Cache-Based Attack and Defense on ARM Platform

Doctoral Dissertation Thesis Defense

Naiwei Liu

10/27/2020

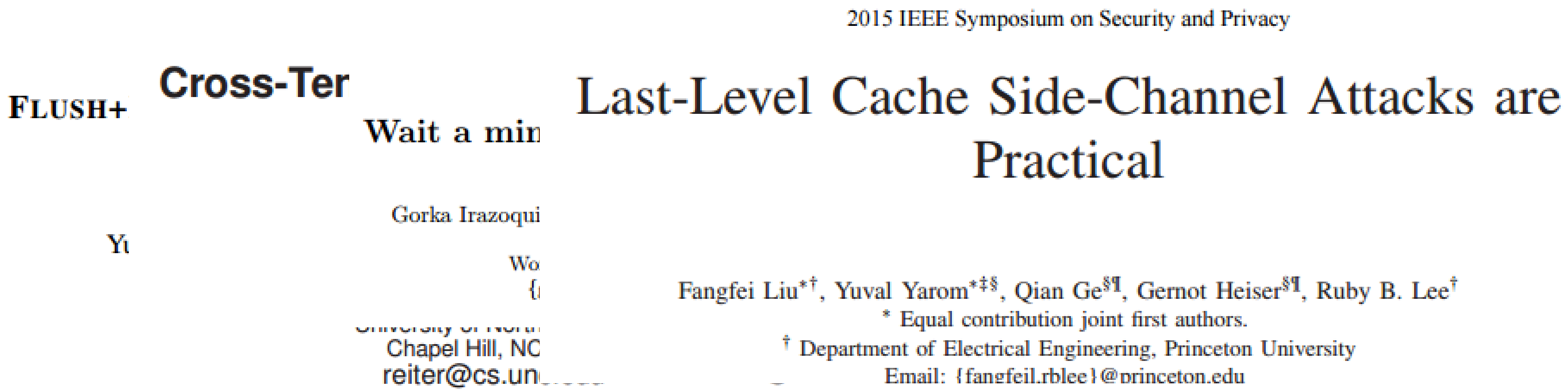
Contents

Introduction

- Abstract
  - In Recent years, many research efforts had been made on secure and safe environment on ARM platform.
  - ARM structure and chips based on ARM had been taking up a lot of number of products in the market.
  - Security problems and potential risks had been discussed.
  - Cache and similar design brings in ‘trouble’ for security purposes.
  - Uniqueness on ARM-based products made things even tougher to solve.
  - What will we do?
    - Design defense framework
    - Evaluate by experiments
    - Optimization

Abstract and Introduction

- Introduction
  - Last-Level Cache (LLC) is always the target of side-channel attack. On x86 structure, it is always L3 cache that is attacked.
  - Last-level cache side-channels are effective enough to extract user’s private information.
  - Side-channel: collecting information like performance counters, timing, power consumption, etc. And process the information to derive information about the victim.
  - Most frequently used: access time-based side-channels.



FLUSH+

Yu

Cross-Ter

Chapel Hill, NC

reiter@cs.un

Wait a min

Gorka Irazoqui

Wo

{

2015 IEEE Symposium on Security and Privacy

Last-Level Cache Side-Channel Attacks are Practical

Fangfei Liu*†, Yuval Yarom*‡§, Qian Ge§¶, Gernot Heiser§¶, Ruby B. Lee†

* Equal contribution joint first authors.

† Department of Electrical Engineering, Princeton University

Email: {fangfeil,rblee}@princeton.edu

Introduction

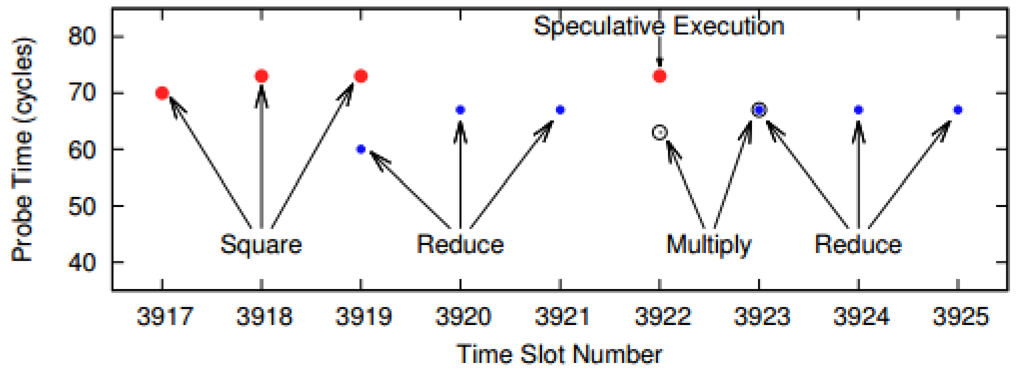
- Introduction (Continued)
  - Side-channel attack based via LLC can be dangerous, even without compromising OS.
  - Both on single OS machine and Virtual Machines (VMs) can be attacked.
  - Most common: FLUSH+RELOAD
    - LLC is shared among processes and threads.
    - FLUSH+RELOAD can be practical using unprivileged instructions.
    - AES key of OpenSSL is recovered by this attack in lab test.
  - Threats to different devices
  - Modern TrustZone Design on ARM platform

Introduction

- Introduction (Continued)
  - Contributions
    - Research on side-channel and covert-channel attack: bandwidth and effect.
    - Investigation on Flush operations on ARM platform and overhead.
    - Study of TrustZone technology and previous security design based on TrustZone.
    - Investigation on critical instructions related to TrustZone operations.
    - Design and test of adaptive control on flush operations.
    - Different discussion based on ARMv8-A and ARMv8-M structures.

