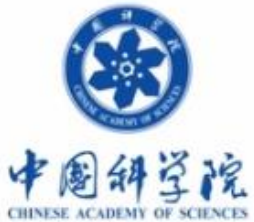
Cache-in-the-Middle (CITM) Attacks:

Manipulating Sensitive Data in Isolated Execution Environments

Jie Wang[1,2,3], Kun Sun[2], Lingguang Lei[1,3], Shengye Wan[2,4], Yuewu Wang[1,3], and Jiwu Jing[5]
[1]*SKLOIS, Institute of Information Engineering, CAS, China*
[2]*Department of Information Sciences and Technology, CSIS, George Mason University*
[3]*School of Cyber Security, University of Chinese Academy of Sciences, China*
[4]*Department of Computer Science, College of William and Mary*
[5]*School of Computer Science and Technology, University of Chinese Academy of Sciences*

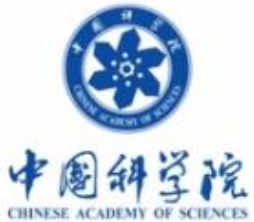


ACM CCS, November 2020

Cache-in-the-Middle (CITM) Attacks:

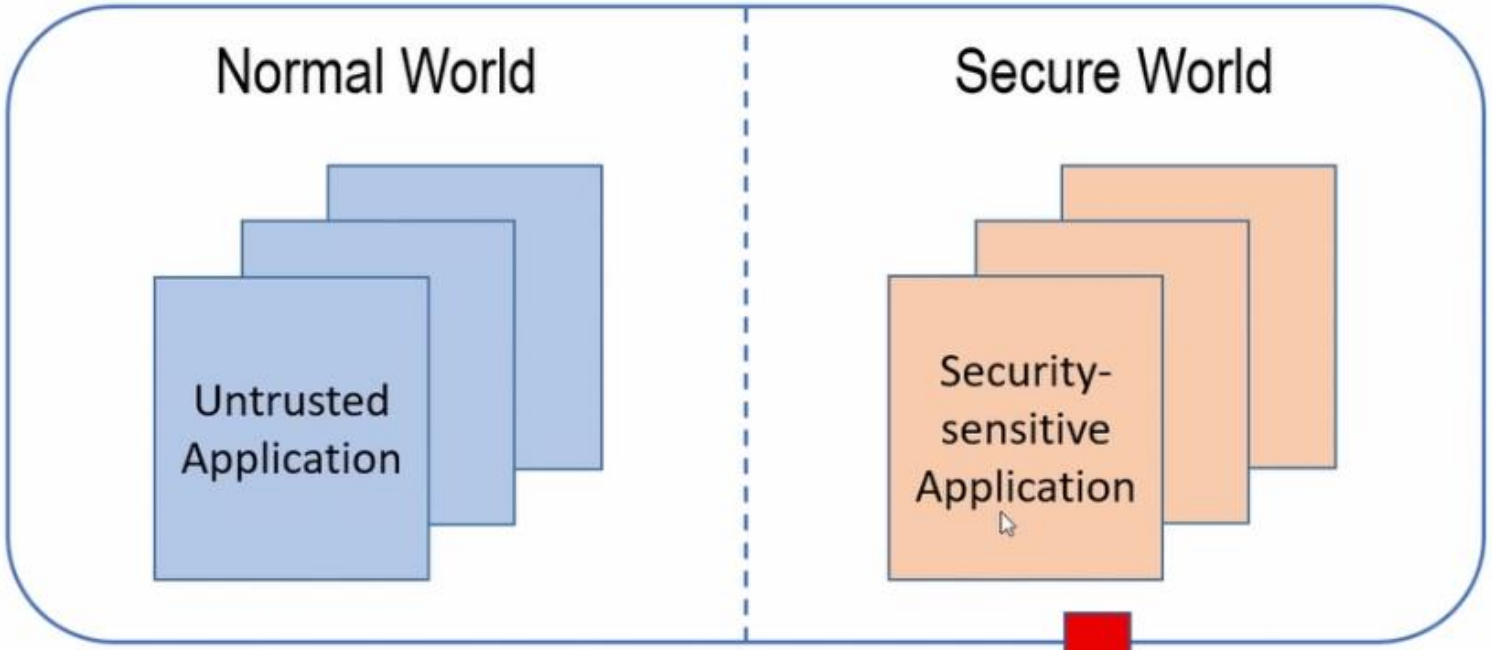
Manipulating Sensitive Data in Isolated Execution Environments

Jie Wang[1,2,3], Kun Sun[2], Lingguang Lei[1,3], Shengye Wan[2,4], Yuewu Wang[1,3], and Jiwu Jing[5]
[1]*SKLOIS, Institute of Information Engineering, CAS, China*
[2]*Department of Information Sciences and Technology, CSIS, George Mason University*
[3]*School of Cyber Security, University of Chinese Academy of Sciences, China*
[4]*Department of Computer Science, College of William and Mary*
[5]*School of Computer Science and Technology, University of Chinese Academy of Sciences*



ACM CCS, November 2020

ARM-based Trusted Execution Environment (TEE)



Restrictions on application installation

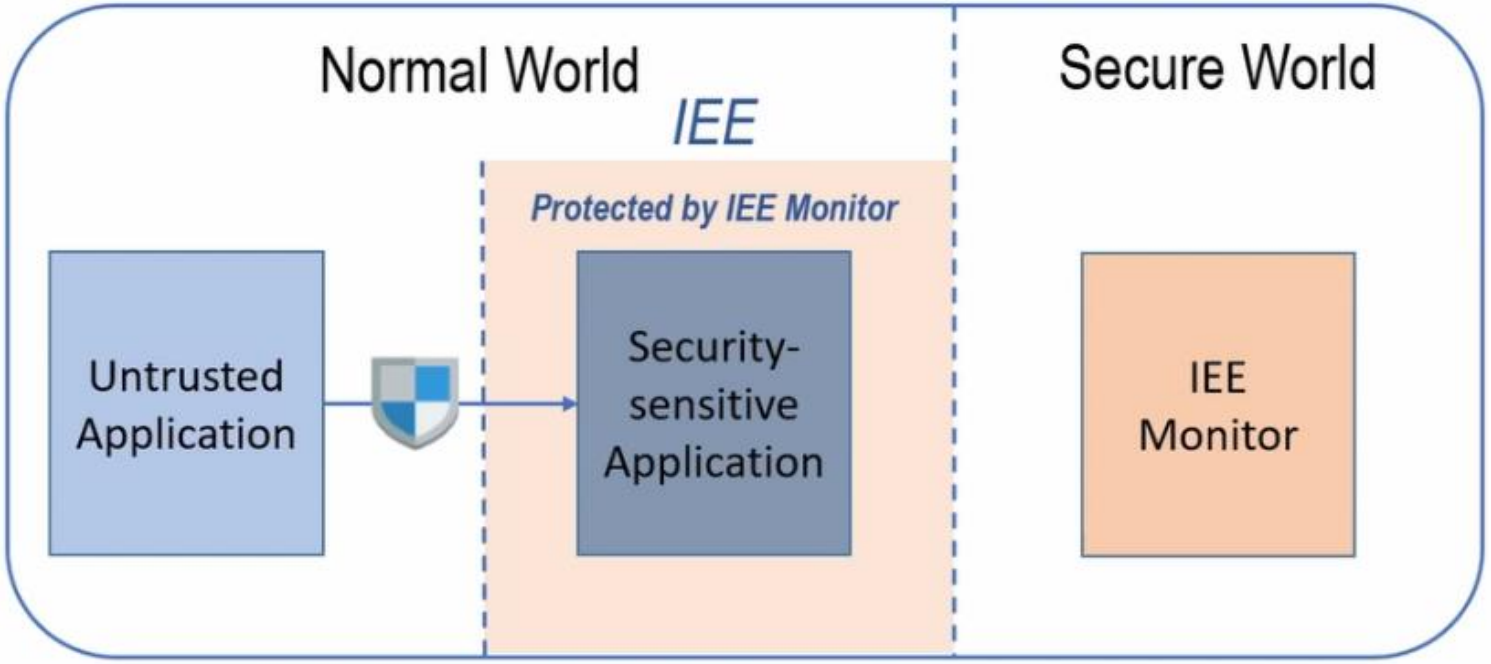
- Increased trusted computing base (TCB) in Secure World.
- Manufacturers prefer to only install their own applications with strict assessment.

Isolated Execution Environment (IEE)

TrustICE (DSN 2015), SANCTUARY (NDSS 2019), Ginseng (NDSS 2019), …….

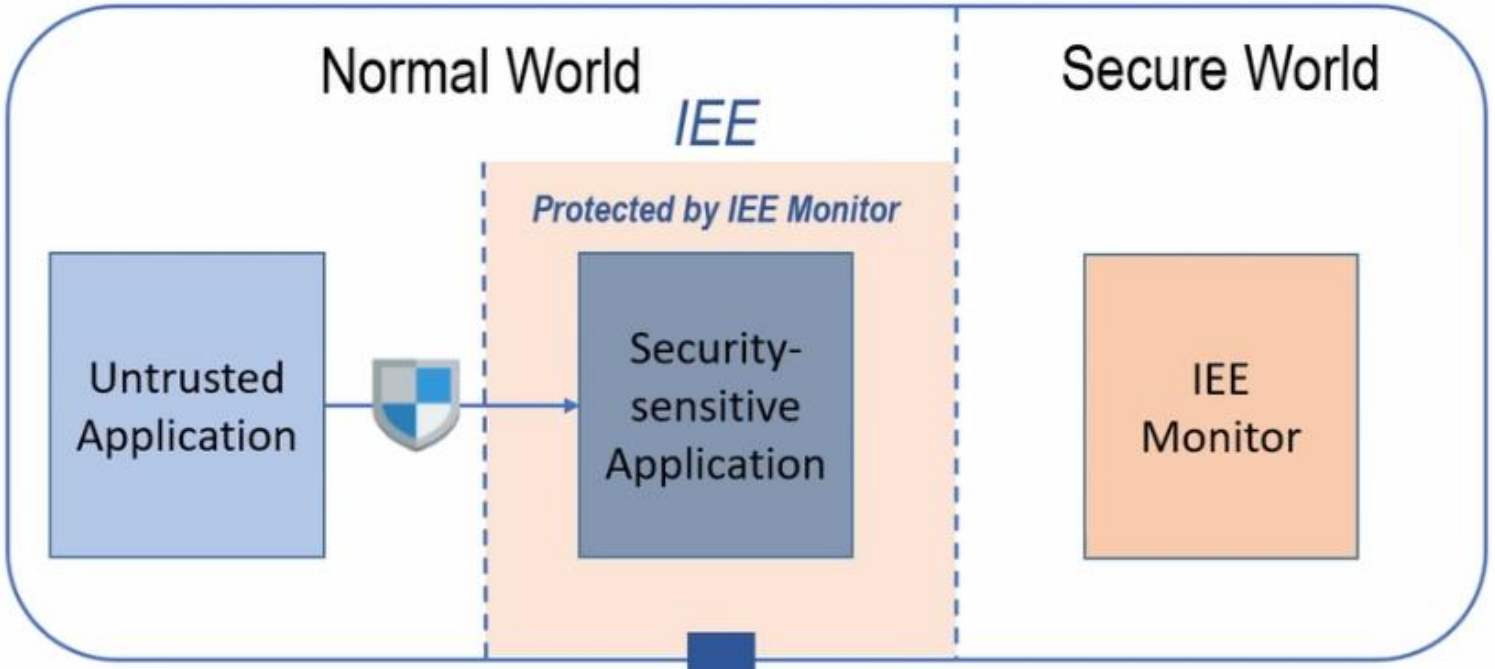
Introducing a new design: Isolated Execution Environment

Isolated Execution Environment (IEE)



- Creating Isolated Execution Environments (called IEEs) in the normal world.
- Using the IEE monitor in the secure world to ensure the security of IEEs.

