arm

Birds of a Feather Session

Security

Arm Research Summit

September 2019

Agenda

90 minute slot, try to keep relatively informal:

- Overview of Security Research at Arm by Stuart Biles, Fellow and Senior Director Research Technology
- Three focused lightning talks by Arm Research Engineers
  - **Jaekyu Lee**: Spectre mitigations
  - **Prakash Ramrakhyani**: Memory encryption and integrity protection
  - **Dominic Mulligan**: Privacy-preserving compute
- Unstructured discussion time

arm

Arm Research

Security Research Overview

Security | Privacy | Correctness

Hugo Vincent

4th September 2019

Security Research Group

Vision: Technology worthy of the trust we place in it.

Mission: Systematically remove excuses for untrustworthiness.

- Develop new security and robustness technologies, and accelerate adoption by eliminating performance costs and other obstacles.
- Reduce what needs to be trusted: minimize the trusted computing base (TCB), and verify what remains.
- Take a principled and quantitative approach to security through the use of formal methods.
- Progress the application of privacy-enhancing technologies to information processing systems — demonstrate that business requirements can be addressed while respecting users and their data.

Security Research Group

What we don't do:

- Vulnerability research (red team).
- Reactive security (anti-virus etc) — we focus on proactive approaches.
- Developing point fixes for vulnerabilities — we want broadly applicable class fixes that generalize over specifics (economy of mechanism).
- Niche applications with specific requirements — we focus on trying to make general software secure, because that's where the bulk of vulnerabilities reside.

Current Security Research Projects

- Current research focuses on securing **distributed and edge computing**, memory safety, **enclave / TEE** technology, and enabling **respectful computing** on sensitive data.
- Four main projects — ECATS, Veracruz, Armour, Icecap.
- Other security activities around *Arm Research*: µarchitectural security (transient execution side channels etc), cryptographically protecting DRAM, attestation architecture for PSA

- Other activities
  - Accelerating post-quantum crypto
  - Privacy-preserving middleboxes using TEEs
  - TrustZone-based attestation of VMs and containers
  - Security standardization for IoT and microcontrollers

Memory (Un)safety

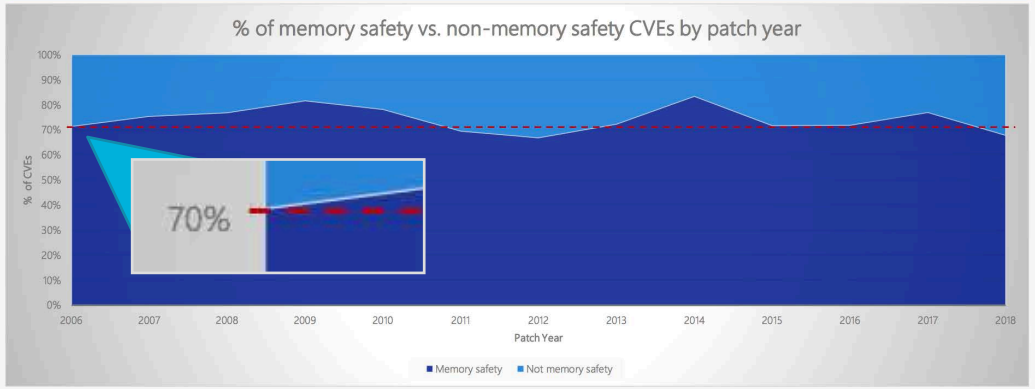
C & C++ memory safety is a mess

- Use-after-free / buffer-overflow / uninitialized memory
- > 50% of High/Critical security bugs in Chrome & Android
- Not only security vulnerabilities
  - crashes, data corruption, developer productivity
- AddressSanitizer (ASAN) is not enough
  - Hard to use in production
  - Not a security mitigation

Kostya Serebryany, Google
LLVM DevMtg (Oct 2018)

Memory safety issues remain dominant

We closely study the root cause trends of vulnerabilities & search for patterns



~70% of the vulnerabilities addressed through a security update each year continue to be memory safety issues

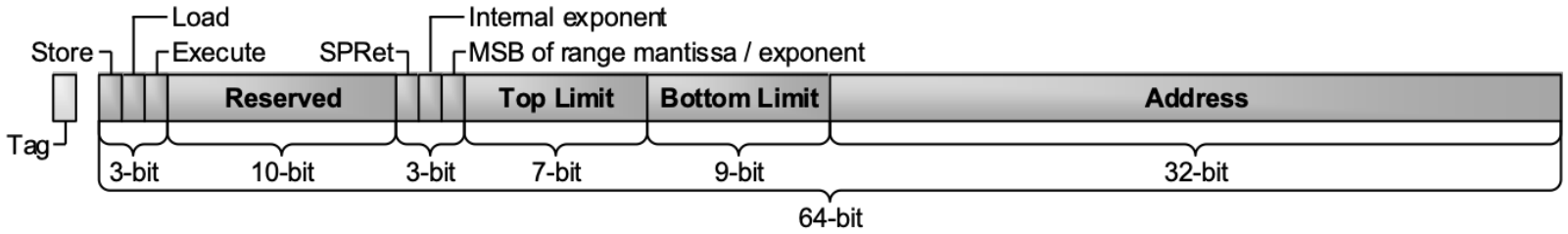
10

Matt Miller, Microsoft Security Response Center
Bluehat (Feb 2019)

A majority of exploits still stem from memory safety bugs (30+ years later)

ECATS: Capability-based security for Cortex-M

- Part of DARPA SSITH (System Security Integrated through Hardware) program, in collaboration with SRI and the University of Cambridge.
- Exploring a subset of SRI/Cambridge's CHERI capability-based memory security architecture on Cortex-M microcontrollers.

