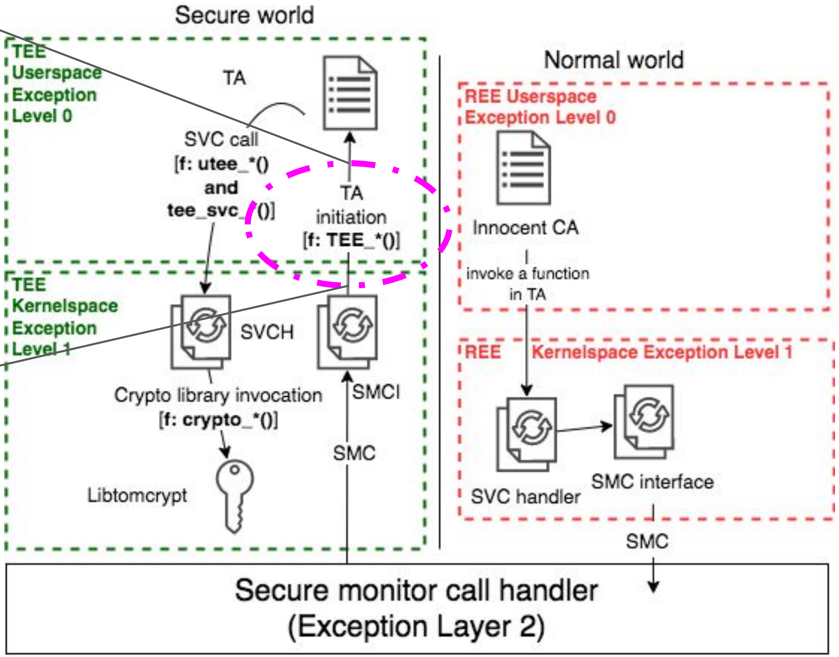
Protected TA memory

Signing key not stored on device

Attack on TEE : Attack Point



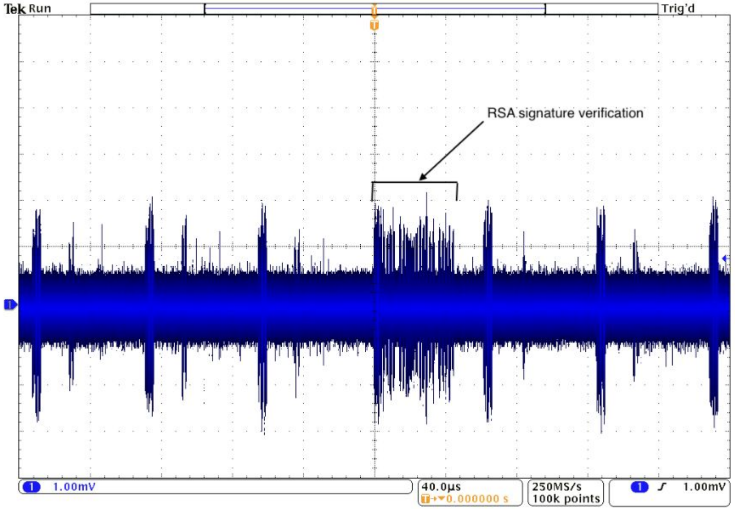
```
#define TEE_SUCCESS 0x00000000
#define TEE_ERROR_SECURITY 0xFFFF000F

TEE_Result verify_signature(char* ta_binary , uint8_t* signature){
    if(/*signature is valid*/)
        return TEE_SUCCESS;
    return TEE_ERROR_SECURITY;
}

// load a TA referenced by a CA
void load_TA(...){
    // some code here
    TEE_Result res = verify_signature(...)
    if(res != TEE_SUCCESS)
        // abort execution
    // some more code here
}
```