Isolated Execution Environment (IEE)



Improving the limitation of TEE systems

- Minimize the TCB of the secure world by only installing an IEE Monitor.
- More third-party applications can be imported for the enhanced security protection.

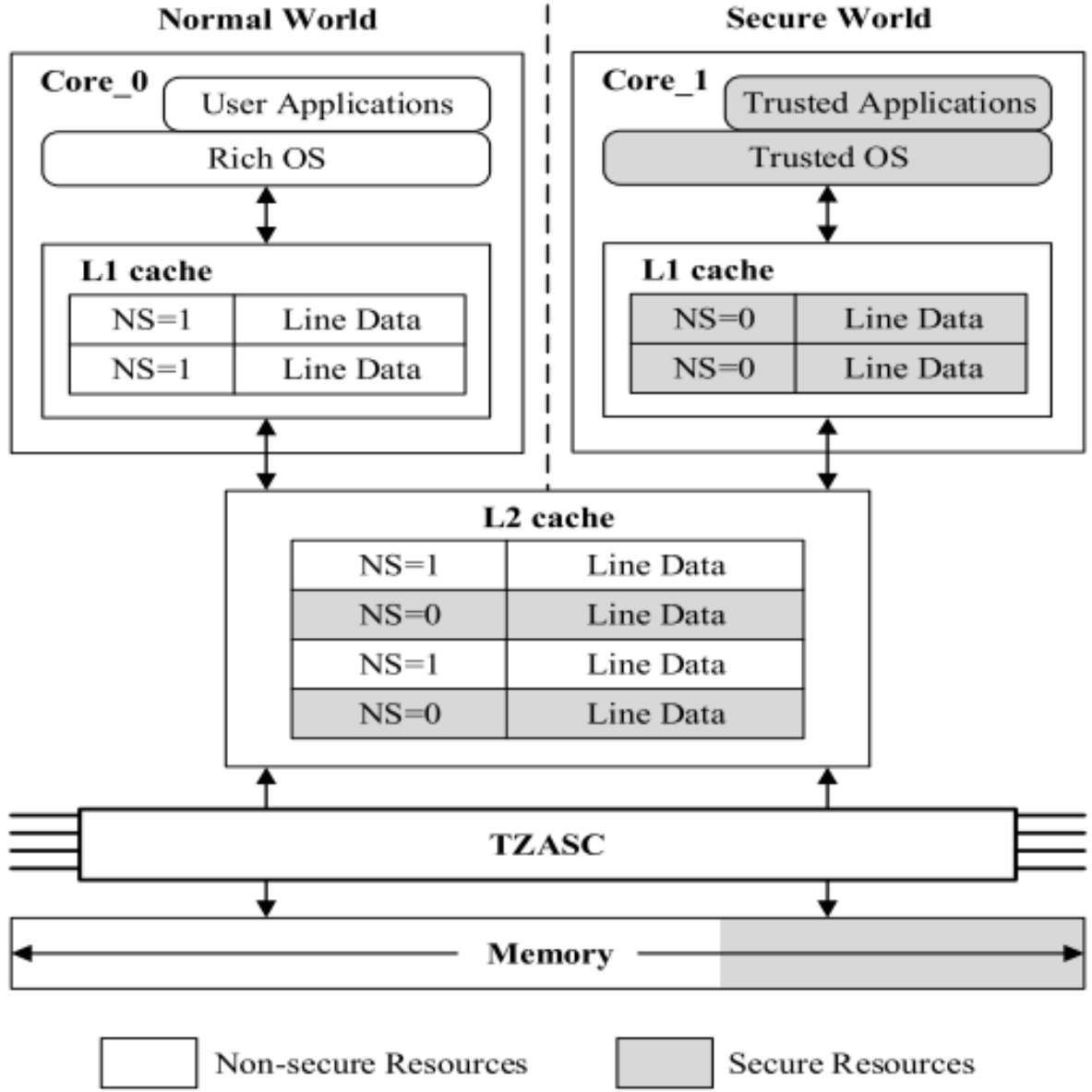
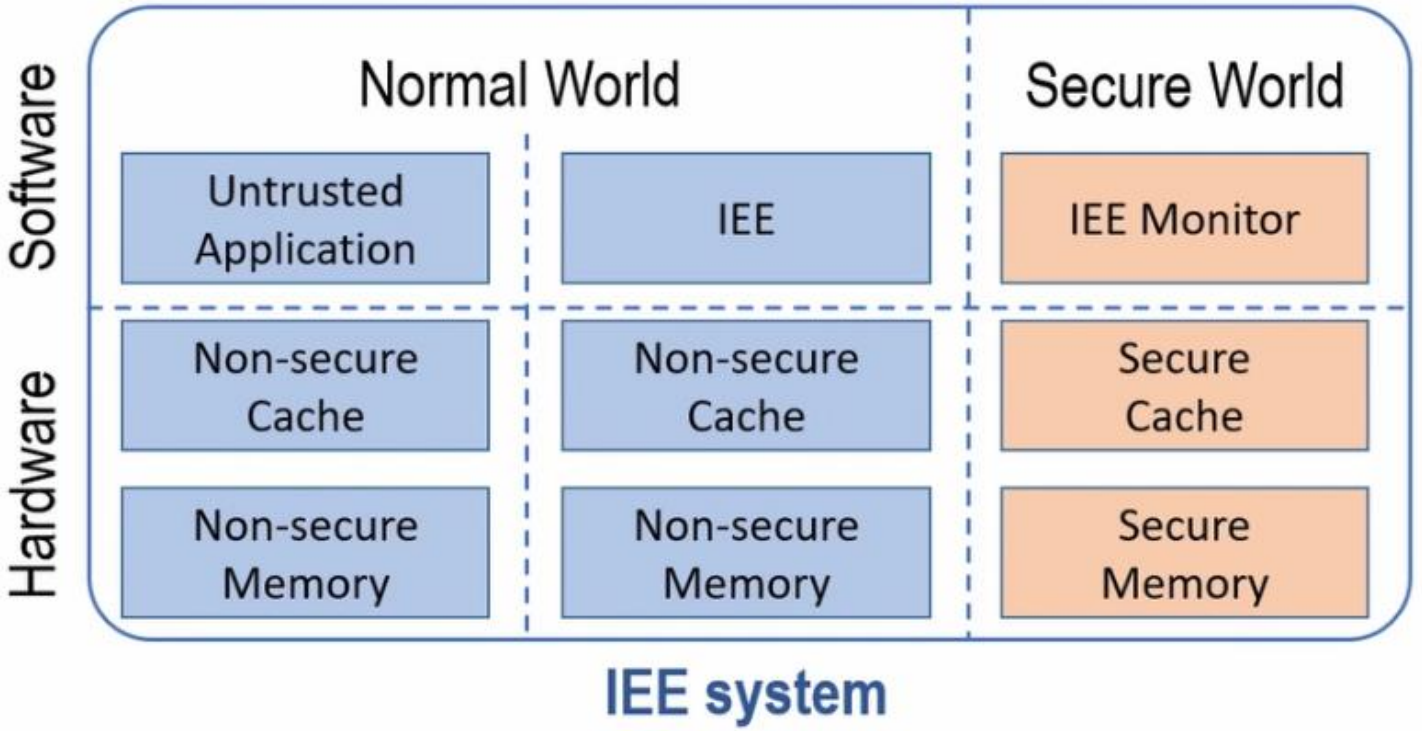
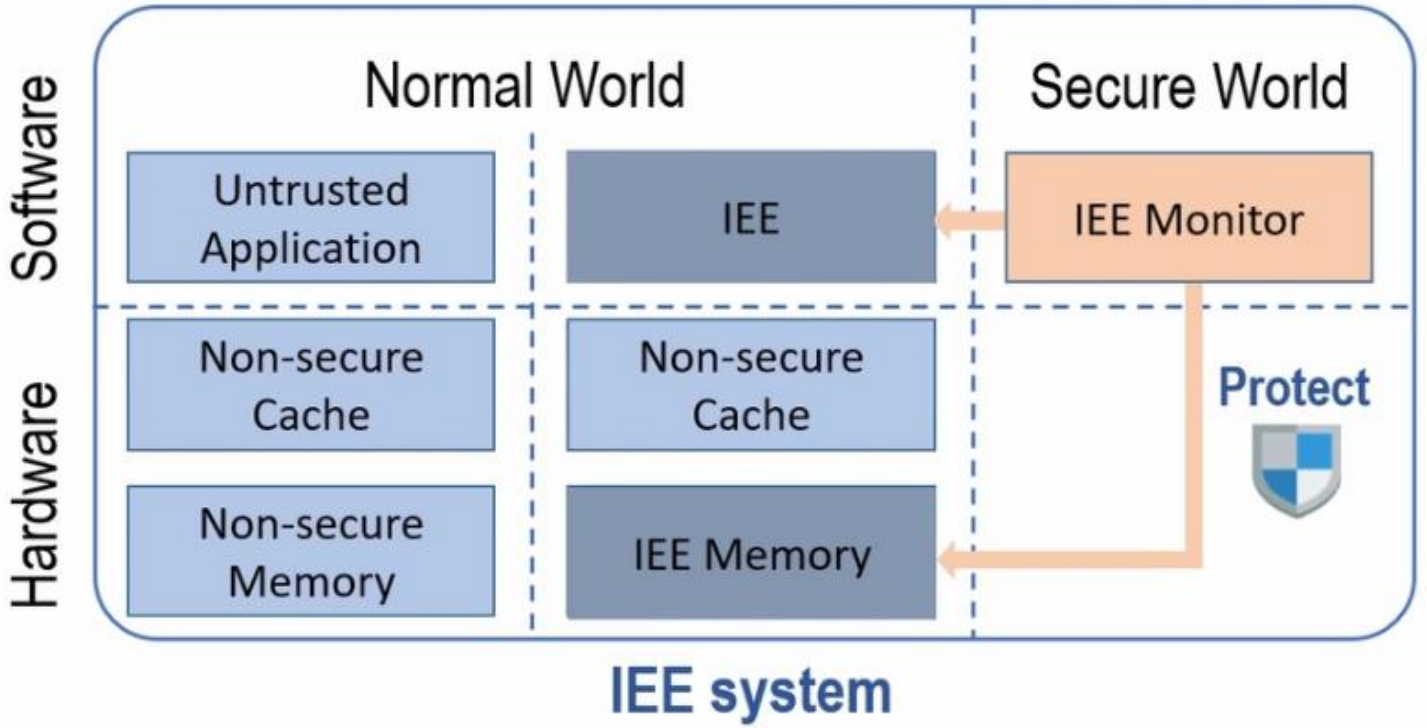