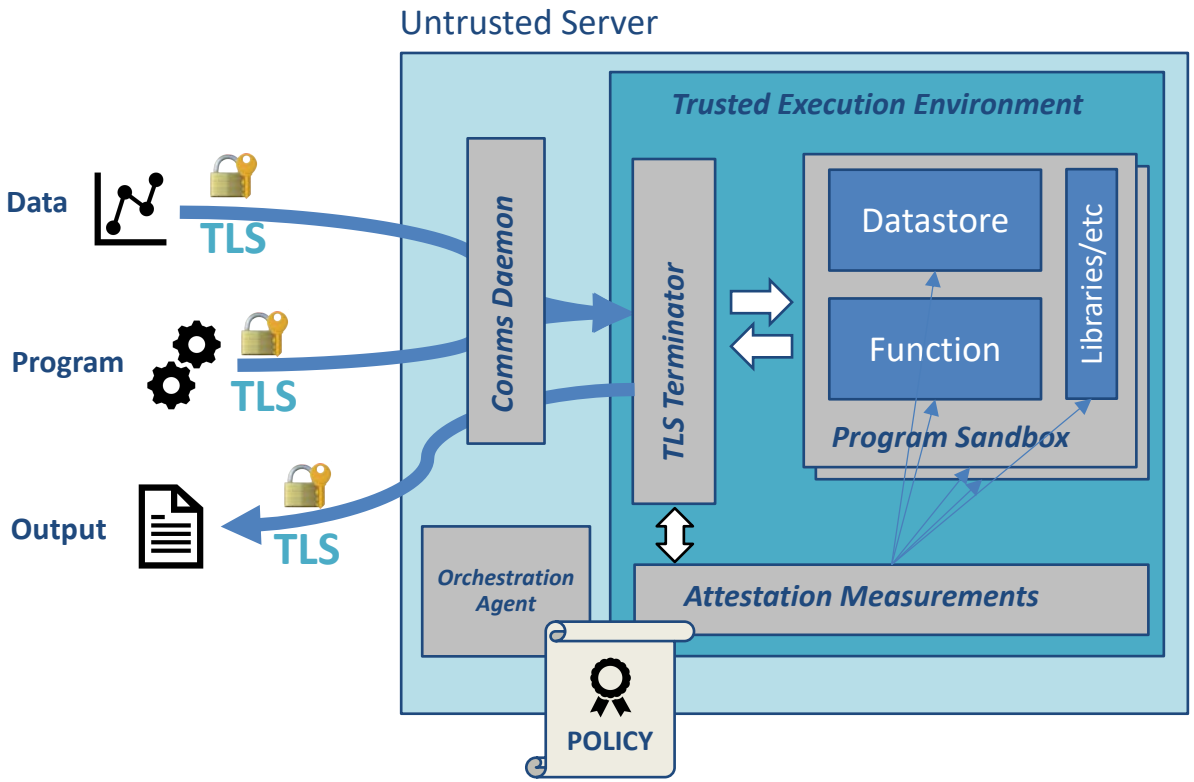


  - Capabilities are unforgeable tokens of authority (guaranteed by hardware memory tagging)
  - Pointers replaced (mostly by the compiler) with fat pointers ("capabilities") that contain bounds and permissions
  - Addresses majority of common C memory security vulnerabilities (buffer overflows etc)

1
1

Veracruz: private delegated computation

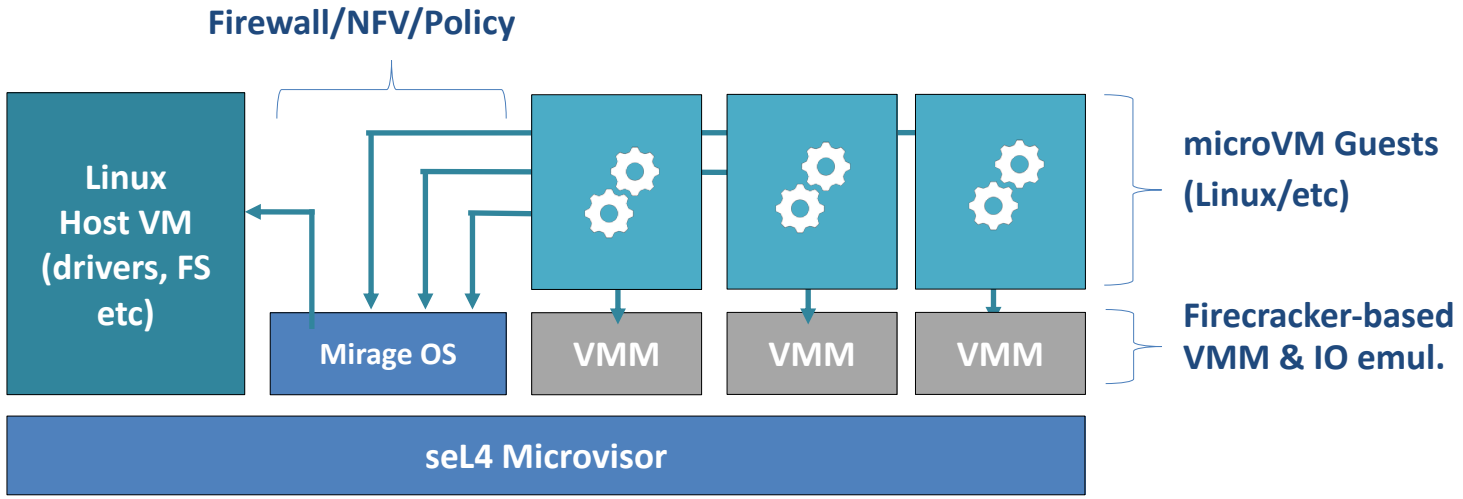
- **Goal:** enable sophisticated processing and analytics on sensitive or secret data
  - on third-party hardware (e.g. cloud or shared edge devices)
  - without revealing anything to the host.
- **Approach:**
  - encrypted programs and data delivered into a TEE-based enclave
  - attestation to owners before releasing keys for decryption
  - WebAssembly-based sandboxing to protect host and other enclave contents against malicious programs



1
2

Icecap: low-TCB virtualization for multi-tenant edge

- To support multi-tenancy on shared lightweight edge compute hardware, we need a lightweight mechanism (like Docker containers) but with stronger isolation (like virtual machines) and minimal, verified TCB. Icecap is our research platform for this.
- Based on a high assurance, formally verified micro-hypervisor (seL4), and using memory-safe / type-safe languages (Rust & OCaml) for system components.
- seL4 verification will guarantee confidentiality and integrity of microVMs
- Prototype running on Raspberry Pi 4 (AArch64) and QEMU.