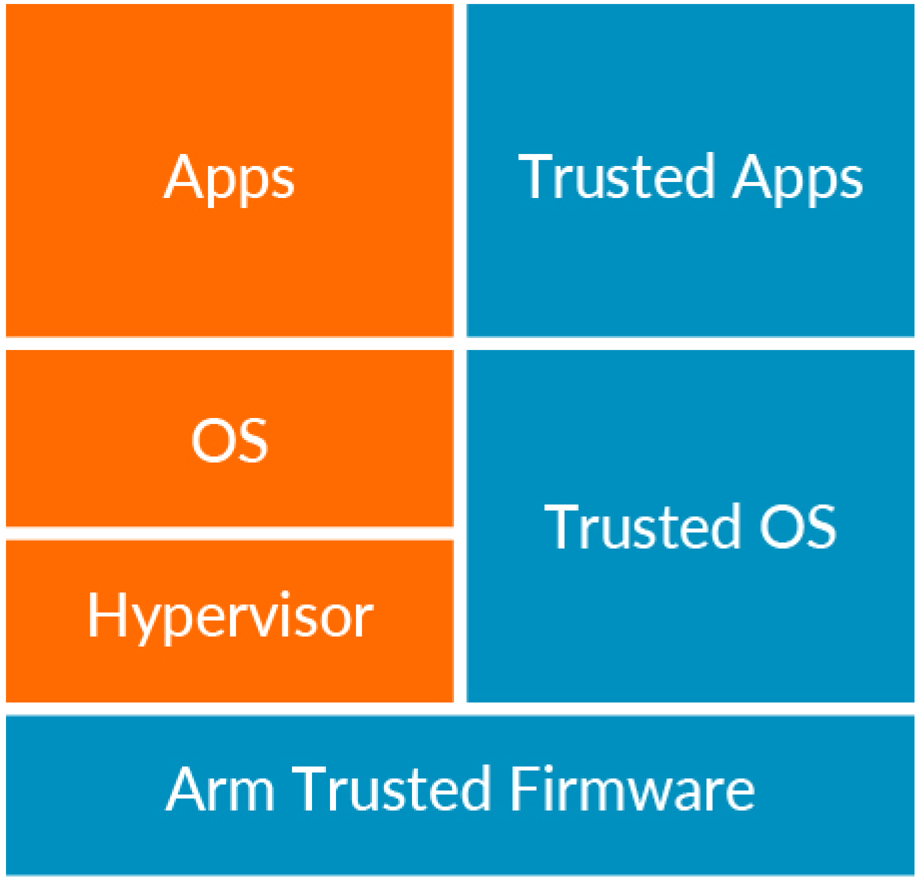
Related Work

- Side Channel Attacks
  - LLC based side-channel attacks: Flush+Reload, Prime+Probe
  - Effectiveness of LLC based side-channels



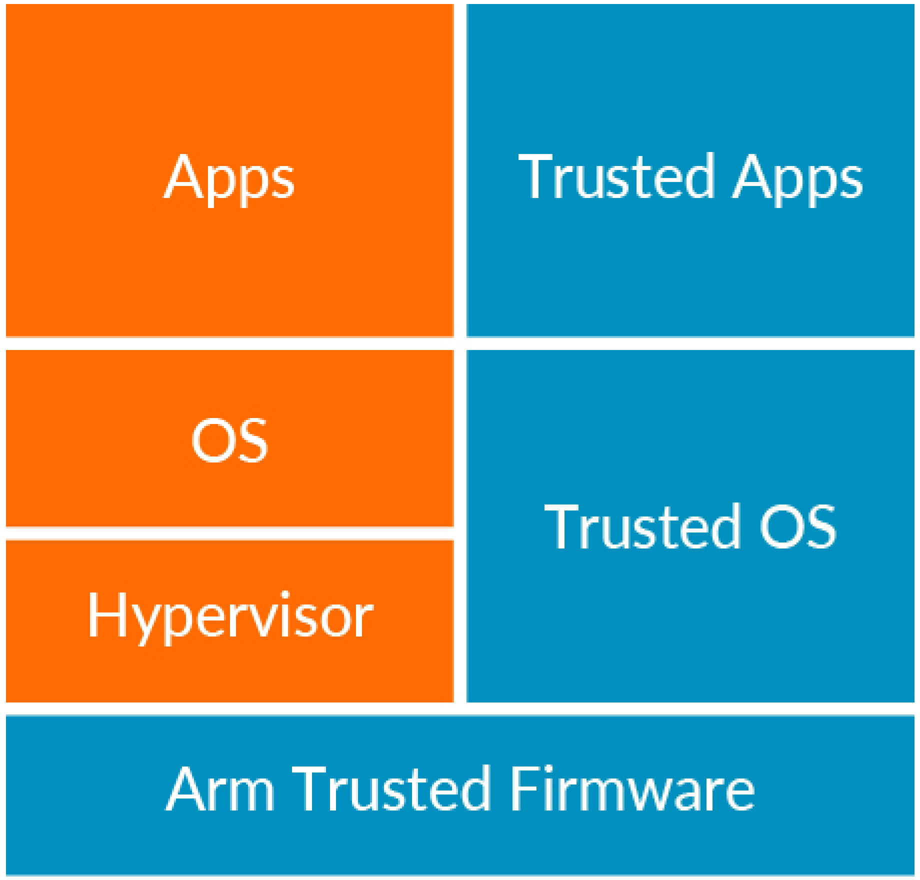
Related Work

- Security Design and Protections
  - Hardware Solution: Intel SGX, ARM TrustZone
    - Hardware isolation for an enclave
    - New instructions to establish, protect
    - Call gate to enter
  - Remote attestation
    - Processor manufacturer is the root of the trust
  - Prime+Probe Attack: March 2017
    - Target to DRAM



Related Work

- ARM TrustZone
  - Based on ARM Cortex-A and Cortex-M series
    - Privileged instructions to call entry/exit
    - Light-weighted comparing with other protection
  - ARM helps in creating Trusted Execution Environments (TEE)
  - Cache Problems
    - ARM Cortex-A series
    - ARM Cortex-M series (ARMv8-M)



Related Work

- Previous Defense Strategy against Side-Channels
  - LLC-level Protection (memory access control)
  - Cache enclaves (Trusted vs. Untrusted)
  - Scheduler-based solutions
  - Others
- Cache Flush against Side-Channels
  - Benefits: easy to implement, ensure safety
  - Problems: high overhead, not adaptive to every situation

Related Work

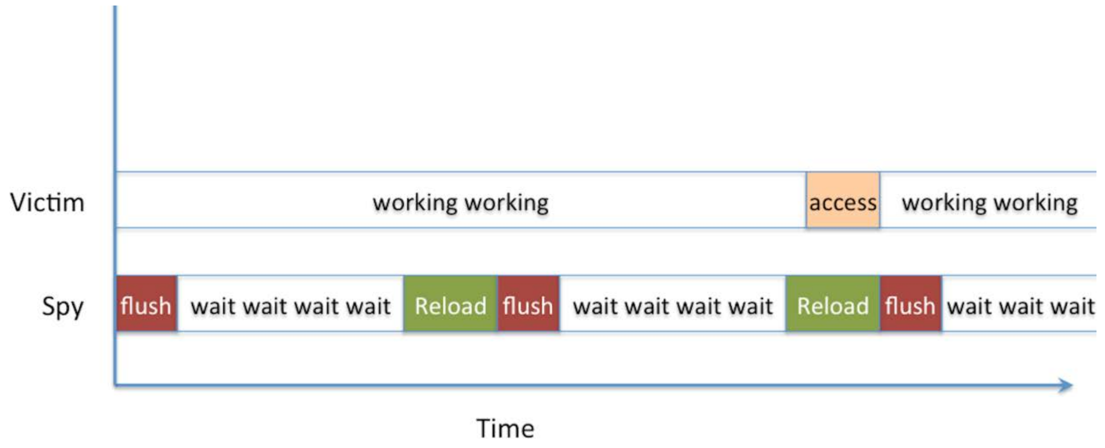
- Recent Research on ARM and TrustZone
  - TrustZone-based defense;
  - Performance measurement without security concerns;
  - Keystone Defense Framework;

Overview and Background

- Overview
- This Dissertation is a re-organization of some published work during years 2018-2020, in which most experimental work had been in 2016-2018.
- To summary, we have two major research projects:
  - We measure the cost and effectiveness of ARM TrustZone entry/exit and the cost of cache operations, such as Flush operations; (Published on July 2020, WISA conference)
  - Based on the measurements and experimental results, we design and implement adaptive defense framework. We also test the defense and have both experimental and theoretical analysis; (Published on March 2018, EAI journal of security and privacy)
  - We also have other experimental results and discussions to help and support our analysis. (Book Chapter in 2020, Eliva Press)