Figure 2: Architecture of ARM TrustZone

Cache-in-the-Middle (CITM) Attacks

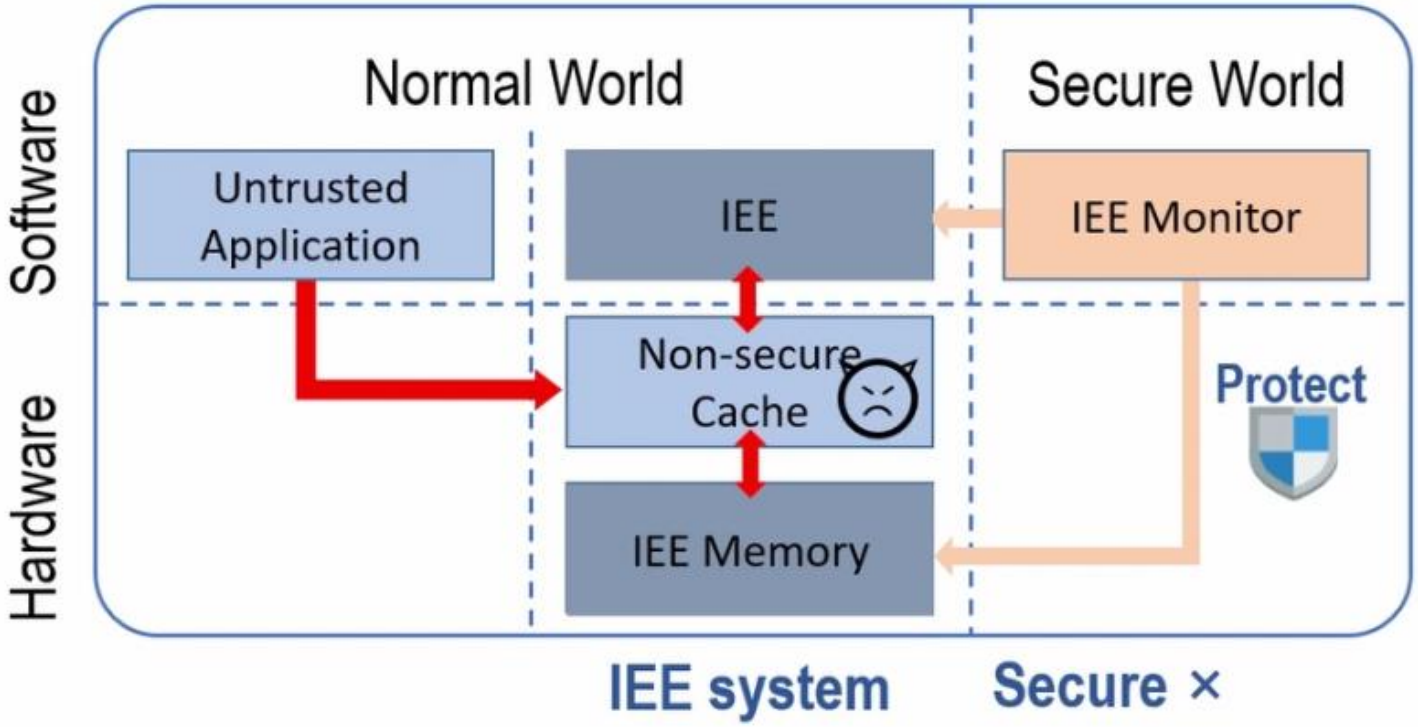


Cache-in-the-Middle (CITM) Attacks



Some existing systems ignore the security of data in the cache.

Cache-in-the-Middle (CITM) Attacks



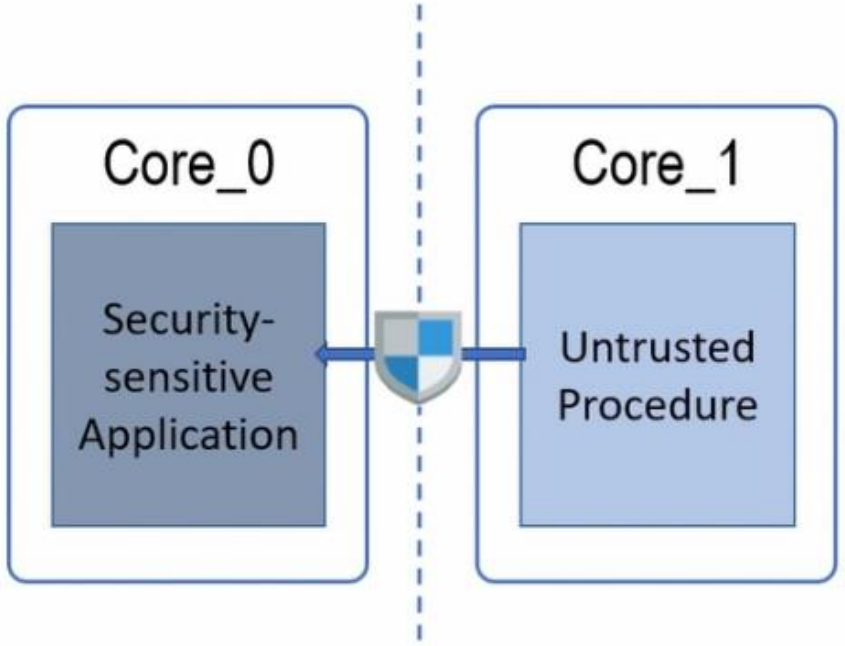
Some existing systems ignore the security of data in the cache.

Attackers can manipulate the cache to influence the protection of IEE systems.

Data Protection Model of IEE Systems

IEE systems are protected ① when they are running concurrently with untrusted procedures, ② when they are suspended or finished and ③ when they are resumed or started.

①



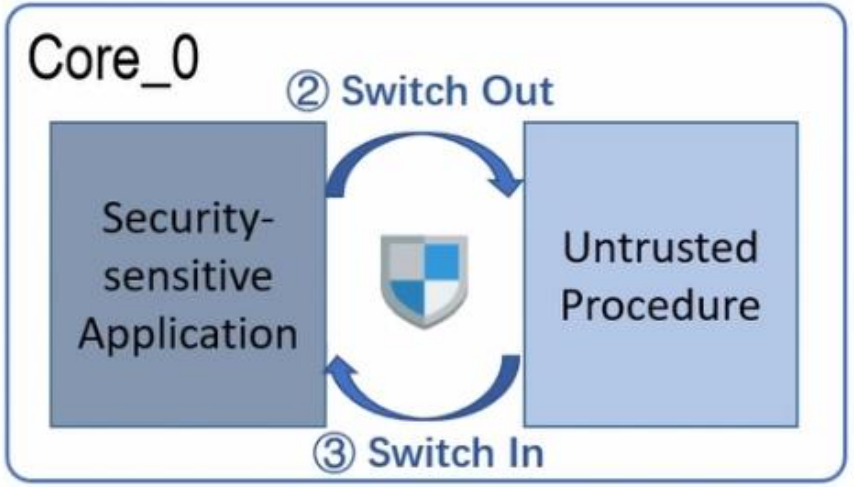
Core-isolated storage

The core-isolated storage can only be accessed by the core executing security-sensitive applications and is inaccessible to the other cores.

Data Protection Model of IEE Systems

IEE systems are protected ① when they are running concurrently with untrusted procedures, ② when they are suspended or finished and ③ when they are resumed or started.

②、③



Enforcing security measures during the context switching processes.

Preventing sensitive data leakage during switching out.

Restoring the sensitive data during switching in.

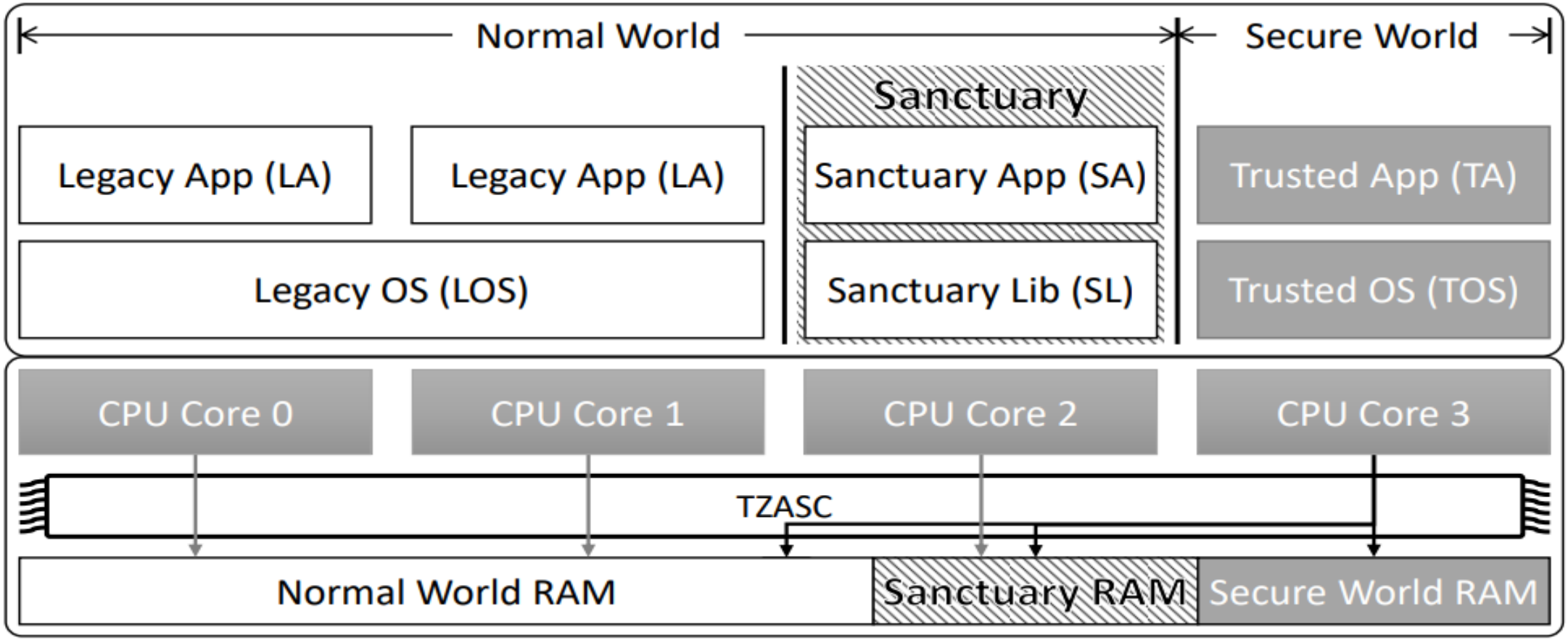
Data Protection Model of IEE Systems

- **Core-isolated storage**
  - Attack I: Manipulating data of core-isolated memory.
- **Security measures during the context switching processes**
  - Attack II: Bypassing security measures.
  - Attack III: Misusing incomplete security measures.