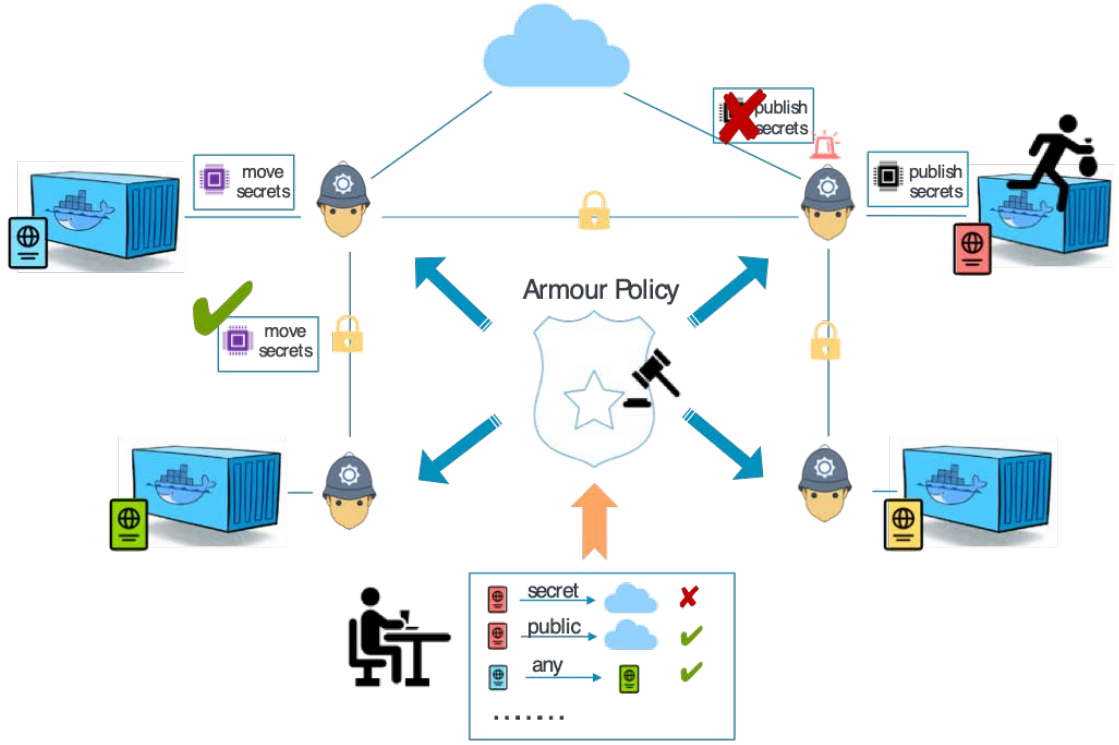


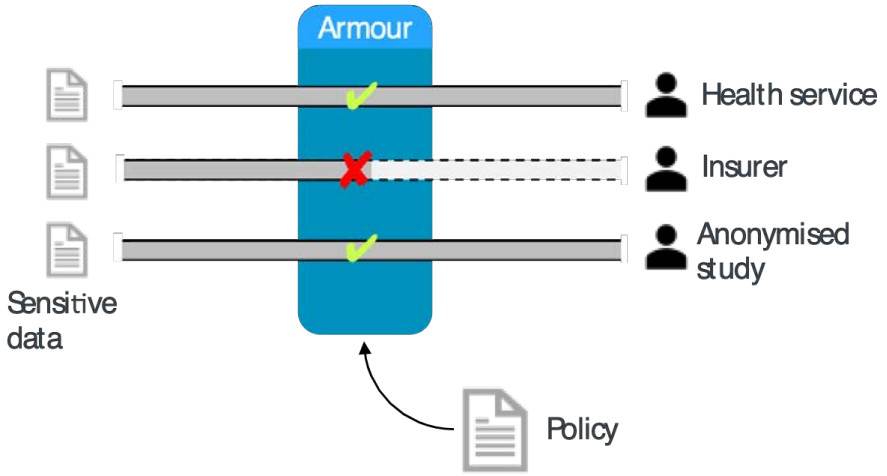
Next steps:

- Integration with Veracruz, Armour, Argus orchestration
- Firmware TPM in TrustZone
- PSA-based measured secure boot & attestation
- Testing & verification

Armour: security policies for edge compute

- Cloud/edge applications increasingly structured as collection of communicating microservices.
- Edge demands coexistence of multiple, distrustful entities on the same platform.
- **Goal:** enable declarative policy to ensure security properties of composition, and monitor and control information flow between components.