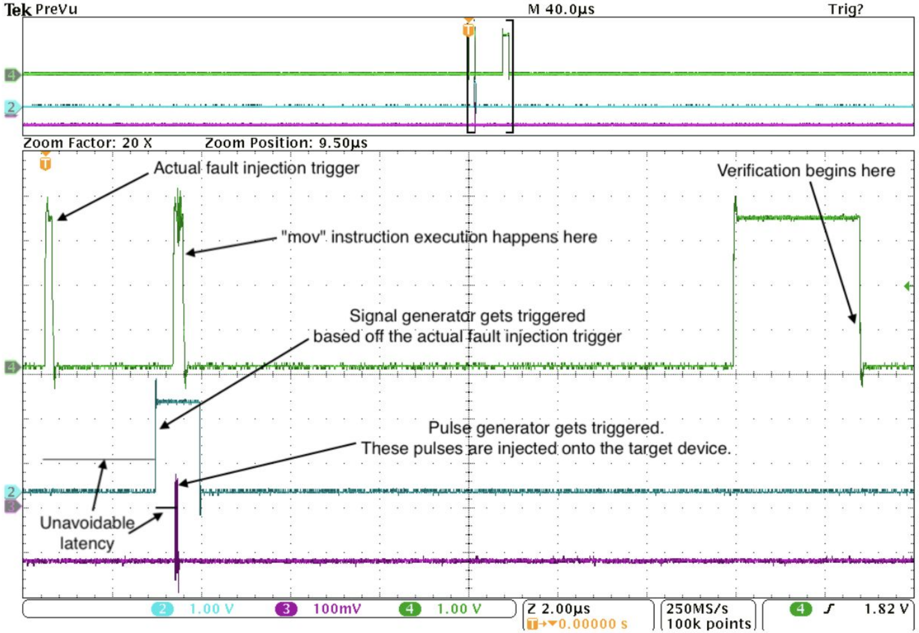
Secure world
Normal world
TEE Userspace Exception Level 0
TA
SVC call [f: utee_*() and tee_svc_*()]
TA initiation [f: TEE_*()]
TEE Kernelspace Exception Level 1
SVCH
SMCI
Crypto library invocation [f: crypto_*()]
SMC
Libtomcrypt
REE Userspace Exception Level 0
Innocent CA
invoke a function in TA
REE Kernelspace Exception Level 1
SVC handler
SMC interface
SMC
Secure monitor call handler (Exception Layer 2)

Register sweeping : Fault a load to **0x0** through bus faults

Attack on TEE : Combined Adversary



+



Power-side channel to inform fault injection in a **non-invasive** way
(no recompilation of OP-TEE kernel necessary)

Actual fault injection on signature verification

Attack on TEE : Fallout

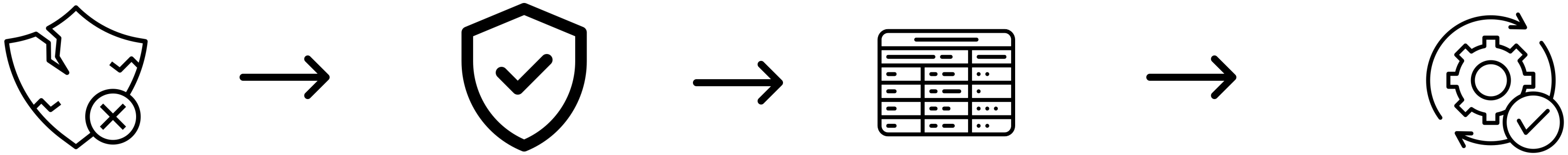
Register sweeping fault attack loads a **self-signed, adversarial controlled** Trusted Application in the secure world of OP-TEE

Attack on TEE : Increasing Capability of self-signed TA

- **Redirect** (encrypted) communication meant for other benign TAs
- **Decrypt** the (encrypted) redirected communication