Data Protection Model of IEE Systems

- **Core-isolated storage**
  - Attack I: Manipulating data of core-isolated memory.
- **Security measures during the context switching processes**
  - Attack II: Bypassing security measures.
  - Attack III: Misusing incomplete security measures.

SANCTUARY



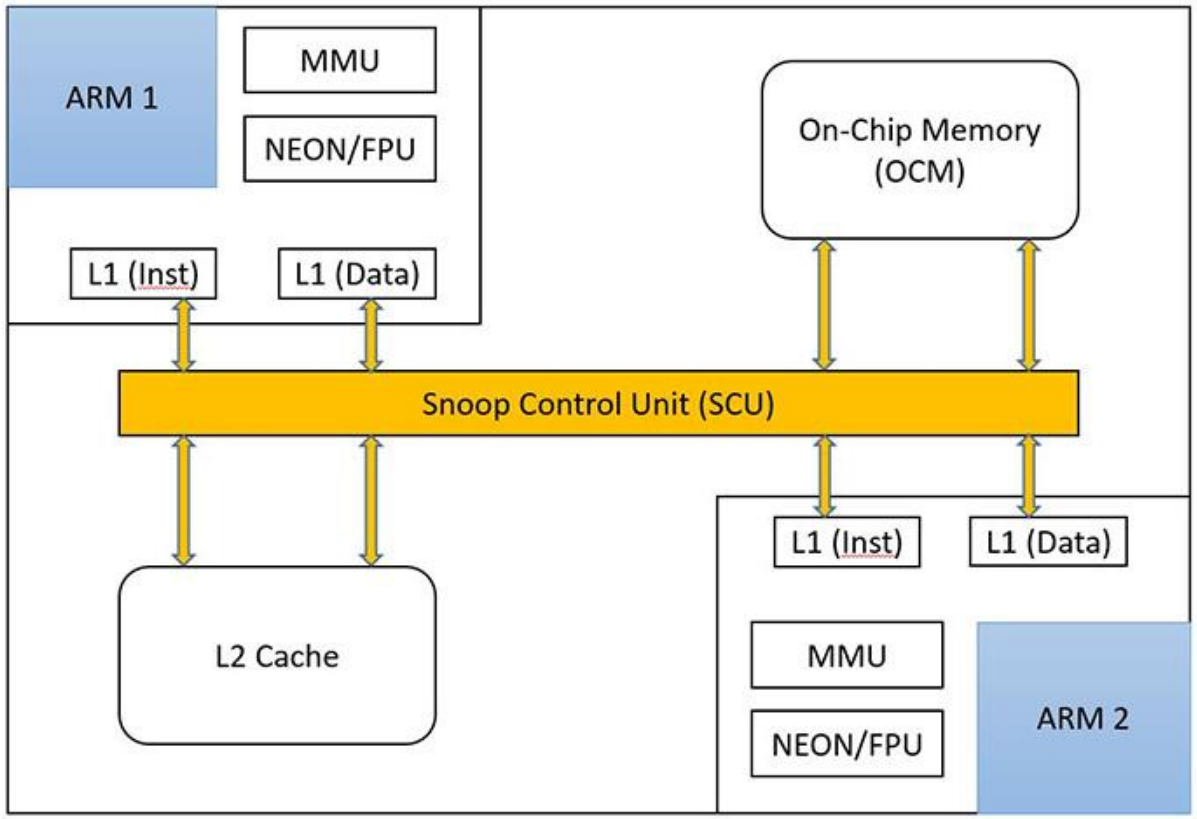
Brasser F, Gens D, Jauernig P, et al. SANCTUARY: ARMing TrustZone with User-space Enclaves[C]//NDSS. 2019.

SANCTUARY Data Protection Mechanisms

- IEE 运行结束时，微内核清理敏感数据
- IEE 运行前，monitor in Secure-World 为其构建一个干净的环境
- 在IEE 运行前后，core-isolated 内存和L1cache 都被**微内核**安全清理
- L2 cache 禁用，禁止在多核之间共享数据

Attack in SANCTUARY

- L1 cache 位于核内部，**不能直接**从其他核访问，在并发执行时，不会对L1 cache提供**额外保护**
- 可通过操作一个核的L1 data cache 读写另一个核的L1 data cache
- 如果其中一个核的 L1 data cache 被修改，当其他同簇核的 L1 data cache 对应的内存页设置为 **inner shareable** 时，SCU会同步更改 L1 data cache
- 当内存页被设置为 outer shareable 时，更改将被同步到所有其他内核
- non-secure cache 与secure cache 不互通数据



SANCTUARY L1 data cache

- A quad-core 1.2GHz ARM Cortex-A9
- 1GB DDR3 SDRAM

- 四个核均设置为NW
- 禁用L2 cache
- 测试内存设置为secure，write-back、write-allocate，其余内存设置为non-cacheable

- 测试内存 L1 cache 初始为0

Table 1: L1 Cache When Enabling Shareable Attribute

Shareability Attribute of the Cores	Value on the Core's L1 Data Cache	
	After Writing 0xffff to Core_0	After Writing 0xdddd to Core_1
Core_0 (Shareable)	0xffff	0xdddd
Core_1 (Shareable)	0xffff	0xdddd
Core_2 (Shareable)	0xffff	0xdddd
Core_3 (Shareable)	0xffff	0xdddd

Table 2: L1 Cache When Disabling Shareable Attribute

Shareability Attribute of the Cores	Value on the Core's L1 Data Cache	
	After Writing 0xffff to Core_0	After Writing 0xdddd to Core_1
Core_0 (Non-shareable)	0xffff	0xffff
Core_1 (Shareable)	0x0	0xdddd
Core_2 (Shareable)	0x0	0xdddd
Core_3 (Shareable)	0x0	0xdddd

Attack Procedure

- L1 cache 可被跨核修改

- ①为 core_1 创建一个页表条目，将内存页的缓存属性配置为 shareable
- ②使其物理地址指向 core_0 的内存页(即受SANCTUARY保护的内存页),
- ③当访问 core_1上对应的虚拟地址时，由于 shareable 属性保证了值的一致性，core_0的 L1 data cache 中的敏感数据可以被窃取或修改

- 获取IEE内存地址
- entire physical memory divided into three parts
- IEE memory对应的cache是 NS
- TEE memory对应的cache是 S
- 读取TEE内存总返回0或异常
- 读取缓存在L1中IEE data 能够返回正常值

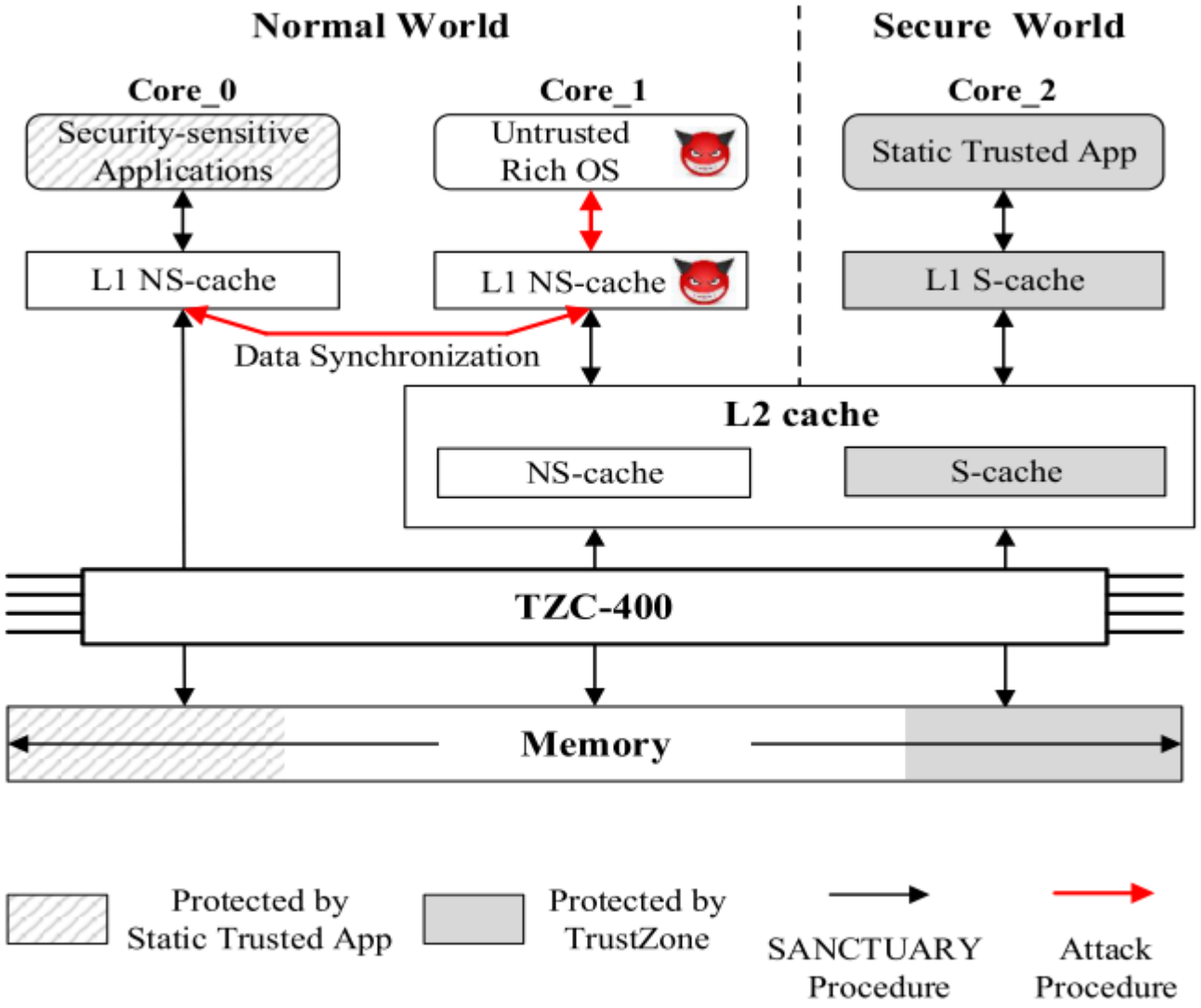
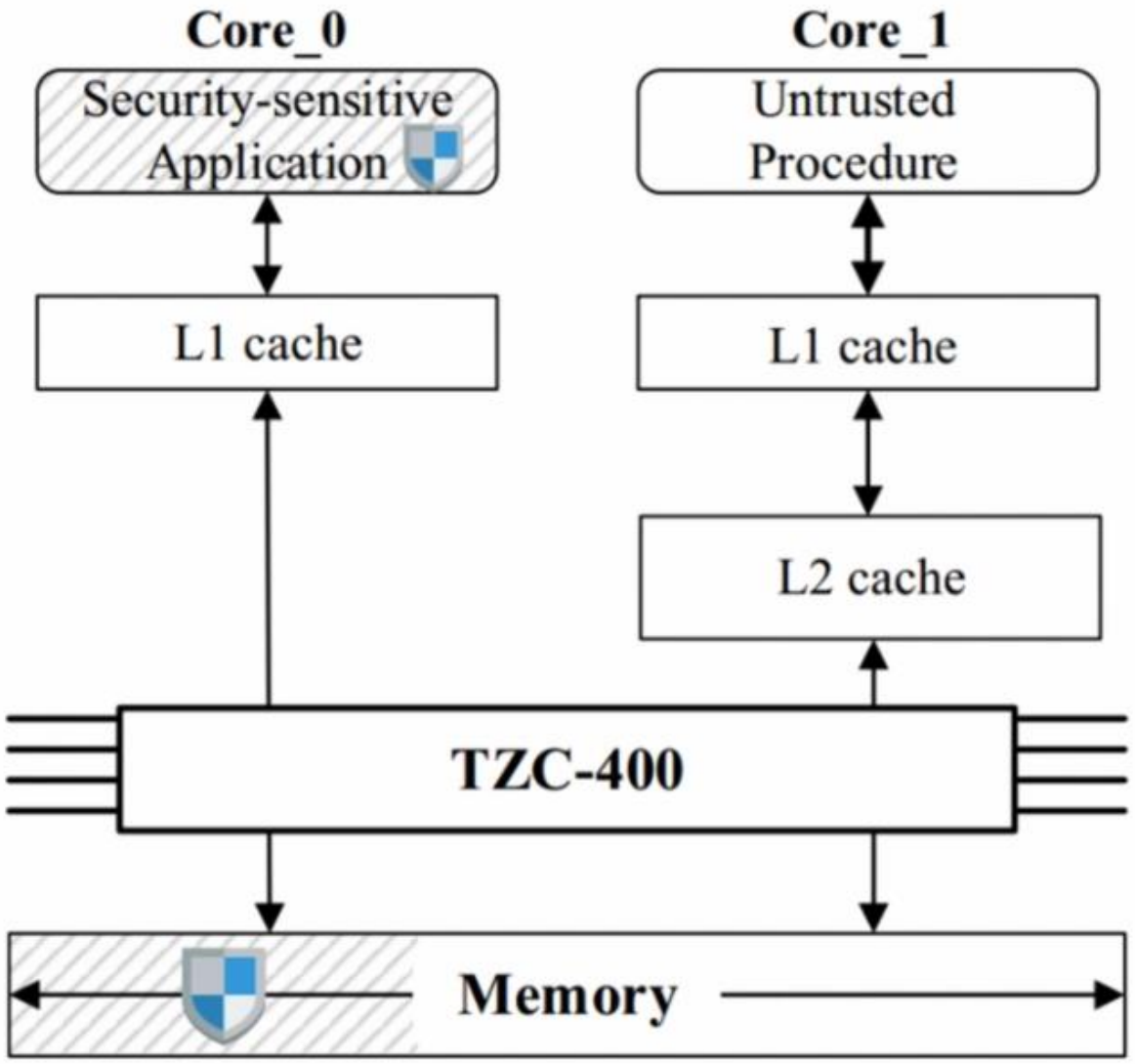