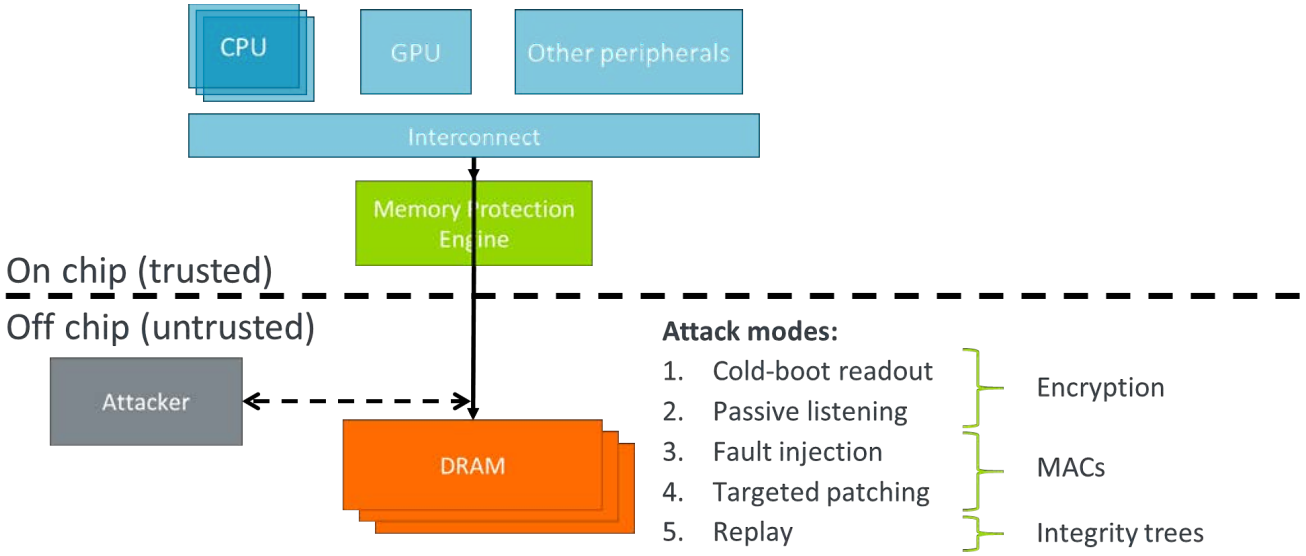


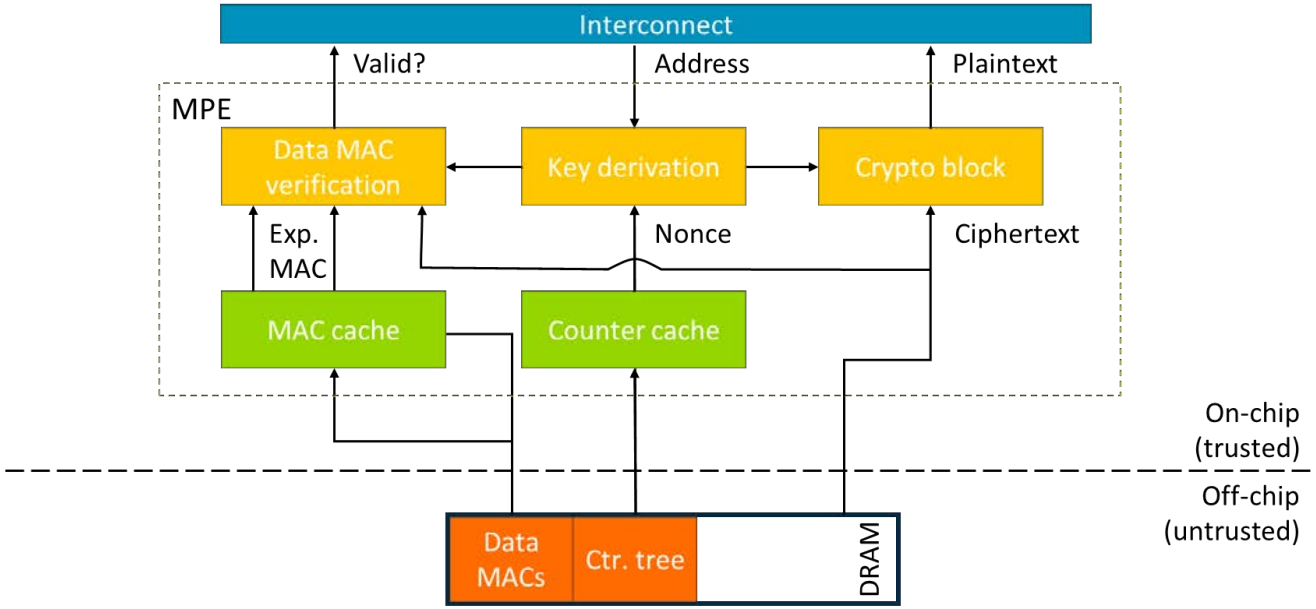


- Armour is a policy language and distributed enforcement architecture.
- Responsive to dynamic changes to workload and security environment.

Memory Protection Engine

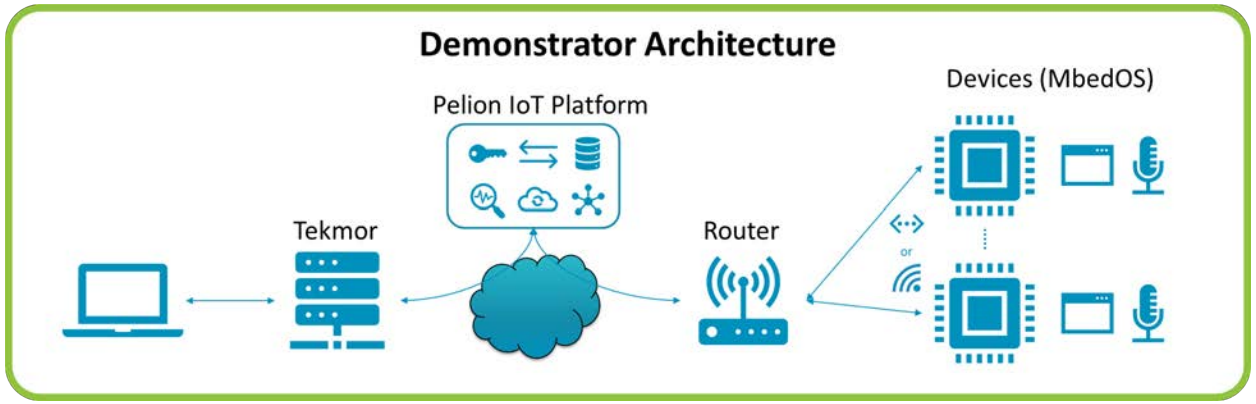
- Encrypts contents of select DRAM pages
- Combines nonce with address in key derivation to ensure temporal and spatial uniqueness
- Utilizes Bonsai-Merkle Trees of MACs to ensure immutability
  - The root of the tree stored on chip to prevent attackers with DRAM access from modifying MACs





Tekmor: Arm PSA Attestation and its applications

- Building on IETF's Entity Attestation Token (EAT), Arm is documenting its model and set of claims, generically applicable for constrained devices
- Claims include:
  - Unique ID for the device
  - ID of the authority
  - Boot seed
  - Software measurements
  - PSA partition ID of the caller
  - Security lifecycle state
  - Challenge



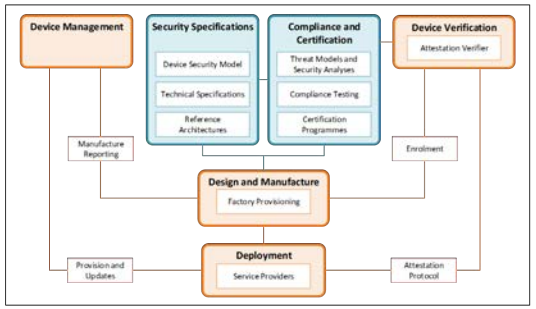
Attestation in PSA

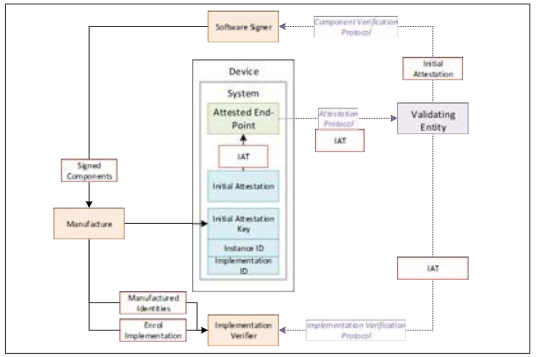
Remote Attestation
Prove (to a remote party) the identity and current state of the system.

Instance ID/Crypto
Each device has a secure and unique attestation key which is operated by PSA Root of Trust.

Isolation
Temporal and HW runtime isolation (e.g. TrustZone-M) protects secure services from less trusted code.

Secure Boot
Get the system into a known state
Establish a chain of trust from the HW to the application.