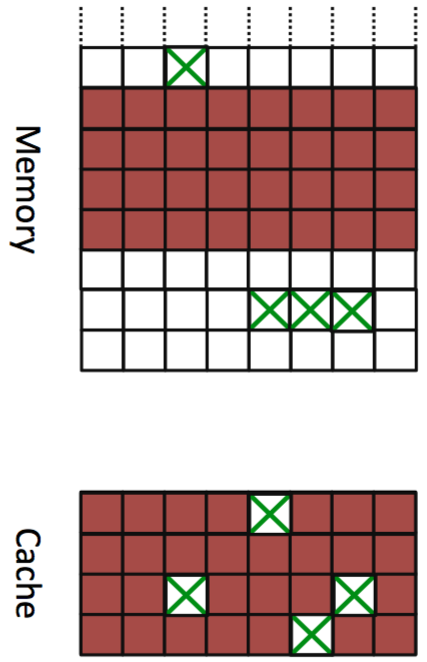
Overview and Background

- Cache Threats: Time-Based Attack
- Flush+Reload Attack



Overview and Background

- Prime+Probe Attack



- Attacker chooses a cache-sized memory buffer
- Attacker accesses all the lines in the buffer, filling the cache with its data
- Victim executes, evicting some of the attackers lines from the cache
- Attacker measures the time to access the buffer
  – Accesses to cached lines is faster than to evicted lines

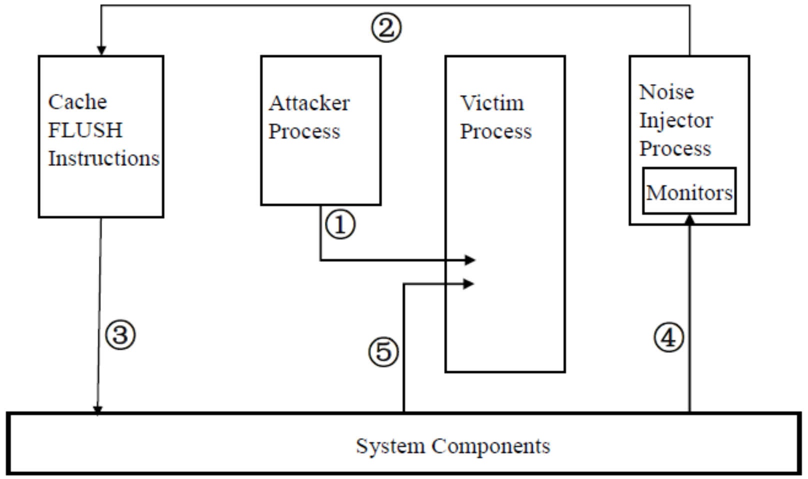
4

Overview and Background

- Threat Model and System Assumptions
  - Side-channel attackers and other cache-based attackers are not based on compromised OS.
  - We assume that the memory is not shared between victim process and the attacker (Covert Channel)
  - On system side, we assume that the operating system components in TrustZone is not compromised.
  - we also assume the system is having a control part, i.e. handler to inject interference into possible side-channel.
  - We also assume that the attacker has sufficient privilege to access the memory access time. This is not the case in real life, but it shows the worst case for the users to be attacked.

Overview and Background

- Step 1:An attacker utilizes the cache to launch side-channel attack, i.e. Flush+Reload attack;
- Step 2: the noise injector sends cache FLUSH request, and connect with system components;
- Step 3: Cache FLUSH instructions;
- Step 4: Monitors collecting performance and other data;
- Step 5: Cache FLUSH makes impact on victim’s listening.



Cache-Based Security Threats and Attack

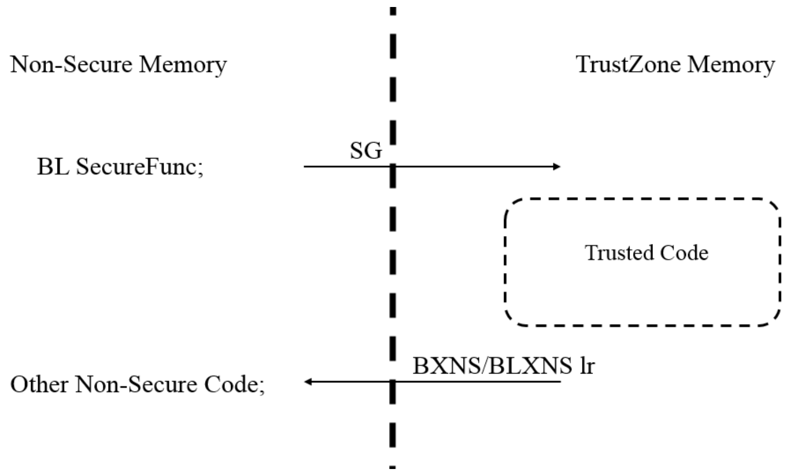
- Overview
- Users’ memory access are not protected by TrustZone – Covert Channel (Sharing resources)
- TrustZone Entry/Exit without Flushing cache – Side-Channel (Malicious collecting access time)
  - Flush+Reload Attack
  - Prime+Probe Attack
- Malicious eavesdropping

Cache-Based Security Threats and Attack

- Side-Channel Attack Experiment
- Flush+Reload Attack
  - step 0: attacker maps shared library → shared memory, shared in cache
  - step 1: attacker flushes the shared line
  - step 2: victim loads data while performing encryption
  - step 3: attacker reloads data → fast access if the victim loaded the line
- Prime+Probe Attack
  - step 0: attacker fills the cache (prime)
  - step 1: victim evicts cache lines while performing encryption
  - step 2: attacker probes data to determine if the set was accessed