Figure 4: CITM Attack on SANCTUARY

Attack I: Manipulating Data of Core-isolated Memory

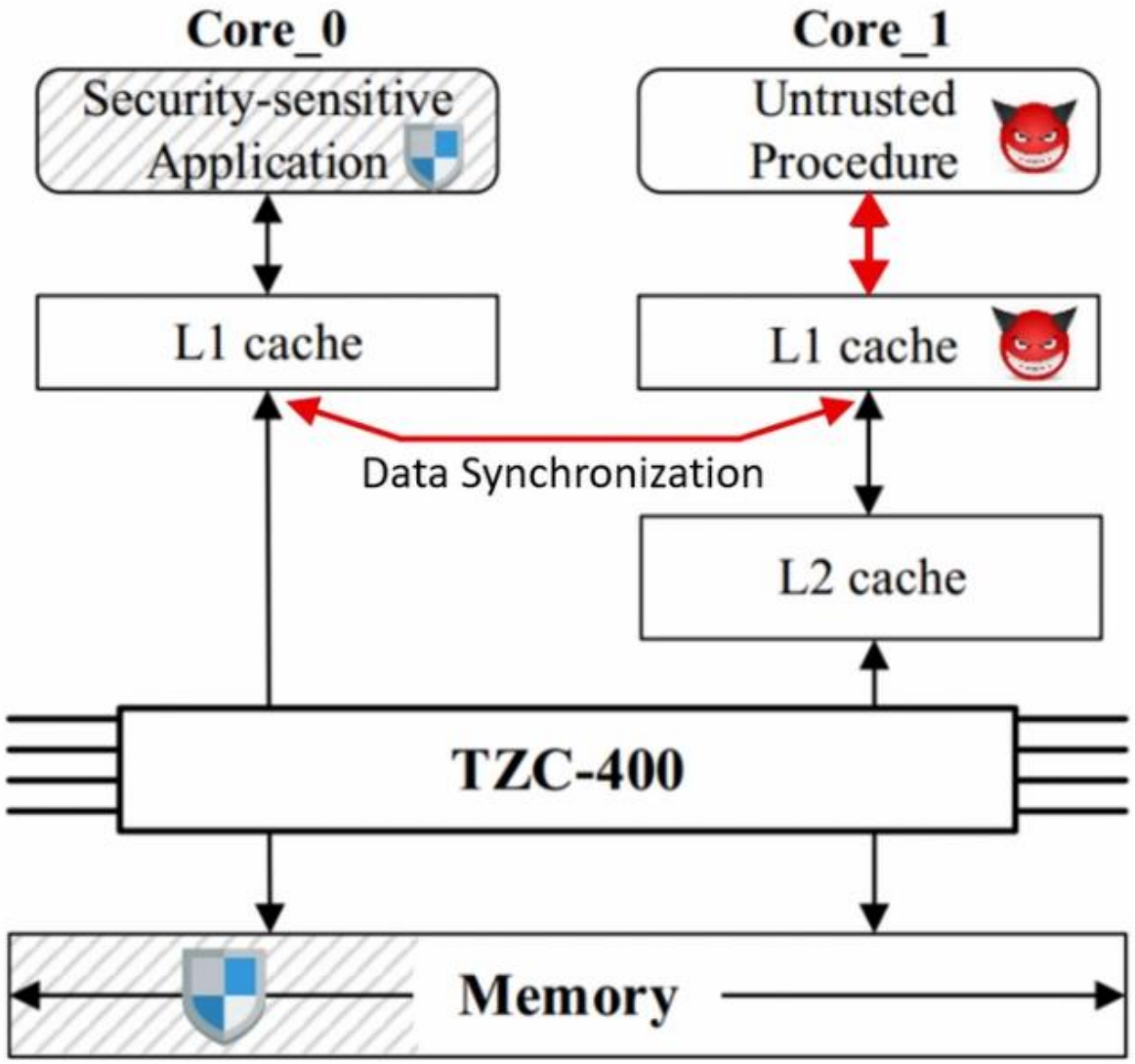


Configuration of core-isolated storage.

(e.g., SANCTUARY)

- Configuring core-isolated memory.
- Excluding the L2 shared cache.

Attack I: Manipulating Data of Core-isolated Memory



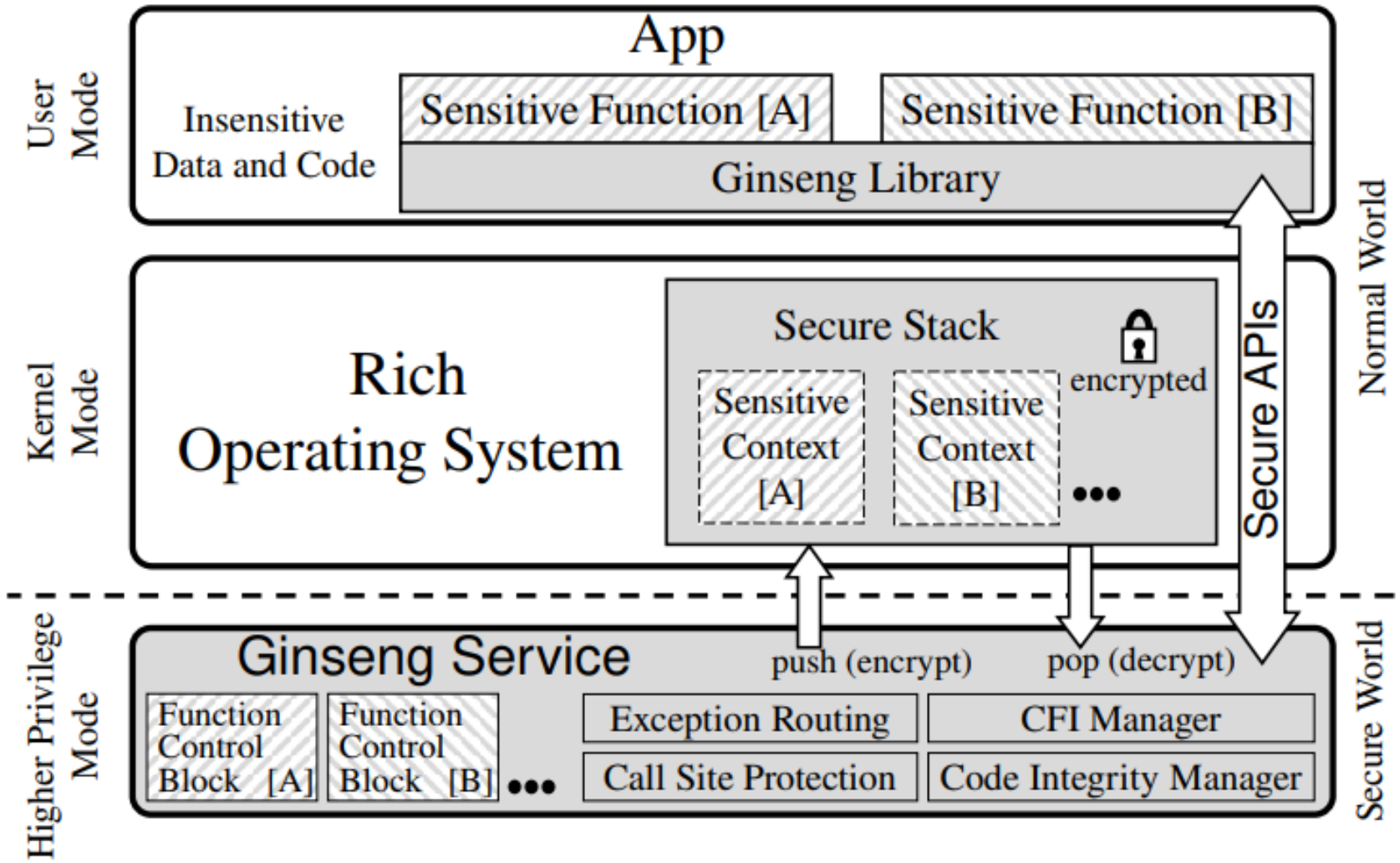
Utilizing the shareability attribute of L1 cache.

- Value coherency of L1 data cache.
- Manipulating L1 data cache to get data of core-isolated memory.

Data Protection Model of IEE Systems

- **Core-isolated storage**
  - Attack I: Manipulating data of core-isolated memory.
- **Security measures during the context switching processes**
  - Attack II: Bypassing security measures.
  - Attack III: Misusing incomplete security measures.