Attack on TEE : Increasing Capability of self-signed TA

Redirect (encrypted) communication meant for other benign TAs



Insecure World

Secure World

Universally **U**nique **ID**entifier (UUID) comparison

Secure Trusted Application execution

Attack on TEE : Increasing Capability of self-signed TA

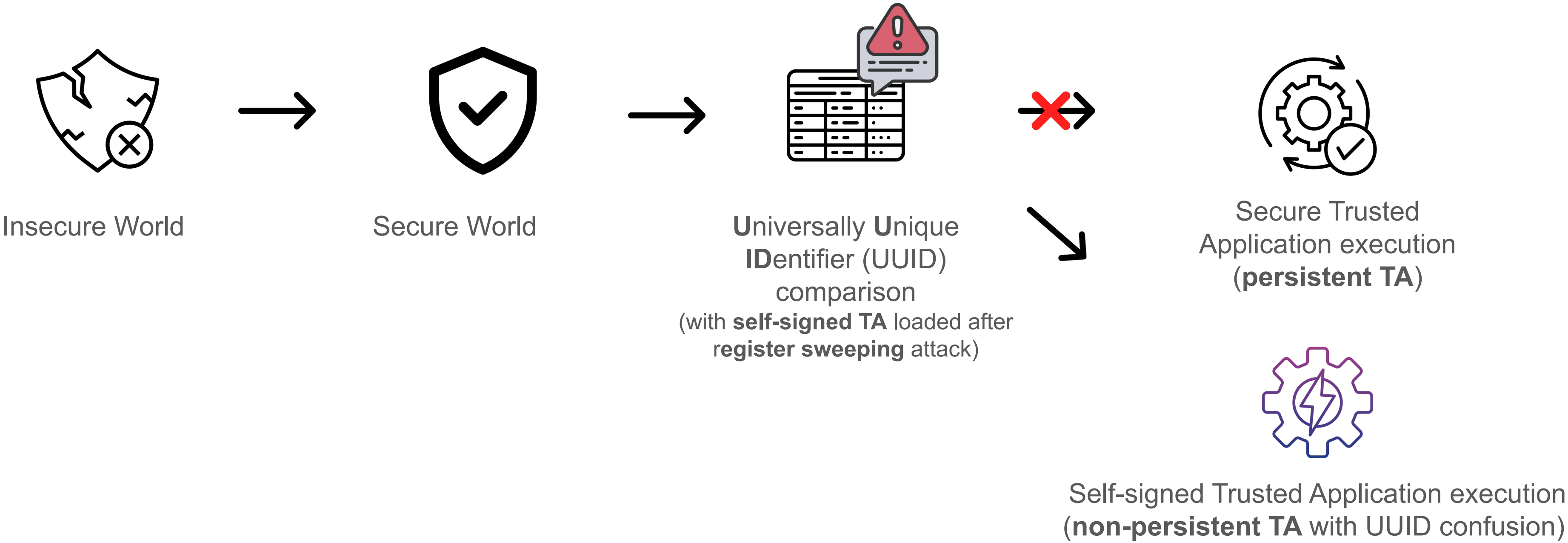
Redirect (encrypted) communication meant for other benign TAs

Our Finding: GlobalPlatform API specification (upon which OP-TEE is constructed) **offloads** the responsibility of choosing UUID to **Original Equipment Manufacturer**. It is the responsibility of the OEM to ensure **no two Trusted Applications (TA) share same UUID.**

UUID confusion: Behaviour of the system when **UUID are non-unique is undefined**. Our empirical conclusion is that, when UUIDs are shared, a **non-persistent TA is preferred over persistent TA.**

Attack on TEE : Increasing Capability of self-signed TA

Redirect (encrypted) communication meant for other benign TAs



Insecure World

Secure World

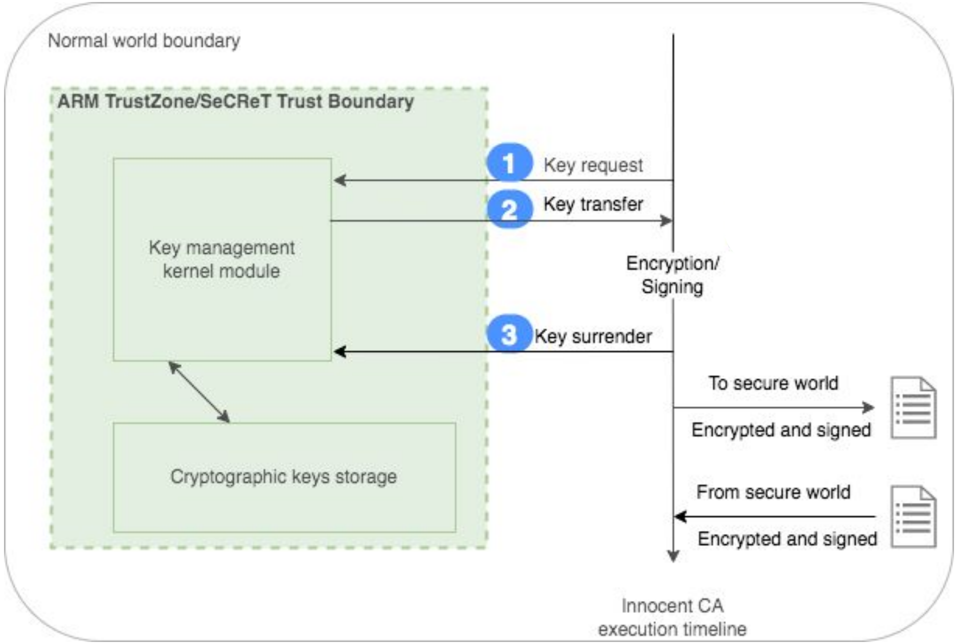
Universally **U**nique **ID**entifier (UUID) comparison
(with **self-signed TA** loaded after **register sweeping** attack)