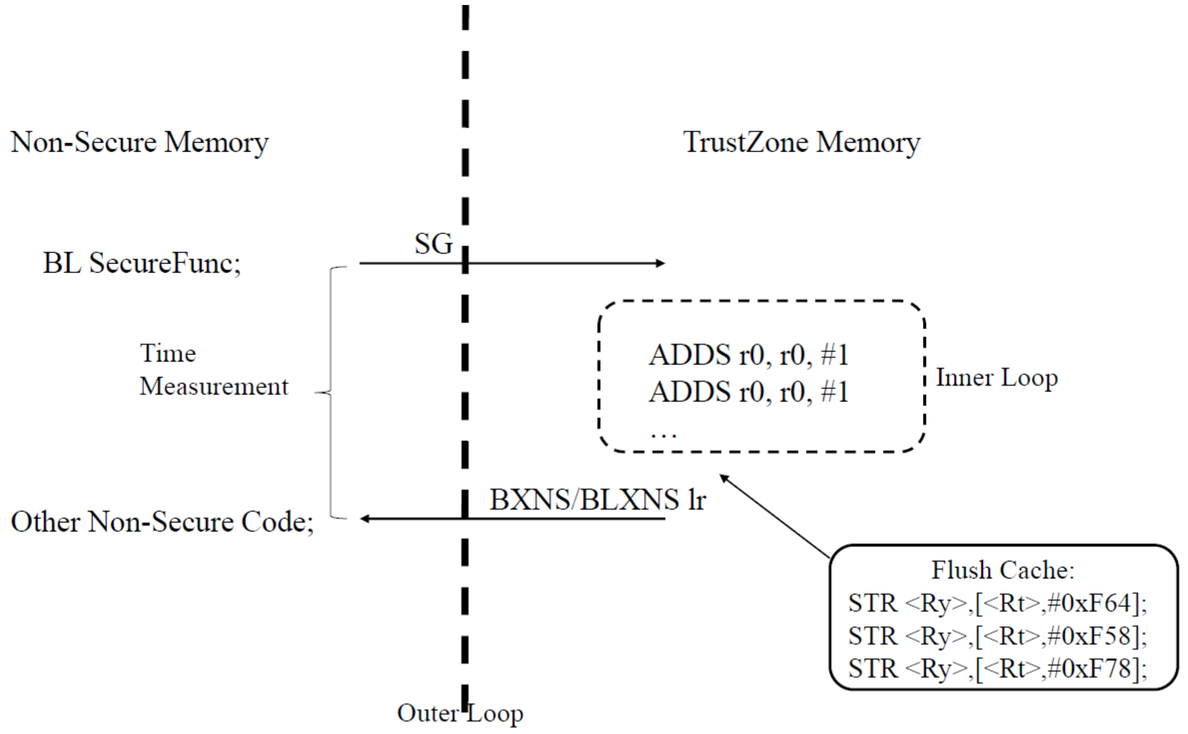
Design and Implementations

- TrustZone-Related Instructions
  - ARMv8-A
    - Test Environment: ARM Juno r1 Board, with A57 and A53 chips; QEMU as testing benchmark.
  - ARMv8-M
    - Test Environment: ARM Development Kits with Cortex-M4



Design and Implementations

- Experiments on TrustZone Instructions
- ARMv8-M
- Our experiments on ARMv8-M are using ARM Versatile V2M-MPS2 Motherboard with an ARM Cortex-M4 chip. It offers 8Mb of single cycle SRAM, and 16Mb of PSRAM. It supports the application of different ARM Cortex-M classes, from Cortex-M0, to M3, M4, and M7.



Experimental Results

- Experiments on TrustZone Instructions
- ARMv8-A
  - We use Ubuntu 16.10 as the normal world OS, with 26 processes running on background, including the workload we use for testing. We count the smc-related instructions that belongs to TrustZone-related operations, and analyze the attributions of them.

Type	Percentage
Non-secure to Secure Test R/W	2.87%
Secure to Non-secure Test R/W	2.91%
Others (Access from Background)	0.01%

Experimental Results

- Experiments on TrustZone Instructions
- Cortex-A
- Using QEMU as shown above.

Operation	Direction	Cost on Average (Clock Cycles)
P0_nonsecure_check_register_access	Non-Secure to Secure	1950
P0_secure_check_register_access	Secure to Non-Secure	2200

Experimental Results

- Experimental results on Cortex-M series chips

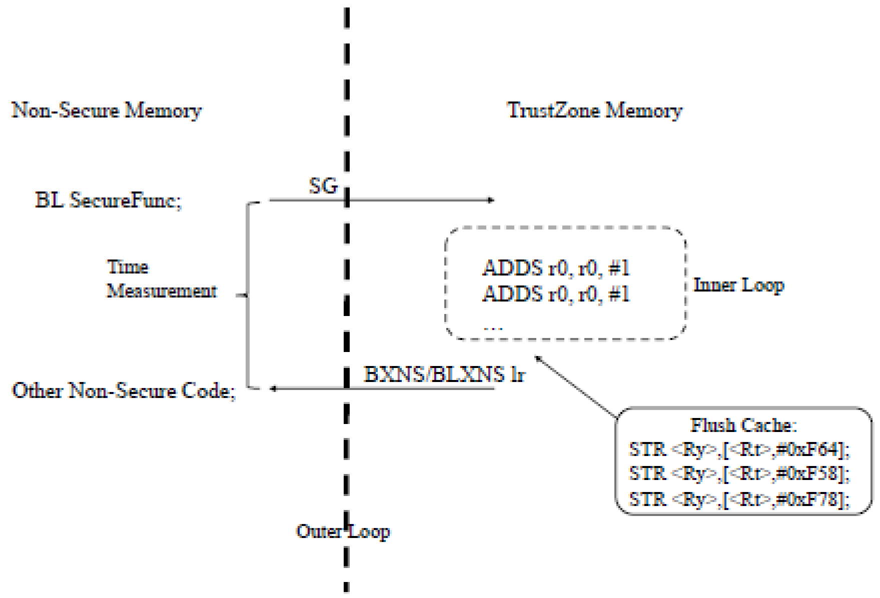
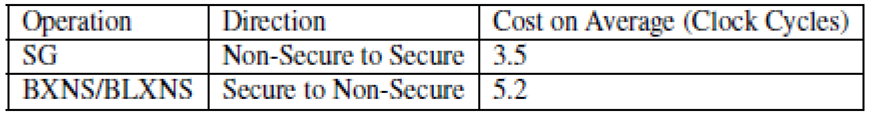


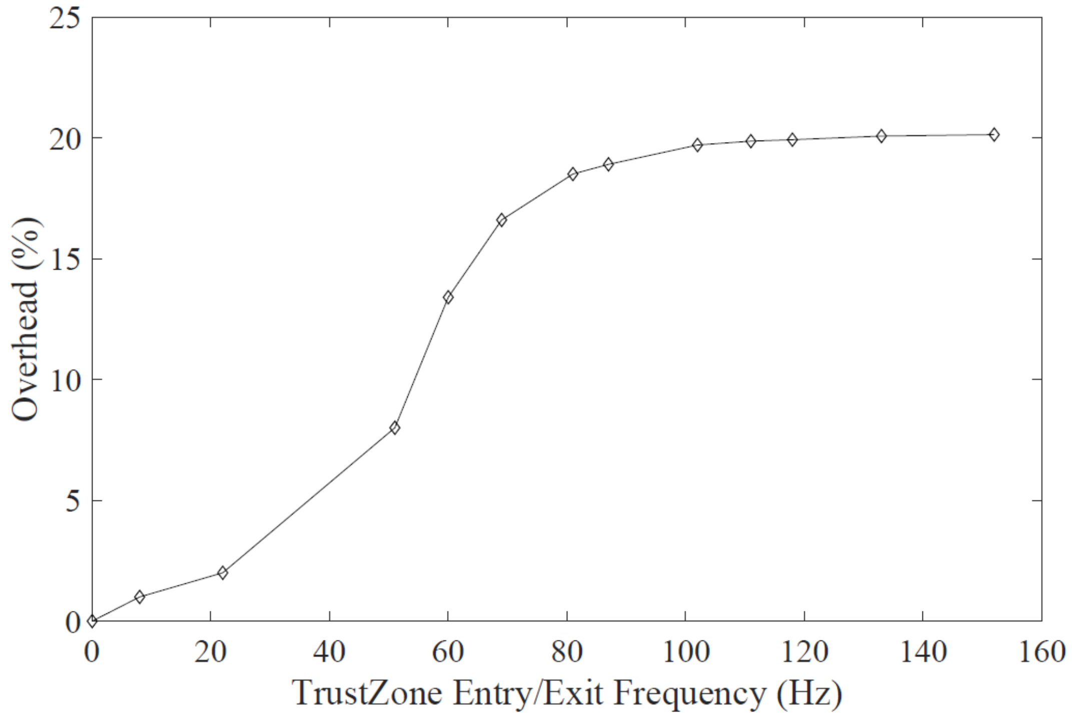
Table 3.3: TrustZone-Related Instructions Cost on ARMv8-M