Ginseng



Yun M H, Zhong L. Ginseng: Keeping Secrets in Registers When You Distrust the Operating System[C]//NDSS. 2019.

Data Protection Mechanisms

- Secure-world 分发秘钥（本例中秘钥为敏感数据）
- 涉及敏感数据的函数被识别为敏感函数，执行前进行代码**完整性校验**
- 非敏感函数执行前**不进行**完整性检查
- 敏感型操作在**寄存器**中进行，非内存

- 提供6个安全API，将控制流从正常用户空间直接传输至Gservice
- 编译时自动插入程序中，代码检查+寄存器清理
- 传输过程中自动加密解密，防止数据泄露

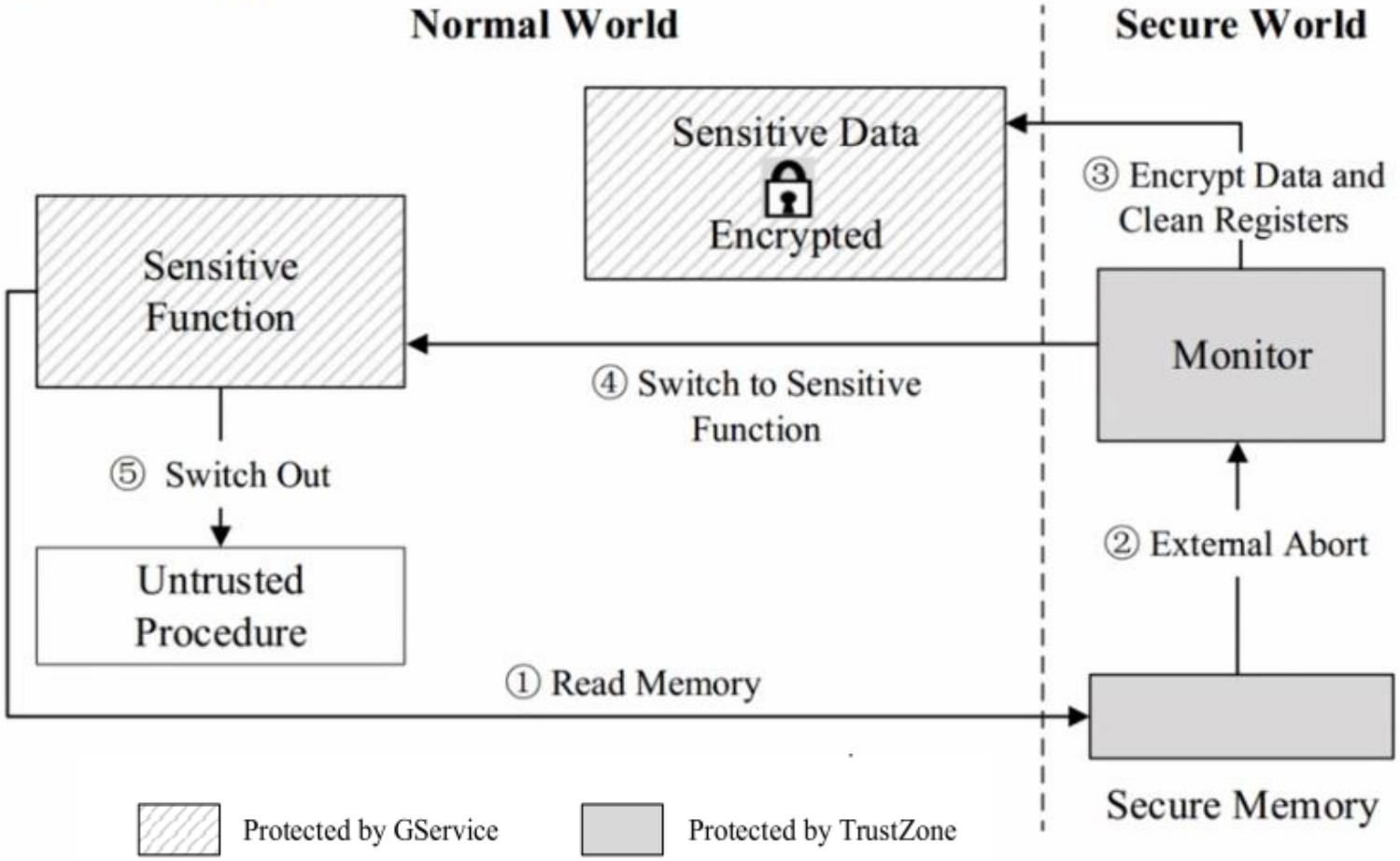
- Ginseng是用户态，不能调用搞特权SMC指令
- 使用外部中断 (EA) 的方式捕获EA并处理API请求

```
1  /*sensitive function*/
2  int genCode(sensitive long key_top,
3              sensitive long key_bottom) {
4      // operations for insensitive data
5      ...
6      // invoke sensitive function
7      hmac_sha1(key_top, //sensitive data
8          key_bottom,    //sensitive data
9          challenge,     //insensitive data
10         resultFull);   //insensitive data
11     // truncate 20-byte hmac_sha1() result to 4-byte
       ↪ truncatedHash
12     ...
13     // invoke insensitive function
14     printf("OTP: %06d\n", truncatedHash);
15     return truncatedHash;
16 }
17
18 /*sensitive function*/
19 void run(){
20     // mark the protected data as sensitive
21     sensitive long key_top, key_bottom;
22     // read keys from TEE secure world
23     ss_read(UUID1, UUID2, key_top);
24     ss_read(UUID3, UUID4, key_bottom);
25     // invoke sensitive function
26     genCode(key_top, key_bottom);
27 }
```

Listing 1: A Sample Program Protected by Ginseng

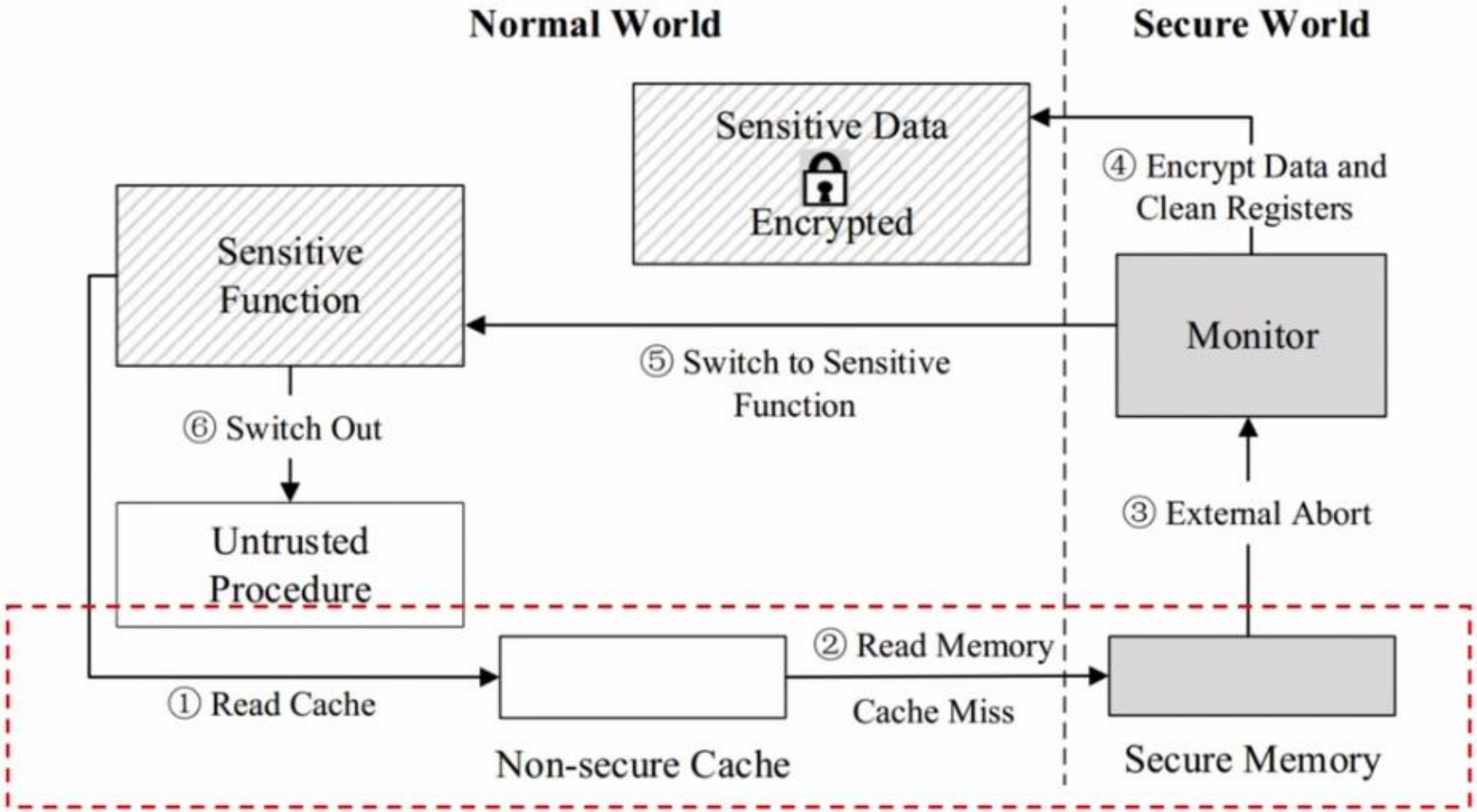
Attack II: Bypassing Security Measures

Accessing secure memory to trigger security measures when switching out (e.g., Ginseng)



Attack II: Bypassing Security Measures

Analyzing the security measures with cache



Ginseng exploites the cache

- __channel_save_clean
- ss_savecleanV (API) 的 secure memory

- Secure memory attribute
- Write-back write-allocate
- cache lockdown technique prevent the cache eviction

- writeSM 写入 secure memory对应cache
- 1. secure MEM 虚拟地址（保存在rich OS 中）加载到x4;
- 2. 寄存器x0 数据存储到位于x4的安全内存
- **3. 数据写入安全内存的缓存**
- **4. 安全内存属性，只有满才换出**
- **5. 缓存锁定技术防止缓存回收**