Secure Trusted Application execution (**persistent TA**)

Self-signed Trusted Application execution (**non-persistent TA** with UUID confusion)

Attack on TEE : Increasing Capability of self-signed TA

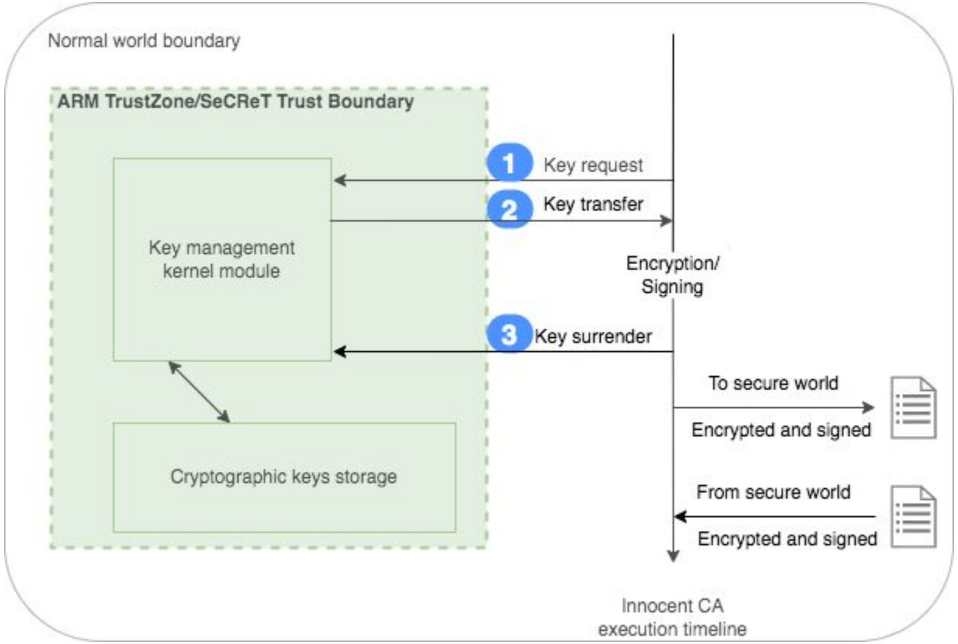
Decrypt the (encrypted) redirected communication



Third Party Extension: SeCReT

Attack on TEE : Increasing Capability of self-signed TA

Decrypt the (encrypted) redirected communication

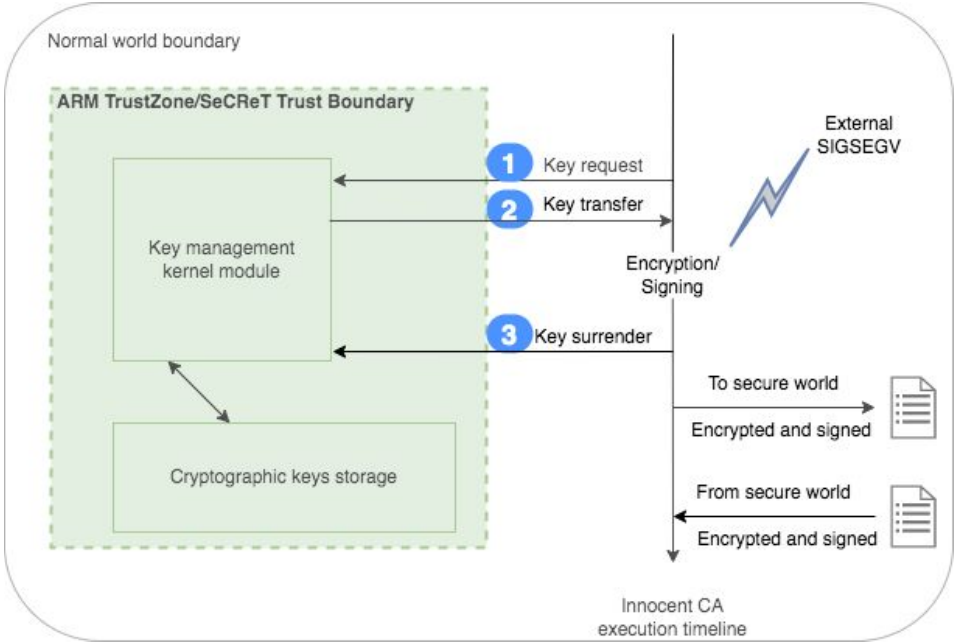


Third Party Extension: SeCReT

- Symmetric key management
- Blocks SIGTRAP
- Blocks unauthorized read to sensitive data pages