Operation	Direction	Cost on Average (Clock Cycles)
SG	Non-Secure to Secure	3.5
BXNS/BLXNS	Secure to Non-Secure	5.2

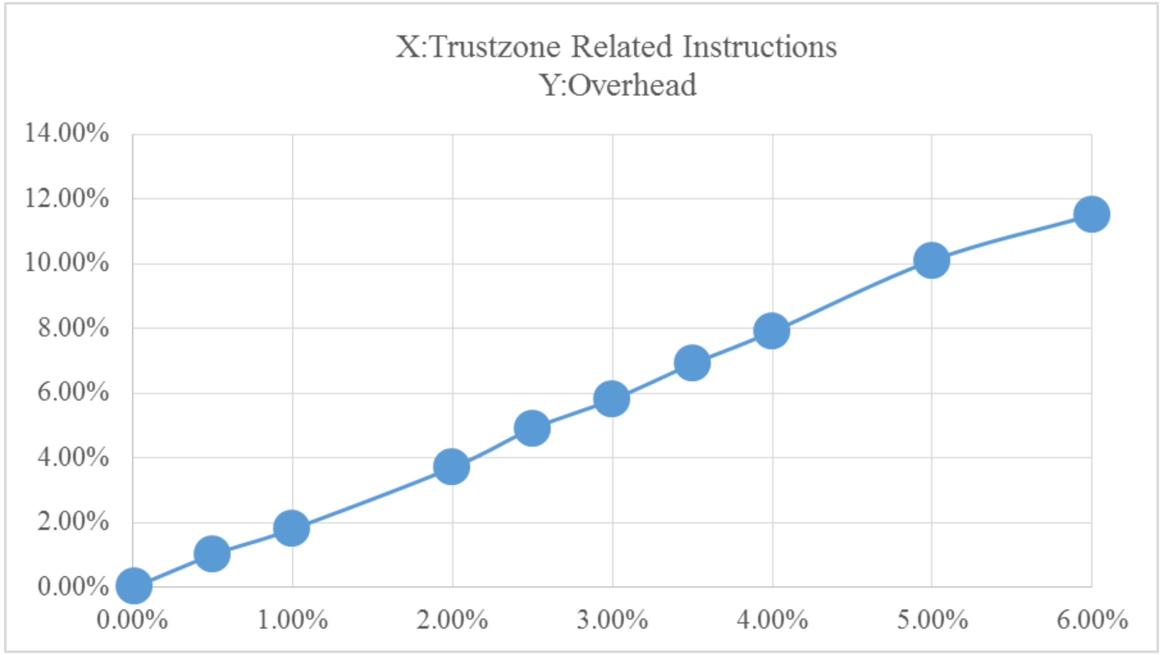
Experimental Results

- Experiments on TrustZone Instructions
- ARMv8-A
- With every smc-related instruction, we operate Flush on cache.



Experimental Results

- Experiments on TrustZone Instructions
- ARMv8-A
- We change the overall percentage of smc instructions and see the overhead difference.



Evaluation

- On the cost-effectiveness balance of defending by Flush operations
  - Flush operations are necessary, but they cost much;
  - We can never wipe out the risk, but can cut down bandwidth;
  - Adaptive strategy is used to keep the balance of performance and effectiveness;
  - On Cortex-A series chips, usually adaptive strategy can cost less than 10% overhead;
  - Even better on ARMv8-M chips.

Evaluation

- On TrustZone related instructions
  - Most of the apps and users are not ‘making use of’ TrustZone features;
  - On IoT devices, TrustZone is not costing much resources;
  - It is possible to move some of the hardware/software security design into TrustZone surface;
  - Cortex-M series chips perform better than Cortex-A series chips.
  - On Cortex-A series chips or x86 chips, cache flush operations are just some instructions with privileges. However, the case are different on ARMv8-M. The allocation of a memory address to a cache address is defined by the designers of the applications.
  - Because of the special structure of ARMv8-M, the cache Flush operations are sets of DSB (Data Synchronization Barrier) operations, with address-related instructions.

Defense Design and Implementations Based on ARM Platform

- Defense Strategy
  - Hardware Defense
  - Privilege level designs
    - Not everybody can flush the cache or do cache related measurement, e.g., ARM
  - When the cache should be flushed?
    - Whenever there is a possibility of information leak - what about the performance?
    - During process context switching?
    - During processor mode switching?
  - Experiments – Performance and Bandwidth

Defense Design and Implementations Based on ARM Platform

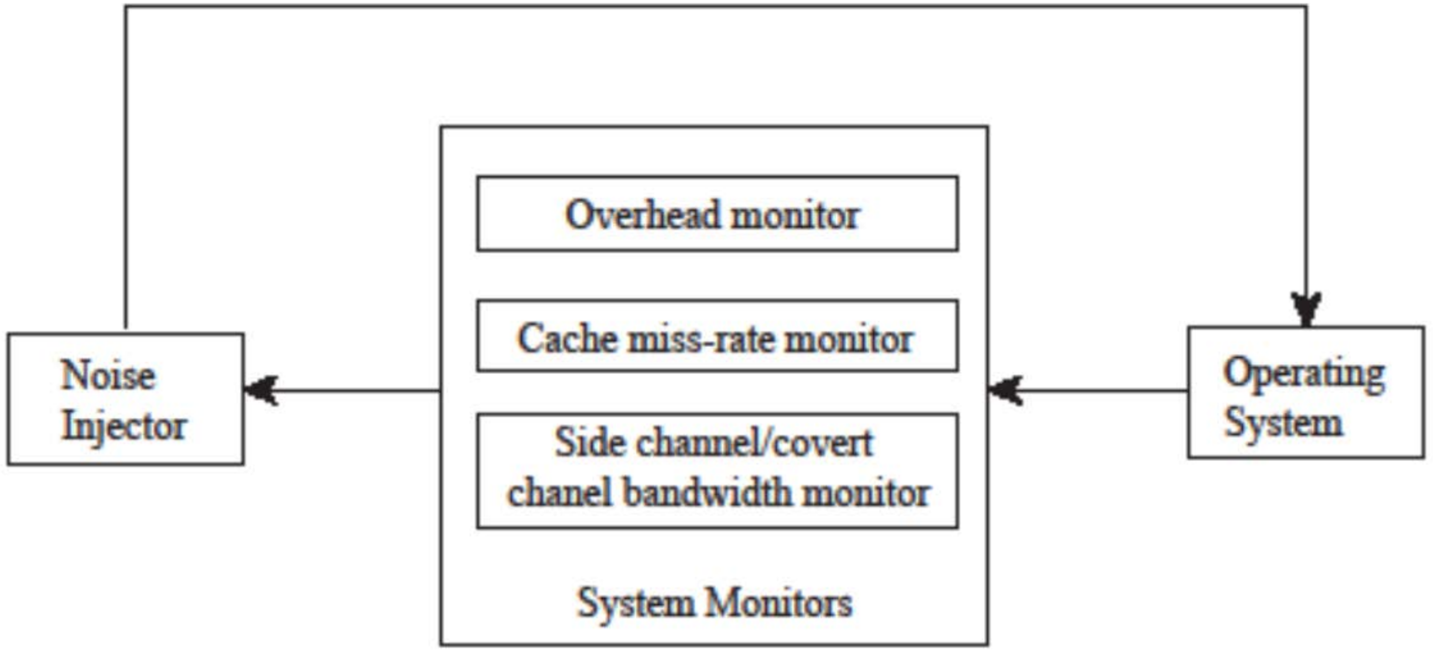
- Defense Strategy
  - Software Solution
  - Design on ARMv8
    - TrustZone entry/exit
  - Noise injections to the channels
    - Also decrease the performance but under control.
  - Experiments – Noise Injection and Bandwidth