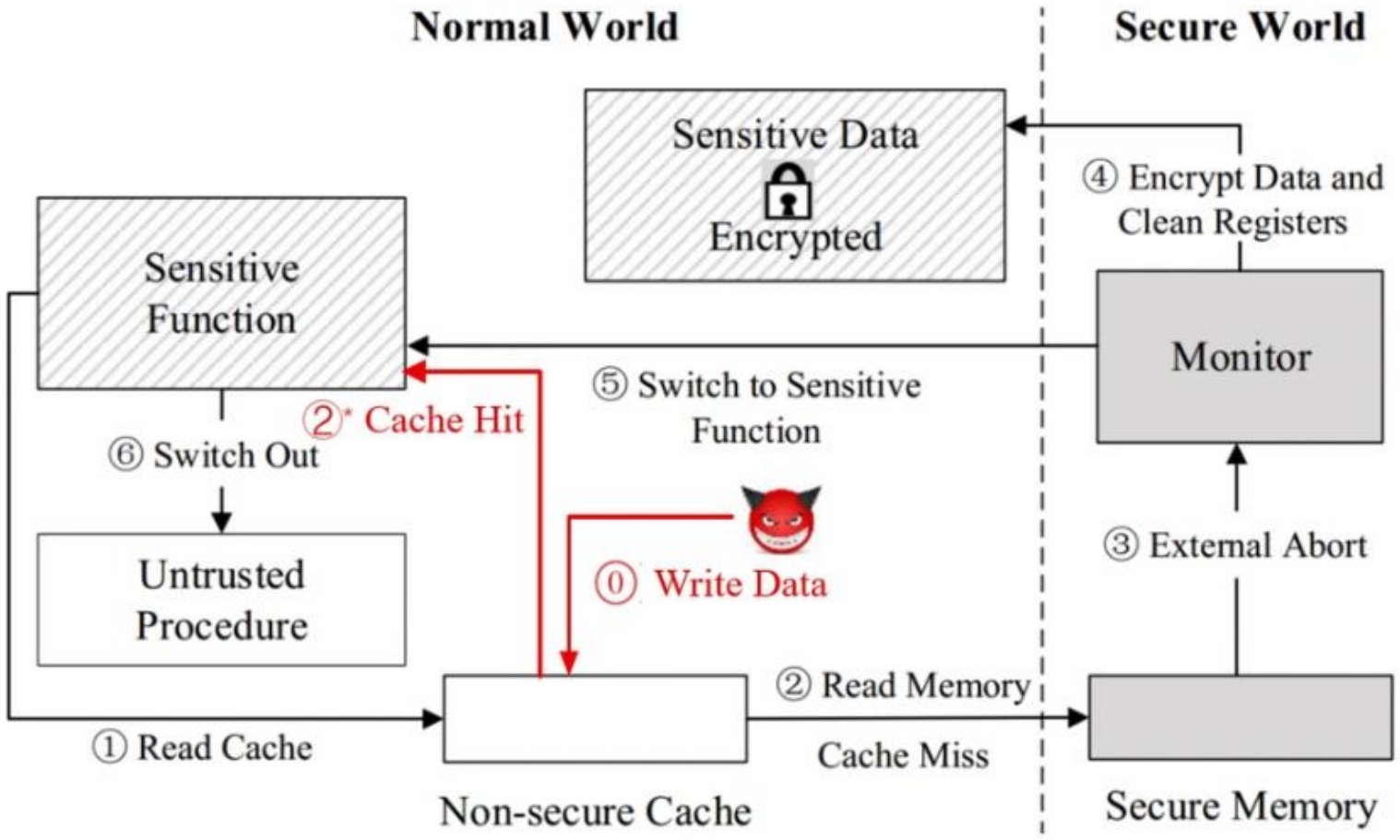
- 攻击实现
- a 8-core ARM Cortex-A53 processor
- HiKey620 development board

```
1   /* Attack preparation: fill in the corresponding cache in
    ↪  advance*/
2   writeSM:
3       /* __channel_save_clean:
4       virtual address of the secure memory
5       assigned to ss_saveCleanV*/
6       ldr     x4, =__channel_save_clean
7       /* store data to secure memory */
8       str     x0, [x4]
9       ret
10
11  /*Implementation of the secure API ss_saveCleanV in Ginseng*/
12  ss_saveCleanV:
13      /* __channel_save_clean:
14      virtual address of the secure memory
15      assigned to ss_saveCleanV*/
16      ldr     x4, =__channel_save_clean
17      /* load data from secure memory */
18      ldr     x0, [x4]
19      ret
```

Listing 2: Exploiting the Cache of Secure Memory

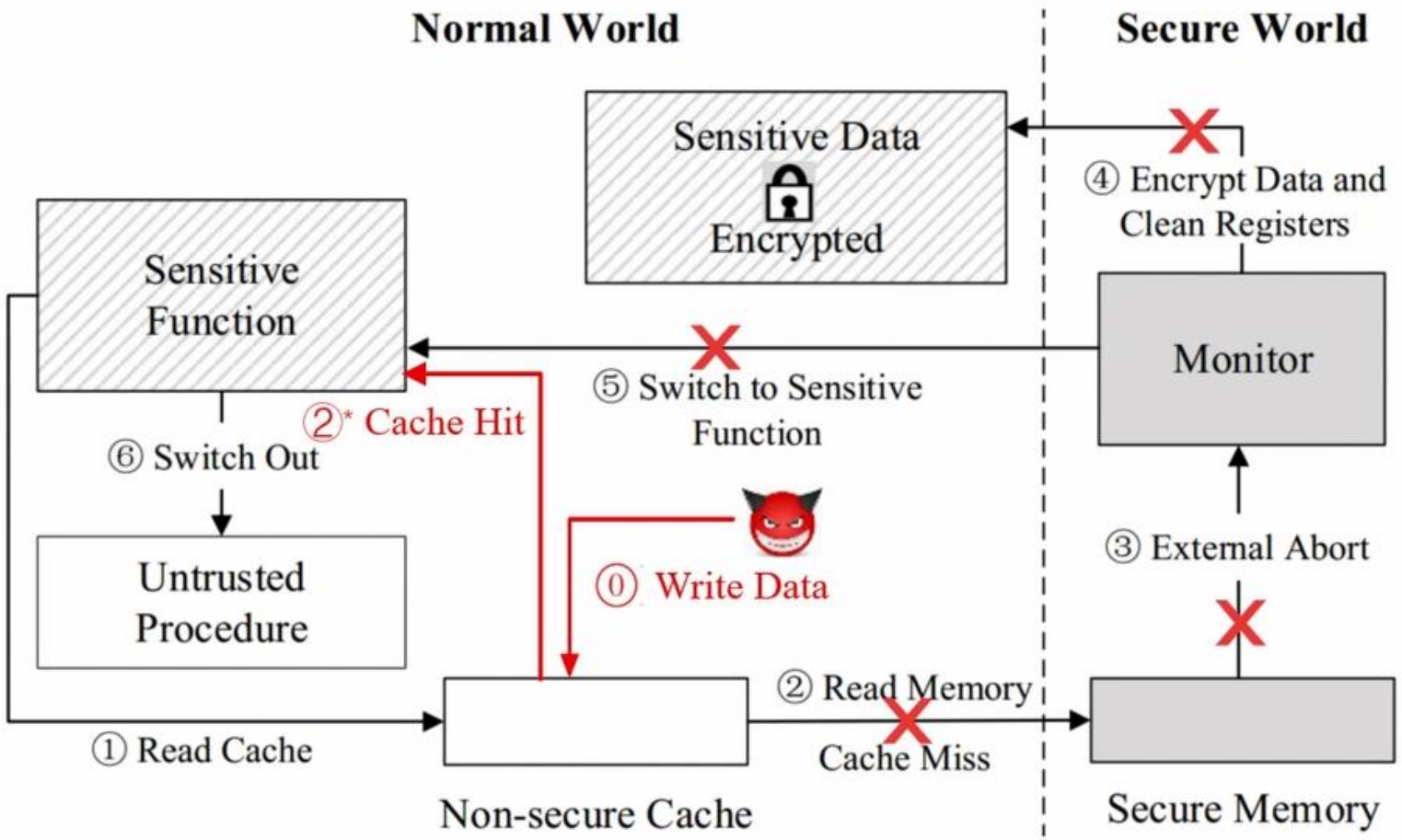
Attack II: Bypassing Security Measures

Bypassing the security measures



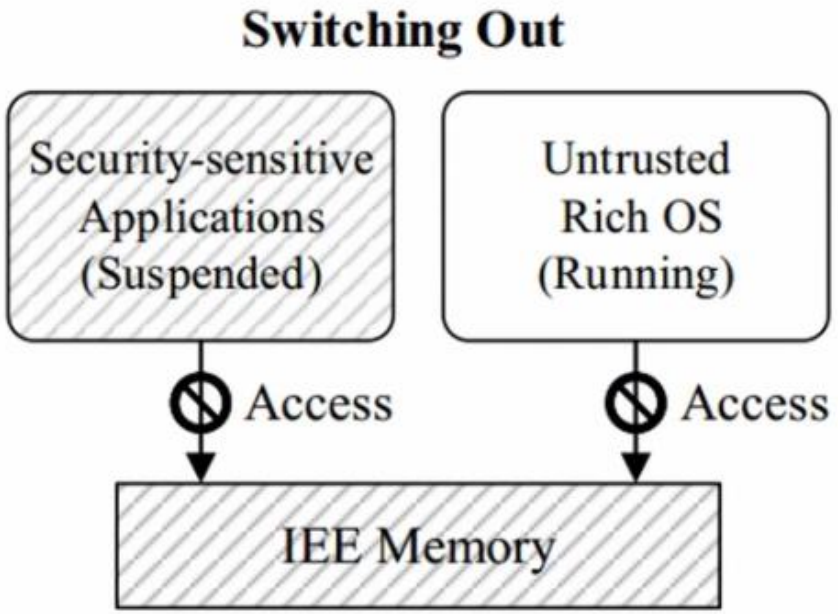
Attack II: Bypassing Security Measures

Bypassing the security measures



Attack III: Misusing Incomplete Security Measures

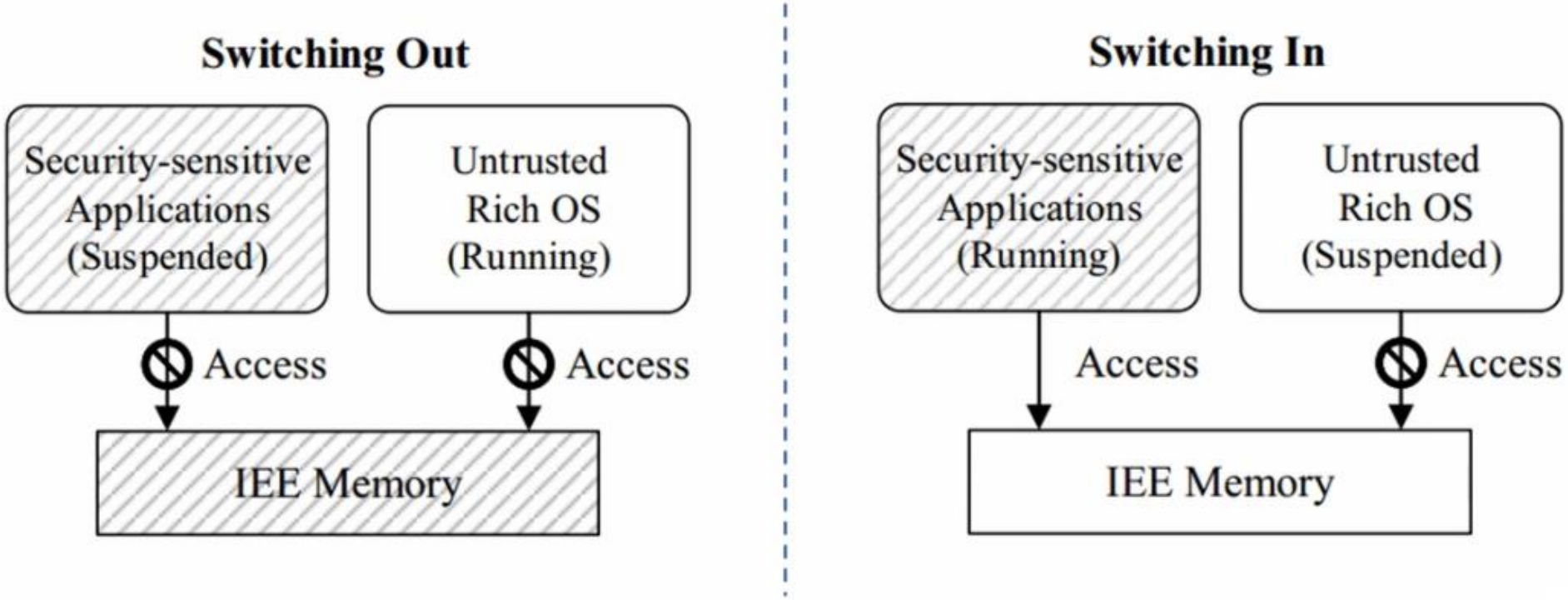
Configuring memory during the context switching processes (e.g., TrustICE)



- The switching out process configures the memory as secure.