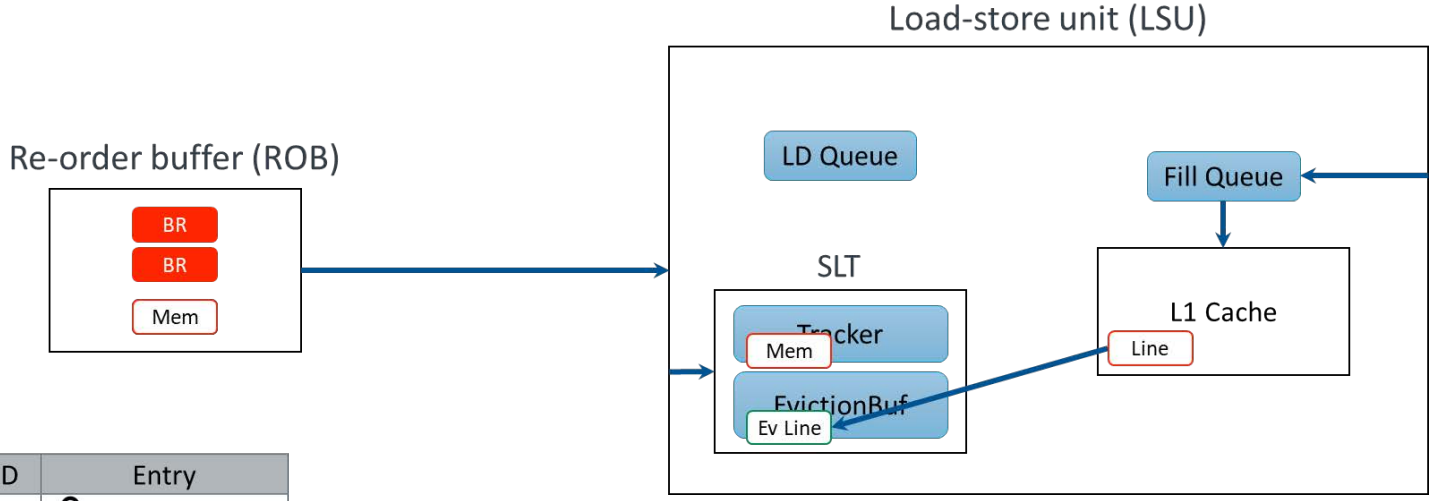


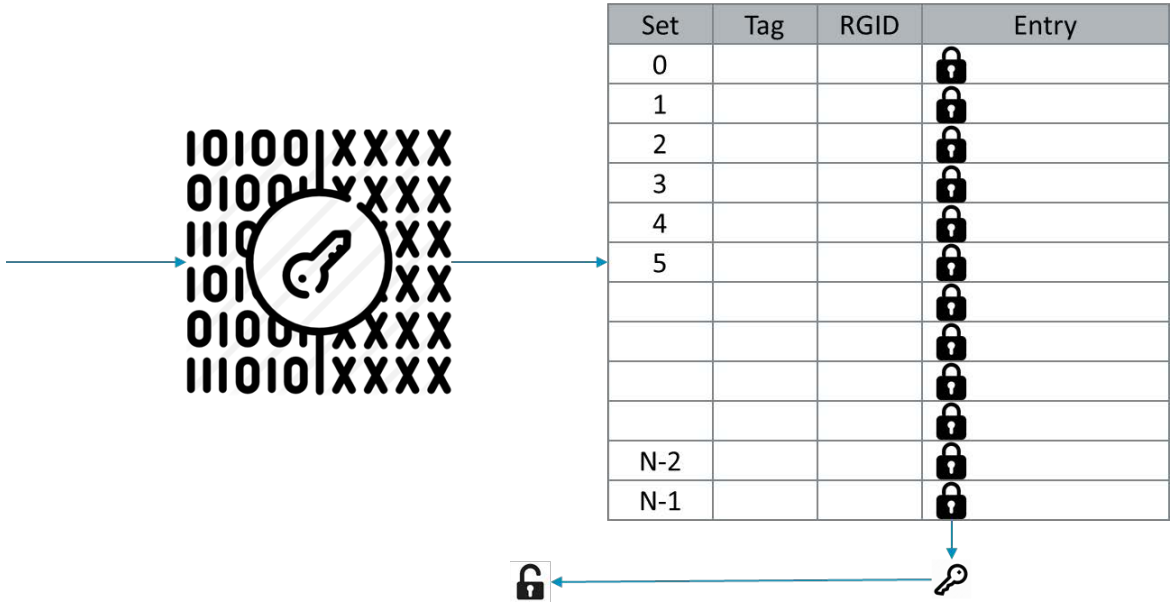


µArch Security: Mitigate various side-channel attacks

Spectre mitigation: Fill-and-restore cache

- Speculative Line Tracker (SLT)
- Track all transient cache fills
- keep the evicted line (by transient fill)
- Recover under mis-predicted branches





Branch predictor side-channel

- Branch target encryption
- Scramble branch predictor state to eliminate cross contain vulnerabilities
- Branch target in BTB entries encrypted per context

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

Arm Research Summit 2019 – Security Meetup Lightning Talk

Mitigating Branch Predictor Side-Channel using Encryption

Jaekyu Lee (Arm Research, Architecture)

September 16, 2019

Branch Predictor Side-Channel

- Branch predictor has been a popular target for various security attacks.
  - Attacks on cryptographic algorithms by Onur Aciiçmez (2006-2007).
  - Branch shadow attack (Security17).
  - BranchScope (Asplos18).
  - Spectre (S&P19).
  - Return address stack (RAS): Spectre returns (WOOT18), ret2spec (CCS18).

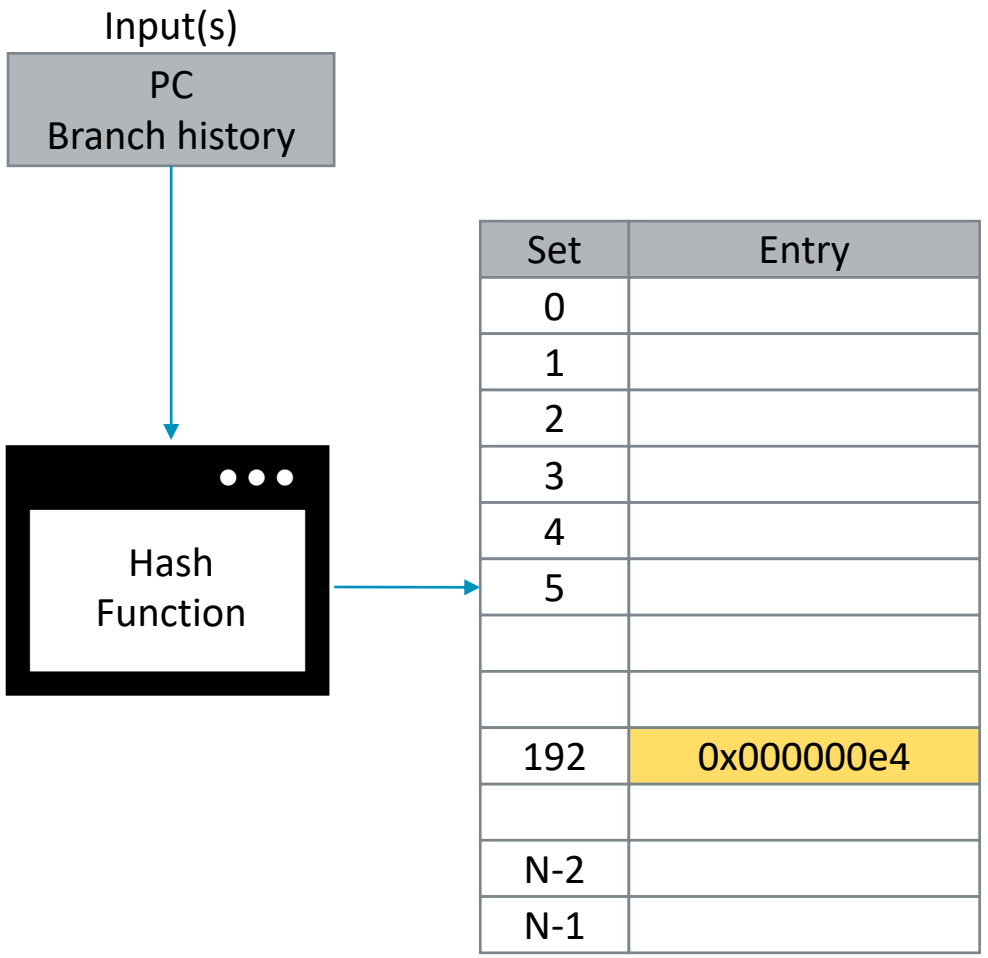
- **Problem statement**: Branch predictor has been exploited by the adversaries using **branch collision** due to the **sharing** between different processes and threads in the same physical core.