Defense Design and Implementations Based on ARM Platform

- Defense on ARMv8-M Challenges and limitations
  - TrustZone for IoT
  - Efficiency in entry/exit
  - Defense based on TrustZone
  - Experiment – Cost of TrustZone operations

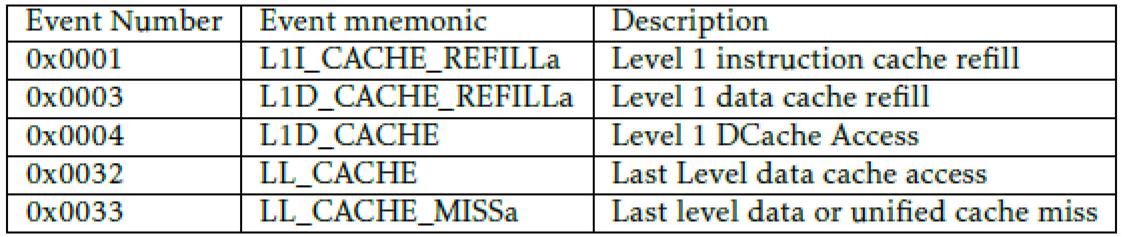
Design and Implementations

- Adaptive Flush Operations on ARMv8-A
- On ARMv8-A tests, we change cache flush frequency when the system is on high frequency of TZ operations.
- Must maintain good performance (low overhead) while keeping lower bandwidth



Design and Implementations

- Monitors Setup
  - Time measurement: special registers and instructions;
  - Overhead: TEE and benchmark;
  - Cache miss rate: Special Registers.



Event Number	Event mnemonic	Description
0x0001	L1I_CACHE_REFILLa	Level 1 instruction cache refill
0x0003	L1D_CACHE_REFILLa	Level 1 data cache refill
0x0004	L1D_CACHE	Level 1 DCache Access
0x0032	LL_CACHE	Last Level data cache access
0x0033	LL_CACHE_MISSa	Last level data or unified cache miss

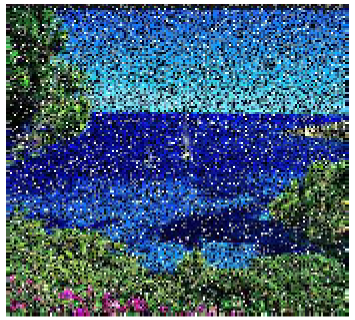
Design and Implementations

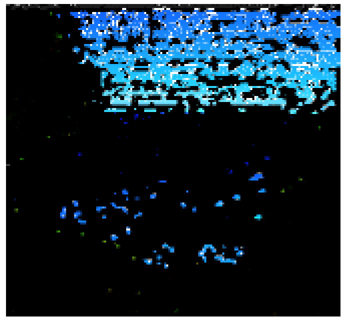
- Other Implementations
  - Error Correction;
  - Flush Operation;
  - TrustZone entry/exit.

Experimental Results

- Experiments on Side-channel
- Flush+Reload Attack on *libjpg*;
- Using CRC to try recovering the original file;
- Calculating Bandwidth and performance overhead difference by Flushing cache.