Attack III: Misusing Incomplete Security Measures

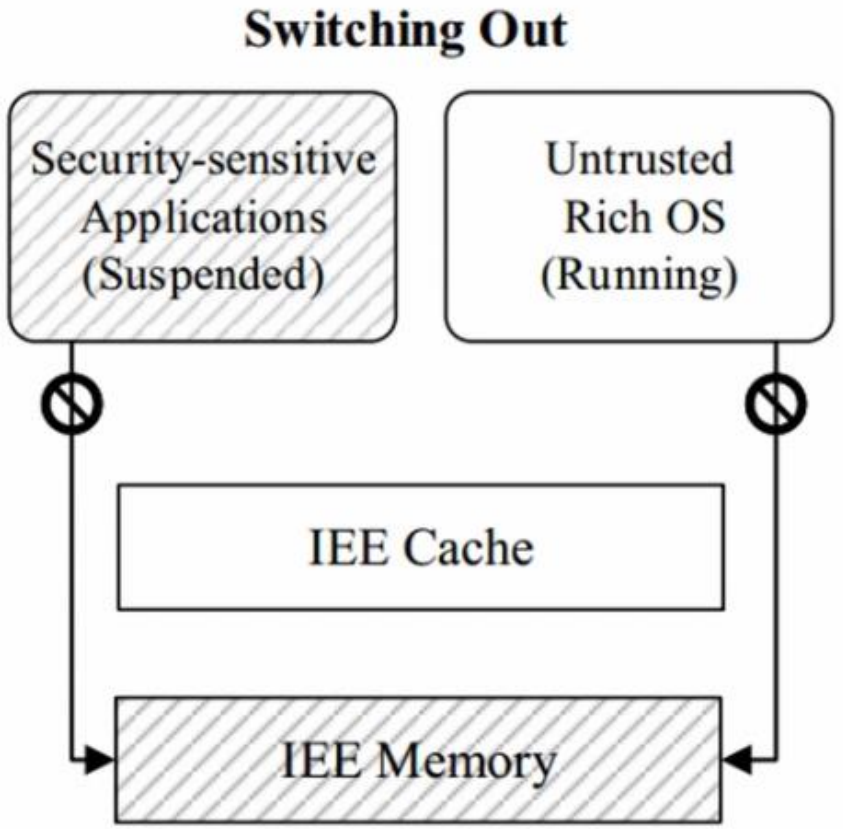
Configuring memory during the context switching processes (e.g., TrustICE)



- The switching out process configures the memory as secure.
- The switching in process configures the memory as non-secure and suspends the untrusted rich OS.

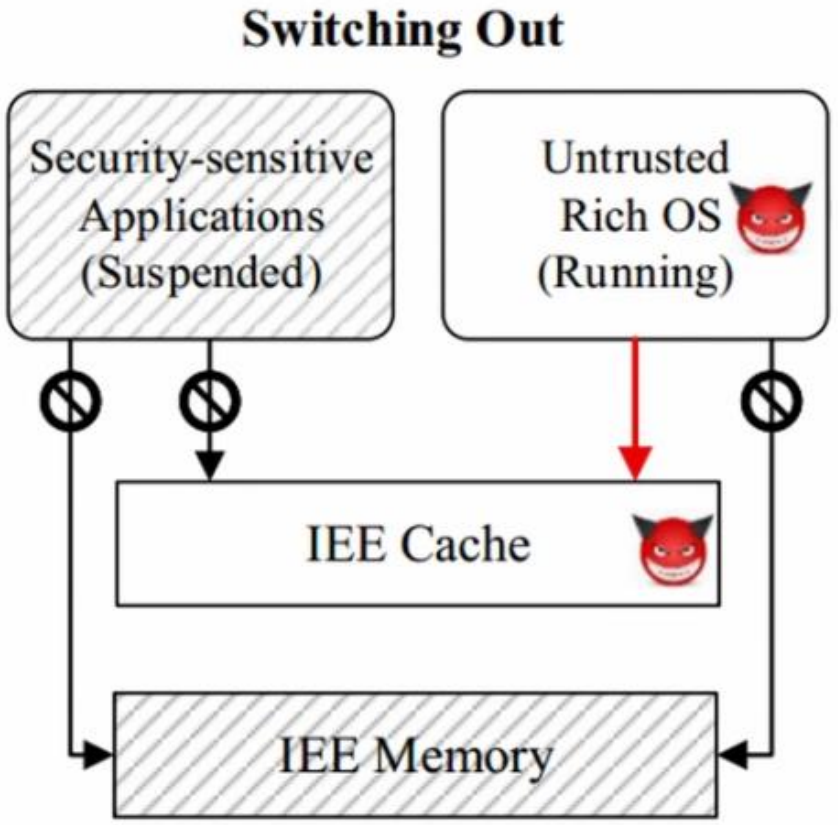
Attack III: Misusing Incomplete Security Measures

Memory configuration doesn't influence the security of cache



Attack III: Misusing Incomplete Security Measures

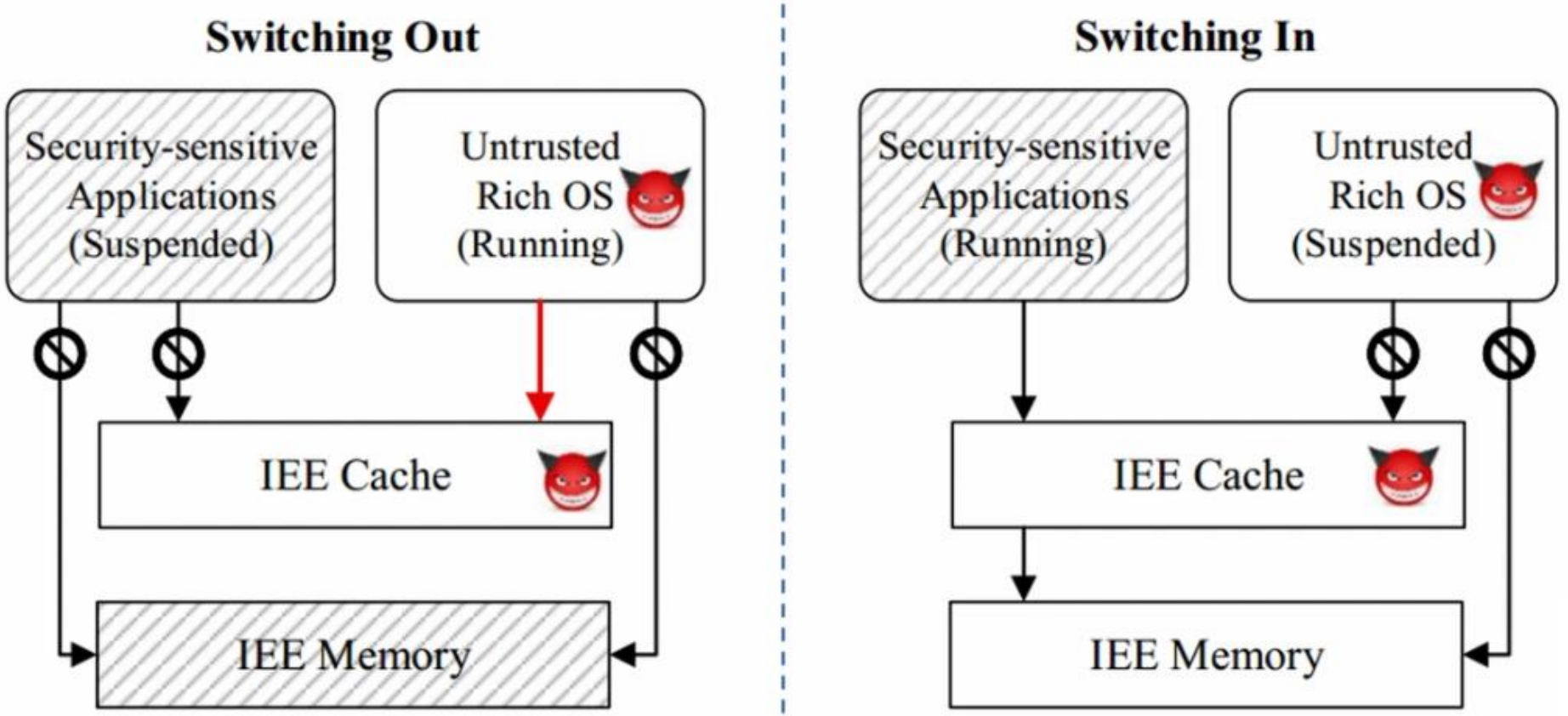
Memory configuration doesn't influence the security of cache



- Manipulating IEE cache when untrusted rich OS is running.

Attack III: Misusing Incomplete Security Measures

Memory configuration doesn't influence the security of cache



- Manipulating IEE cache when untrusted rich OS is running
- Reading polluted IEE cache when security-sensitive applications are running

安全措施

- 配置IEE内存的缓存属性
- inner write-through、non-write-allocate、outer non-cacheable、non-shareable

- 在环境切换时清理IEE内存的缓存
- 在“switch in”和“switch out”进程中清除 cache line

Evaluation

- i.MX6Quad SABRE development board
- a quad-core ARM Cortex-A9 1.2GHz with 1GB DDR3 SDRAM
- 1,000 iterations and report the average
- Benchmark AnTuTu 2.9.4

Table 4: Benchmark Results on Rich OS

Test Item	Protection Disabled	Protection Enabled	Overhead
RAM	486	475	2.26%
CPU Integer	698	692	0.86%
CPU Float-point	567	564	0.53%
2D Graphics	282	281	0.35%
3D Graphics	861	852	1.05%
Database I/O	310	255	17.74%
SD Card Write	38	36	5.26%
SD Card Read	186	182	2.15%
Total	3428	3337	2.65%

Table 5: Loading Time Results on Rich OS (in Seconds)

Test Item	Protection Disabled	Protection Enabled	Overhead
Kernel	22.26	23.71	6.51%
Android Home	87.42	89.81	2.73%
Calculator	3.01	3.22	6.98%
Calendar	3.14	3.34	6.37%
Music	1.26	1.37	8.73%
Settings	3.77	3.95	4.77%

Conclusions

We must realize the importance of considering memory and cache together when designing IEE systems.