Branch Collision

- Target set-associative structures.
- Hash function for the set index.
  - (Folded) branch history ⊕ PC.
  - Partial index matching: aliasing.

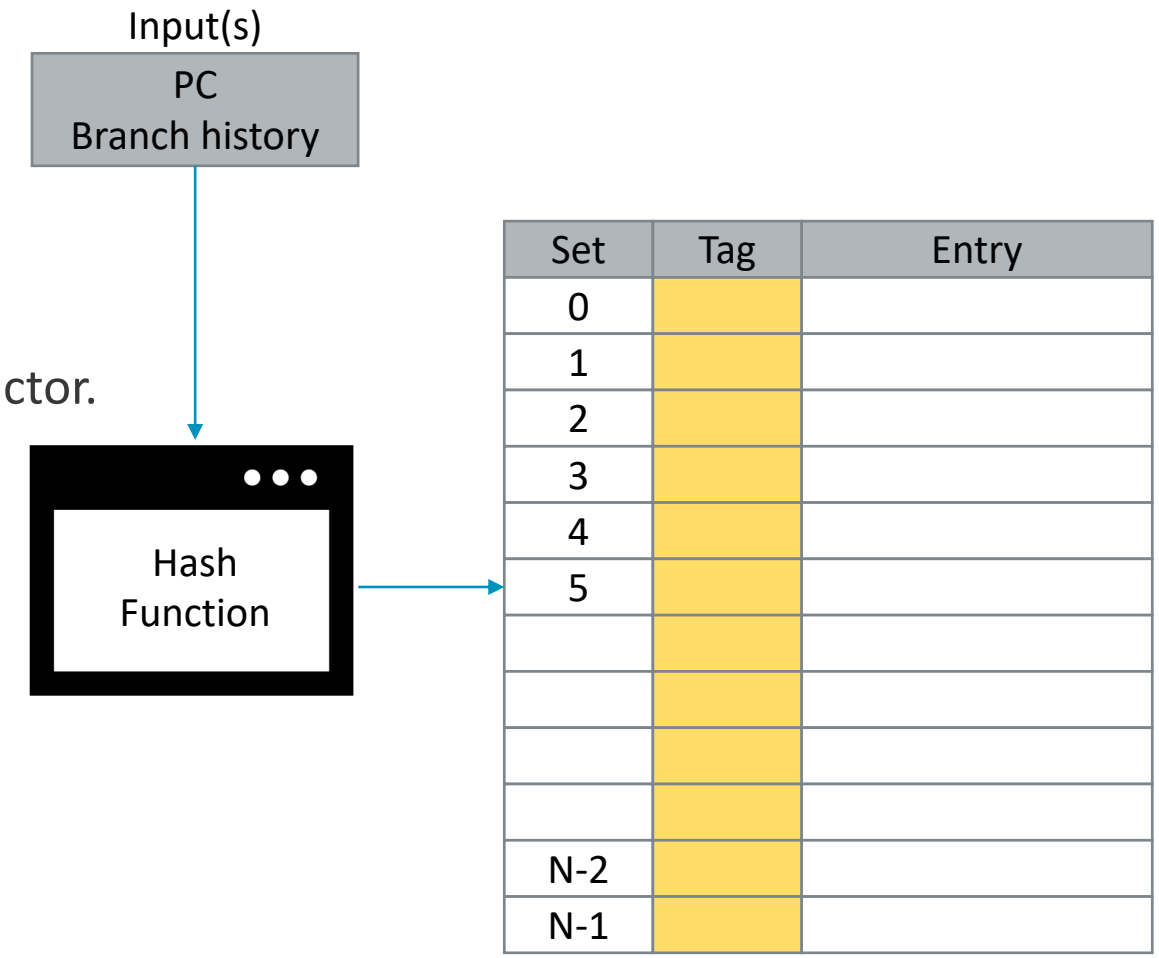
ASID	PC	Target	Set
0	0x800000c0	0x800000d8	192
1	0xC00000c0	0xC00000e4	192

- How to avoid malicious collisions?
  - Protect indexing (access).
  - Protect contents.