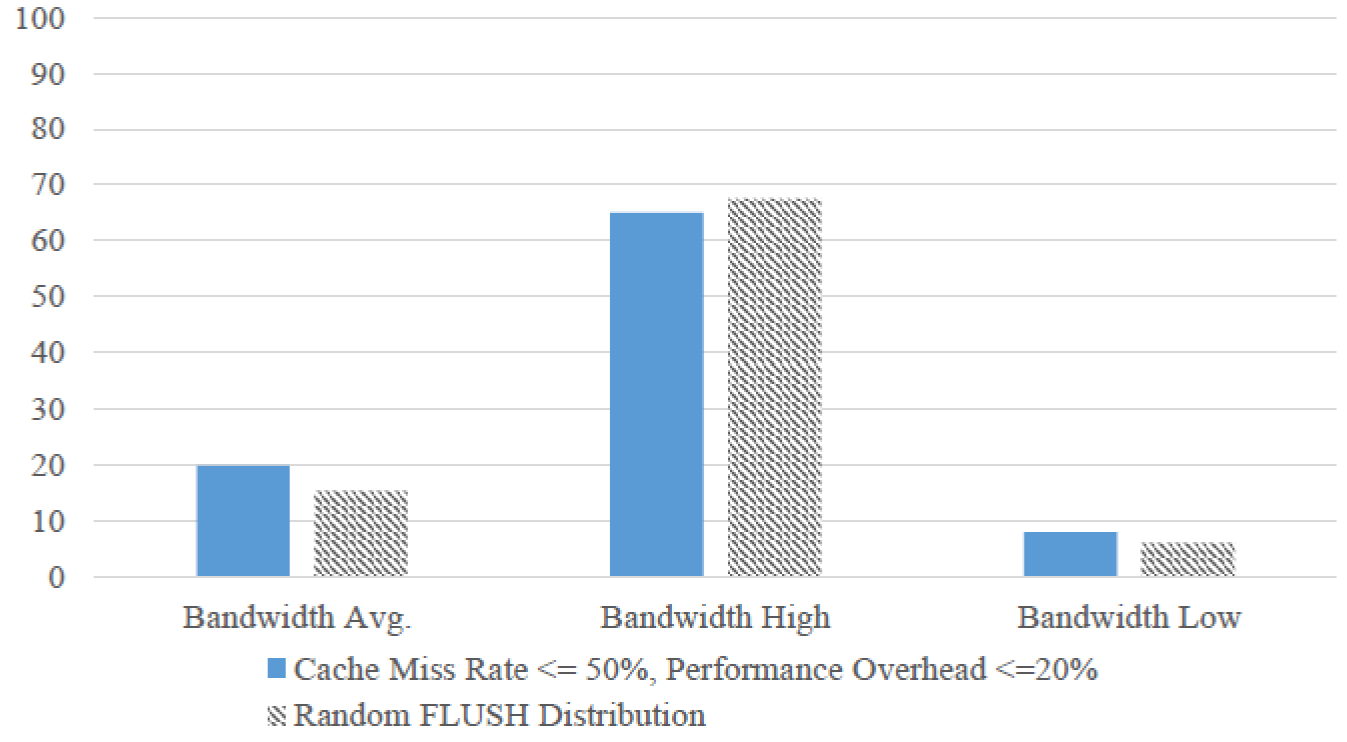
Original


Noisy


ECC operated

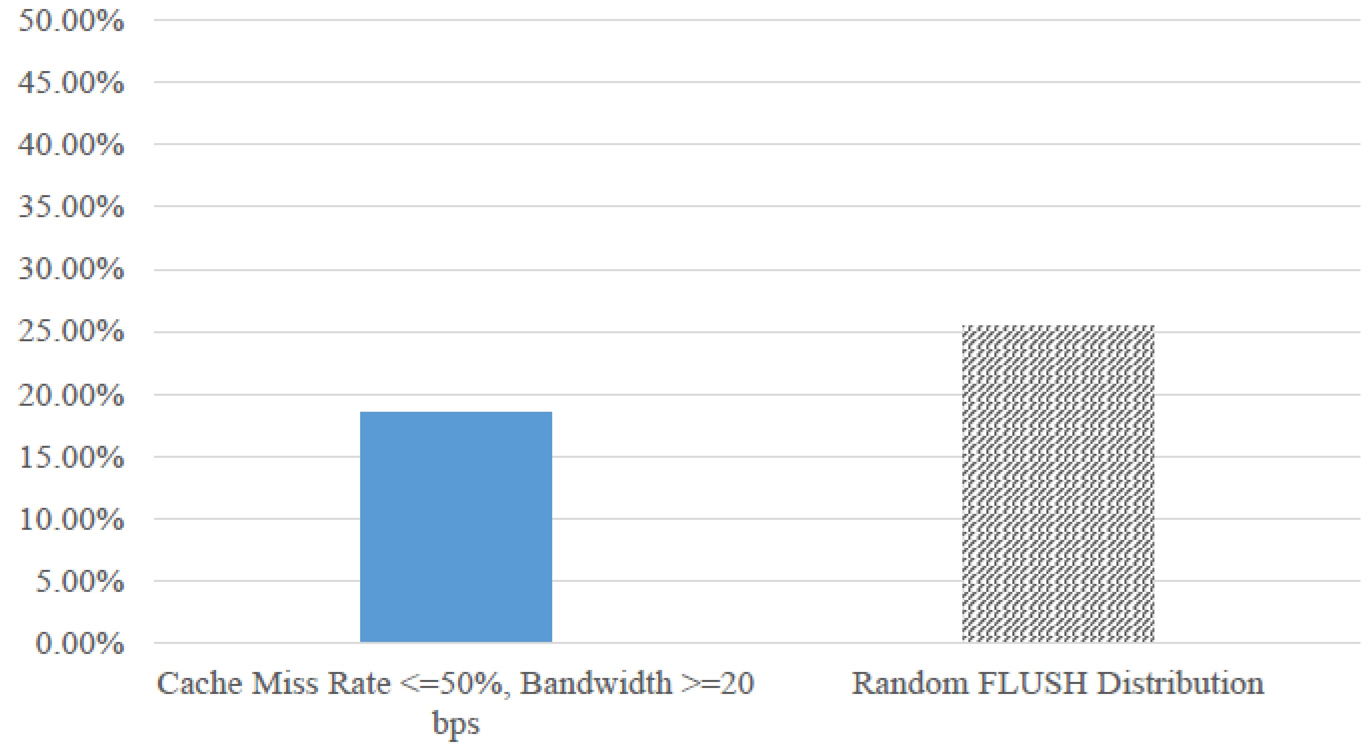
Experimental Results

- Test: Cache miss rate and overhead balance.



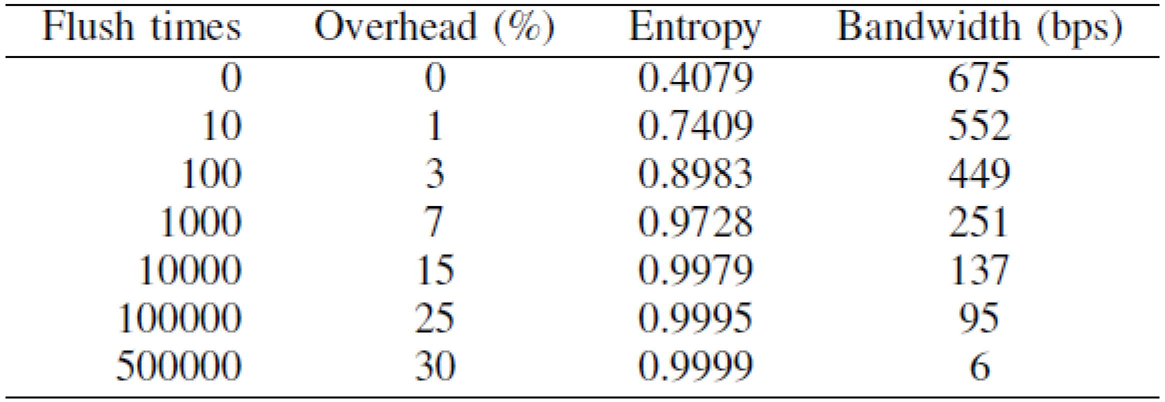
Experimental Results

- Test: Cache miss rate and Flush frequency balance.



Evaluation

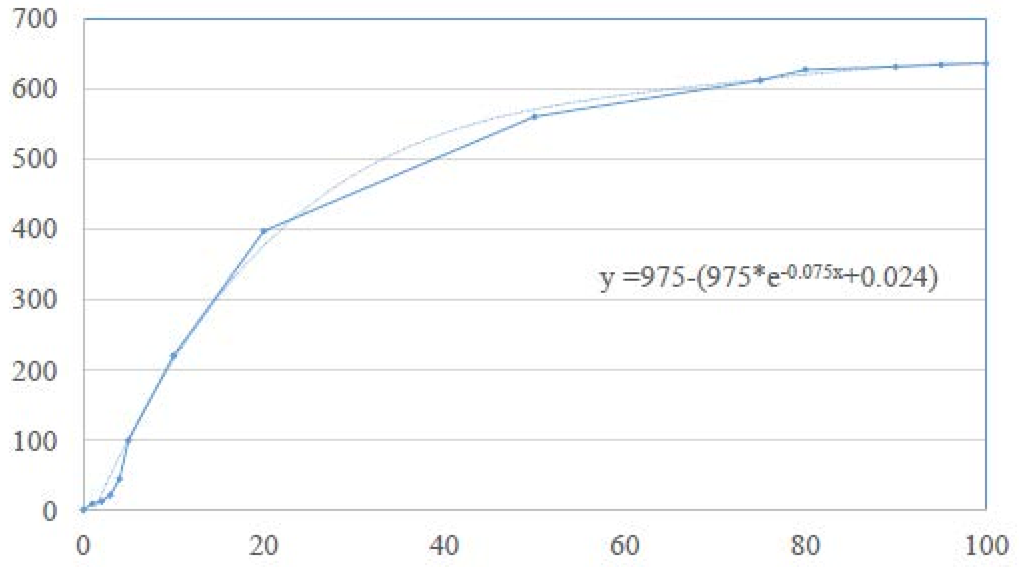
- Dealing with covert channel is also a problem of balancing overhead and effectiveness.
- From our test results, it is almost impossible for some malicious users to launch covert channels with high entropy and very low bandwidth, which means that they cannot retrieve useful information, or the time consuming is not acceptable.