Attack on TEE : Increasing Capability of self-signed TA

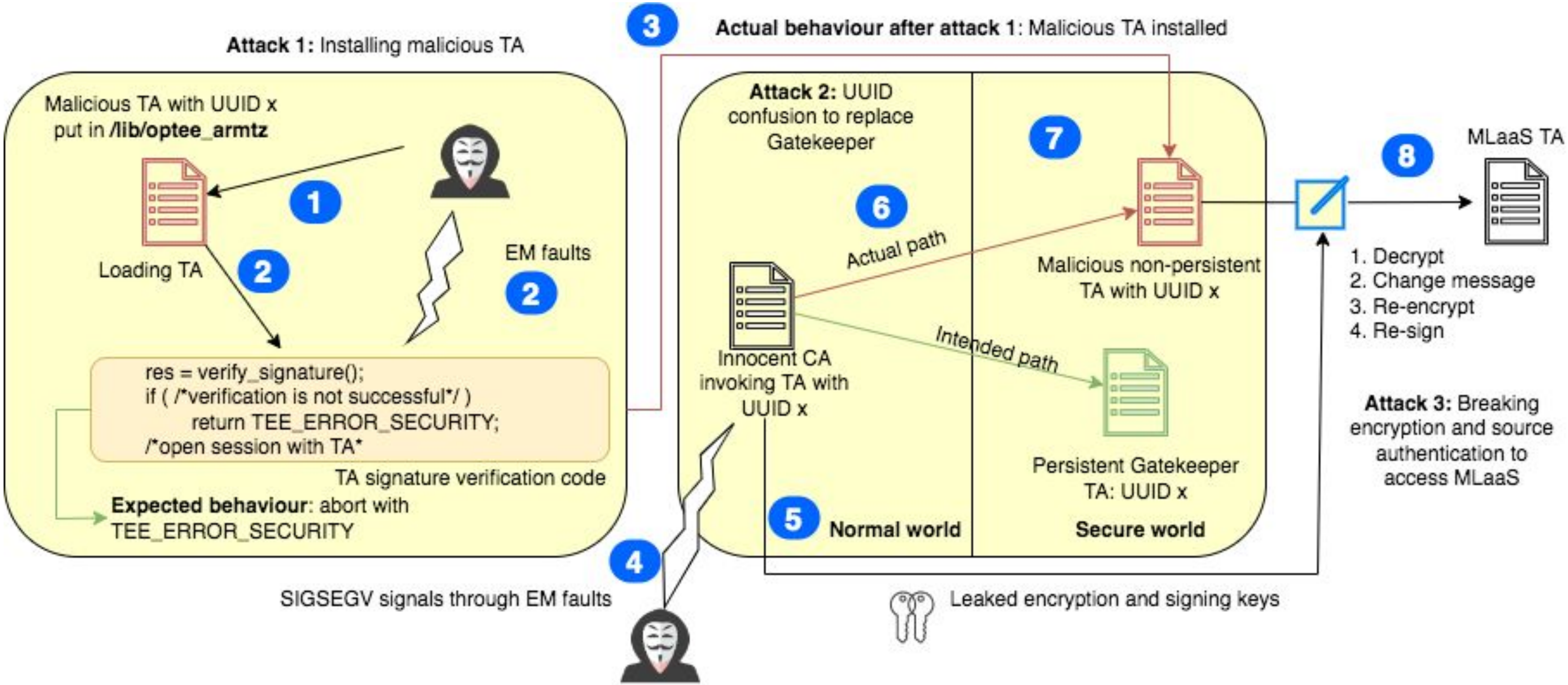
Decrypt the (encrypted) redirected communication



Third Party Extension: SeCReT

- Symmetric key management
- Blocks SIGTRAP
- Blocks unauthorized read to sensitive data pages
- Does not block SIGSEGV. Leaks key through coredump

Attack on TEE : End-to-End Attack on an example TA (MLaaS)