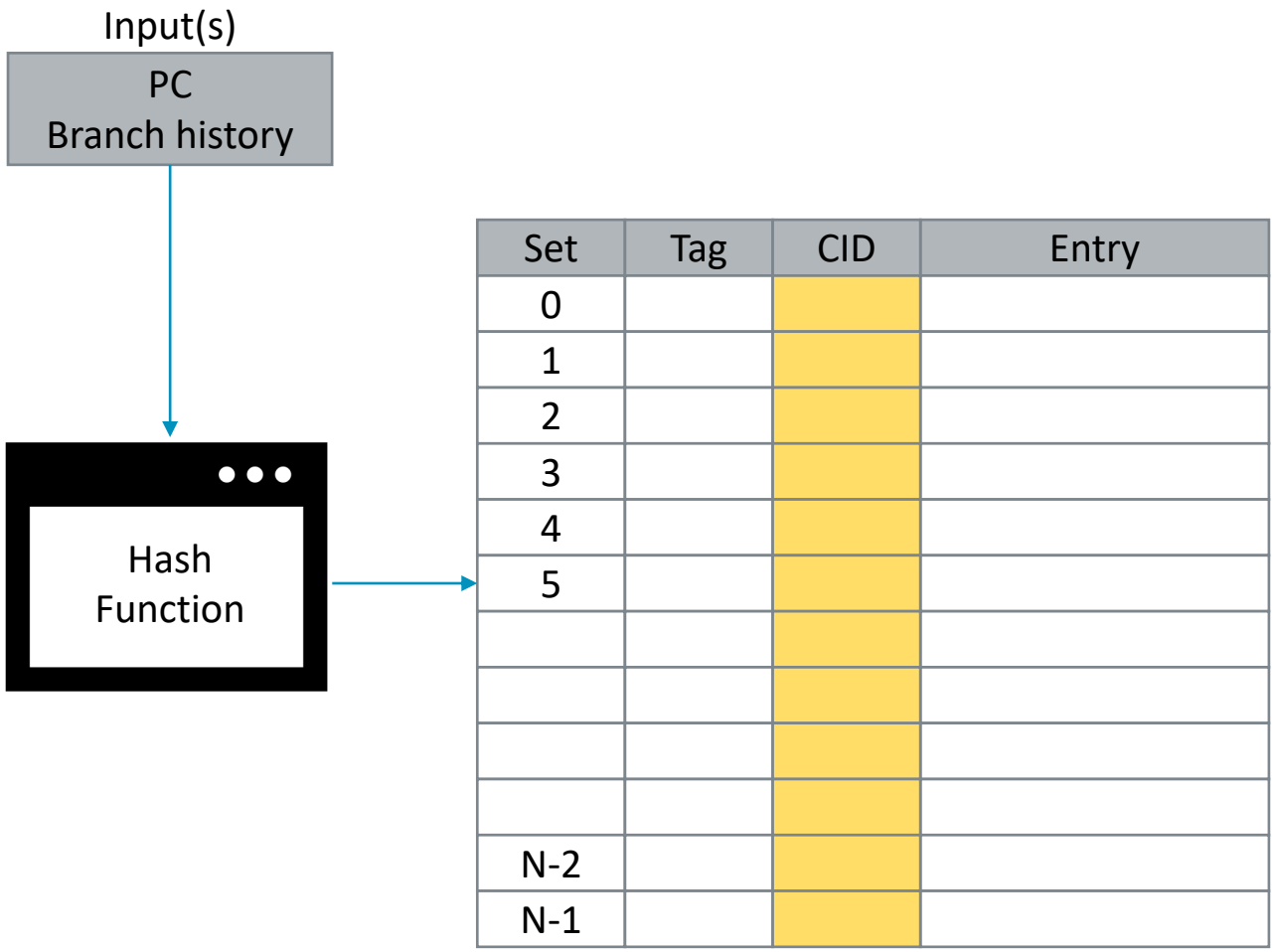
1. Tagging (branch history)

- Add tags (branch history).
  - Example) TAGE predictor
- Full tag is expensive (w.r.t. storage).
  - Partial tag matching: low N-bits are used.
  - This results in aliasing (false hit) and the attacker can still exploit the branch predictor.



2. Tagging (Context ID)

- Add context ID (CID).
  - Intel
    - 12-bit process-context ID (PCID)
    - 16-bit virtual processor ID (VPID)
    - 4 protection levels (Ring 0-3)
  - Arm
    - 8-bit address space ID (ASID)
    - 8 or 16-bit virtual machine ID (VMID)
    - 4 exception levels (EL0-3)
- Too expensive: ~20-bit per entry.



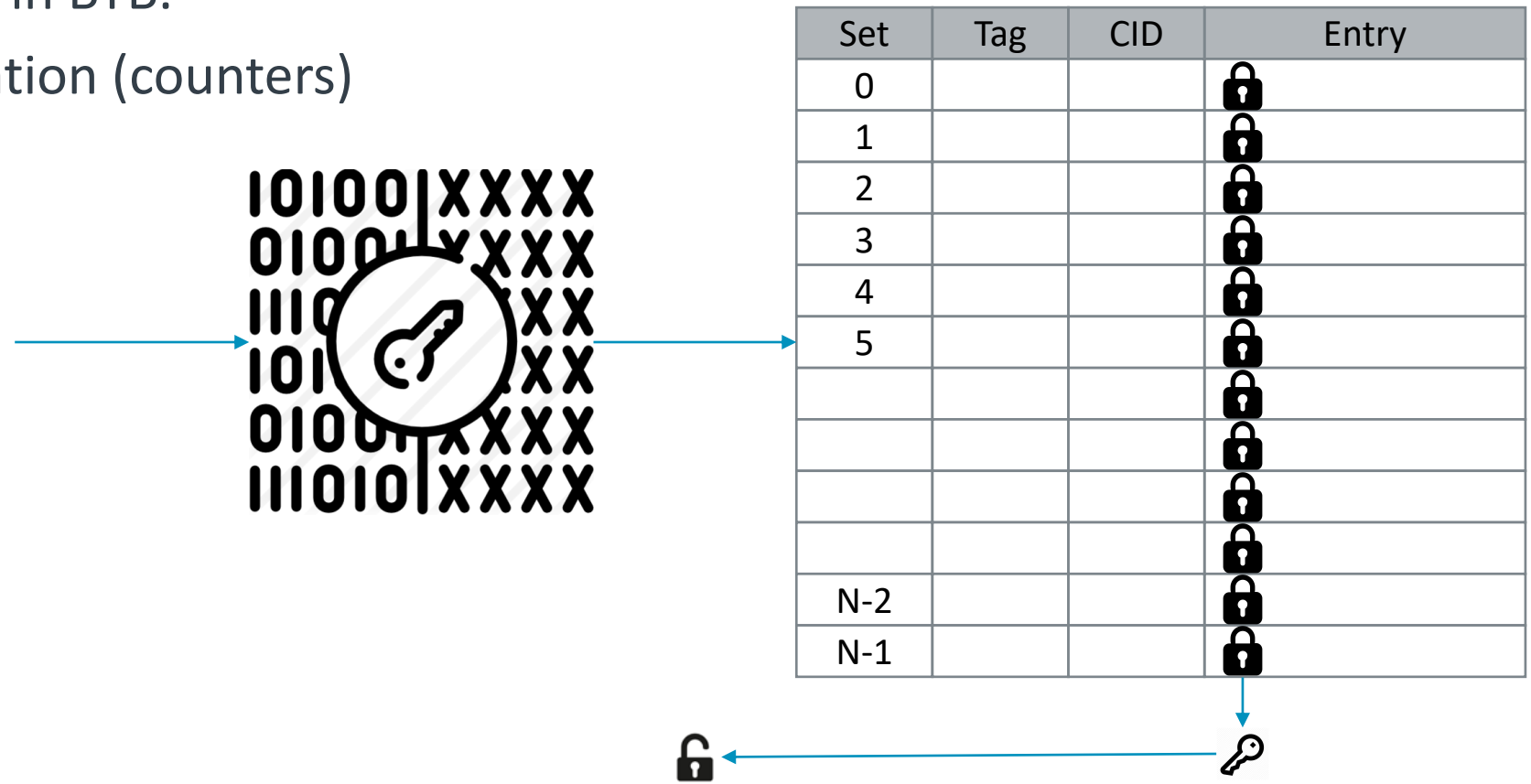
3. Partitioning

- Statically or dynamically partition entries among contexts.
- Provide strong isolation.
- However, effective size will be reduced.
  - What if 10+ processes share the branch predictor with frequent context switches?