Flush times	Overhead (%)	Entropy	Bandwidth (bps)
0	0	0.4079	675
10	1	0.7409	552
100	3	0.8983	449
1000	7	0.9728	251
10000	15	0.9979	137
100000	25	0.9995	95
500000	30	0.9999	6

Evaluation

- In the experiments where we randomly insert flush operations to interfere with the side-channels, the time of injecting noise is randomly distributed. Also, the interval of each pair of operations is randomly distributed. Exponential distribution is usually used to describe the distribution of intervals of a set of statistically independent events.



Conclusions

- Cache-based attack are new focal point on security design, with risks of leaking information through side-channel and covert channels.
- Flushing cache is effective to cut down the risk, but with high performance overhead, and sometimes not affordable.
- On IoT devices, the performance of connecting with TrustZone can be better, which brings the possibility to making use of TrustZone.
- Adaptive strategy is still needed for the balance of the performance and the defense effectiveness.

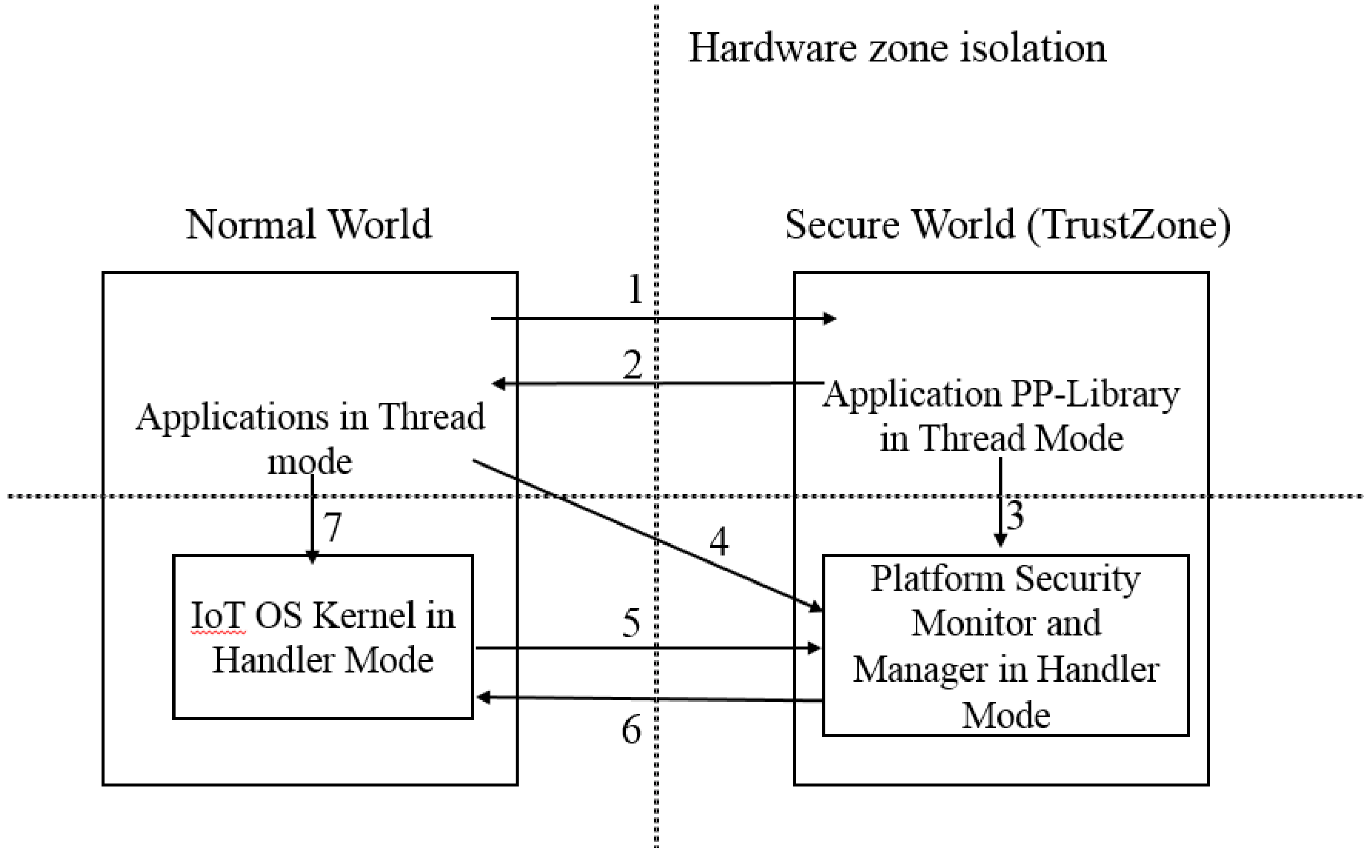
Future Work and Plan

- Implementations and Experiments
  - Design and implement a defense framework based on ARMv8-M.
  - Test the performance of defense framework using some benchmarks, and optimize the framework to good effectiveness and lower overhead.
  - Port defense framework to new ARMv8-M boards: M23 and M33 series chips.

Future Work and Plan

- Theory Work
  - Study adaptive control method in theory to match the experimental results and predict the optimal solution of best adaptive control in defense.
  - Investigate entropy theory based on experimental results, predictions and related theory.
  - Discuss performance of implemented defense framework in theory and try to have theoretical conclusion on defense against cache-based attack.

Future Work and Plan


Hardware zone isolation
Normal World
Secure World (TrustZone)
Applications in Thread mode
IoT OS Kernel in Handler Mode
Application PP-Library in Thread Mode
Platform Security Monitor and Manager in Handler Mode
1
2
3
4
5
6
7

Publications

- Liu N, Zang W, Chen S, Yu M, Sandhu R: Adaptive Noise Injection against Side-Channel Attacks on ARM Platform, EAI Endorsed Transactions on Security and Safety, 2019;
- Liu N, Zang W, Yu M, Sandhu R: On the Cost-Effectiveness of TrustZone Defense on ARM Platform, The 21st World Conference on Information and Security Applications (WISA), 2020, Maison Glad, Jeju, Korea
- Liu N, Yu M, Sandhu R: Cache Security on ARM: Side-channel Attack and Defense: Introduction to Side-channel on ARM Platform, Book Chapter by Eliva Press, ISBN: 978-1952751264, 2020;
- Liu N, Zang W, Yu M, Sandhu R: Cost and Performance of TrustZone Defense against Cache Threats on ARM Platform, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA, Invited), 2020.

Q&A Time

Thank you so much for your questions!