Attack 1: Installing malicious TA
Malicious TA with UUID x put in **/lib/optee_armtz**
1
Loading TA
2
EM faults
2
res = verify_signature();
if (/*verification is not successful*/)
return TEE_ERROR_SECURITY;
/*open session with TA*
TA signature verification code
Expected behaviour: abort with TEE_ERROR_SECURITY
3
Actual behaviour after attack 1: Malicious TA installed
Attack 2: UUID confusion to replace Gatekeeper
6
Actual path
Intended path
Innocent CA invoking TA with UUID x
7
Malicious non-persistent TA with UUID x
Persistent Gatekeeper TA: UUID x
5
Normal world
Secure world
4
SIGSEGV signals through EM faults
Leaked encryption and signing keys
8
MLaaS TA
1. Decrypt
2. Change message
3. Re-encrypt
4. Re-sign
Attack 3: Breaking encryption and source authentication to access MLaaS

Attack on TEE : Impact

- CVE 2022-47549
- Worked together with Linaro to deploy countermeasure in OP-TEE kernel

Research @ Secured Embedded Architecture Laboratory, IIT Kgp

(Some) Research Directions

- Power/EM **Side-channel evaluation** of FPGAs/micro-controllers/SoCs
- **Fault** Attacks, Fault Analysis, and design of countermeasures
- Evaluation of **Micro-architectural attack** scenarios on workstations as well as embedded systems
- Others directions…

Research @ Secured Embedded Architecture Laboratory, IIT Kgp

(Some) Research Directions

- Power/EM **Side-channel evaluation** of FPGAs/micro-controllers/SoCs
- **Fault** Attacks, Fault Analysis, and design of countermeasures
- Evaluation of **Micro-architectural attack** scenarios on workstations as well as embedded systems
- Others directions…

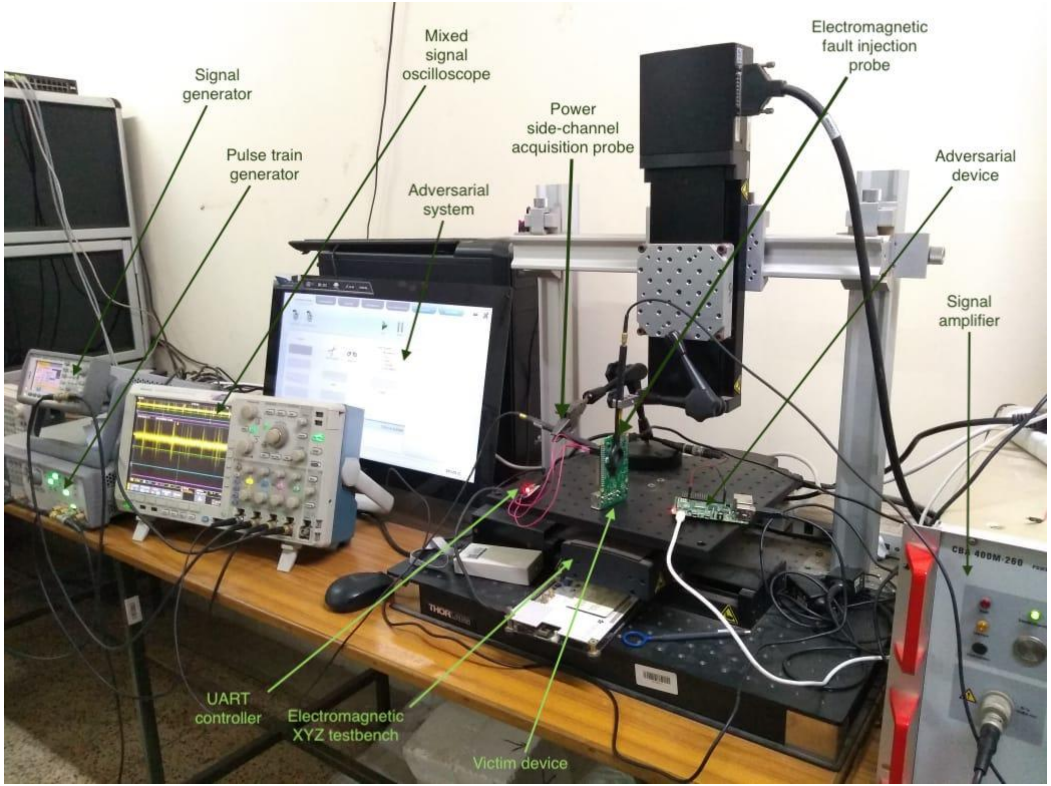

Signal generator
Mixed signal oscilloscope
Electromagnetic fault injection probe
Pulse train generator
Power side-channel acquisition probe
Adversarial system
Adversarial device
Signal amplifier
UART controller
Electromagnetic XYZ testbench
Victim device

Fig: Fault Attack testbed used for this work

Thank You!