Set	Tag	CID	Entry
0			
1			
2			
3			
4			
5			
N-2			
N-1			

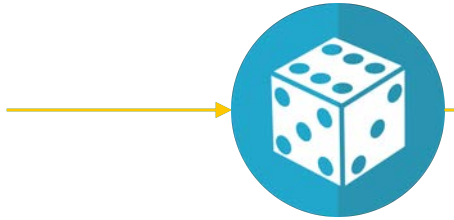
4. (Content) Encryption

- Encrypt the information – the attacker can read the information that is useless without knowing the key.
- Branch target information in BTB.
- Branch prediction information (counters) in directional predictors.



5. Randomization (Index Encryption)

- Randomize indexing to avoid branch collisions.
  - CEASER [Qureshi'18]: Cache index randomization.
  - Physical line address is encrypted using global key.
  - Physical (line) address to set mapping is hard to know.
  - Still, branch collision can occur due to the global key.



Set	Tag	CID	Entry
0			
1			
2			
3			
4			
5			
N-2			
N-1			

6. Branch Retention Buffer

- Vougioukas et al. “BRB: mitigating branch predictor side-channels” in HPCA 2019.
- Partitioning + Checkpointing:
  - Each context maintains small tables to keep important branch predictor entries.
  - All other predictors will be reset upon context switch.
  - Upon being active, a context can quick-start branch predictions from small retention buffer.

Our Proposal

- Protecting index function using **randomization**.
  - Each context has its own key to encrypt PC.
  - Consequently, branch predictor accesses will be randomized per context.
- Protecting contents using **encryption**.
  - BTB: encrypt target information.
  - Directional predictors: encrypt branch prediction information (counter value).
- Dynamic key changes to enhance robustness.
  - Update previously trained information with a new key to alleviate performance degradation.

Our Proposal (Continued)

- Similar to CEASER, but dynamic update cannot be easily applicable due to timing criticality in the branch prediction.
- **Index** encryption: simple weaker encryption to enable quick index encryption and easier dynamic update upon key changes.

Simpler encryption for the index

Set	Tag	CID	Entry
0			
1			
2			
3			
4			
5			
N-2			
N-1			

Stronger encryption for the content

- **Content** encryption: in less timing critical path, so use stronger/more complex encryption.

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

Branch Predictor Side-Channel

- Branch direction.
  - Spectre v1 (array bound check bypass)
    - Train branch predictor to strongly taken.
    - Lead the control flow to wrong-path instructions.
  - Branch shadow and BranchScope:
    - Prime the PHT (strongly taken or strongly not taken).
    - Let victim run a branch.
    - Regain control and run shadow branch.
    - Identify the branch outcome of the victim.
- BTB (Spectre v2) – inject branch target to lead control-flow to malicious gadget.
- RAS (return address stack) – corrupt RAS to mismatch with the address stored in the memory stack.

Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing, Lee et al., Security 2017

BranchScope: A New Side-Channel Attack on Directional Branch Predictor, Evtyushkin et al., ASPLOS 2018

Branch Predictor Side-Channel – BTB

- Branch target injection
  - Spectre v2: using branch collision, injected target is used for the target prediction, which leads to the gadget. Through wrong-path instructions, secret will be written into the cache.
  - Branch Shadow and BranchScope: using branch collision, try to identify the outcome of branch in the victim applications.

Branch Predictor Side-Channel – RAS

- Return Address Stack (RAS)
  - For ret instructions, return address resides in the stack. Upon function call, return address is stored in the RAS.
  - Maintaining call order is quite simple with RAS.
  - Return address will be eventually read from the stack, but this is slow memory operation.
  - Return address is predicted from RAS.

- Conventional attacks: corrupt return address in the stack by stack overflow attack.

- Spectre attacks: tainted target from the RAS temporarily lead to the gadget + speculative executions + secret written into the cache.

How to Mitigate Branch Predictor Side-Channel Attack

- Software-based mitigations
  - Algorithmically remove dependencies between secret value and branch outcome.
  - Eliminate all conditional branches (with cmov).
  - Control flow randomization (obfuscation).
    - ZigZagger: execute all branches, but the real target is set using cmov instruction.
    - Randomized trampolines.

- Hybrid
  - Remove prediction for sensitive branches.
  - Hint from programmer/compiler/run-time
  - Hardware-based prediction bypass.

Is Static (fixed) Protection Robust?

- Encryption vastly improves security.
- However, fixed/static encryption key is known to be vulnerable.
  - Liu+, Last-level cache side-channel attacks are practical, S&P 2015.
  - Requires roughly $(0.42 \times \#\ \text{lines})^2$ accesses, which is 22 seconds for 8MB LLC.

- As a result, we need to periodically change the secret key ➔ Dynamic encryption.

Dynamic Encryption (Key Changes)

- For each context, the secret key periodically changes.
- However, all trained information will be lost.
  - **Do nothing** – lose information, maybe some reuse, maybe security risk?
    - Performance loss, security risk
  - **Reset** – lose information, but prevent malicious exploit by the adversary.
    - Performance loss (due to loss of trained data), performance loss (due to reset process)
  - **Store** key information within the entry.
    - Hardware overhead (per entry key storage)
  - **Update** all information with a new key.
    - Our goal to quickly update entries with the previous key without affecting performance and without hardware overhead.

How Encryption+Randomiztion Works?

- Each region has a secret key.
- Set id is encrypted using the secret key.
- Branch information is also encrypted.

Region Table

Region ID	ASID	EL	Key
0	0		0x032a
1	1		0x7bd1

ASID	PC	Target	Orig Set
0	0x800000c0	0x800000d8	0x00c0
1	0xC00000c0	0xC00000e4	0x00c0

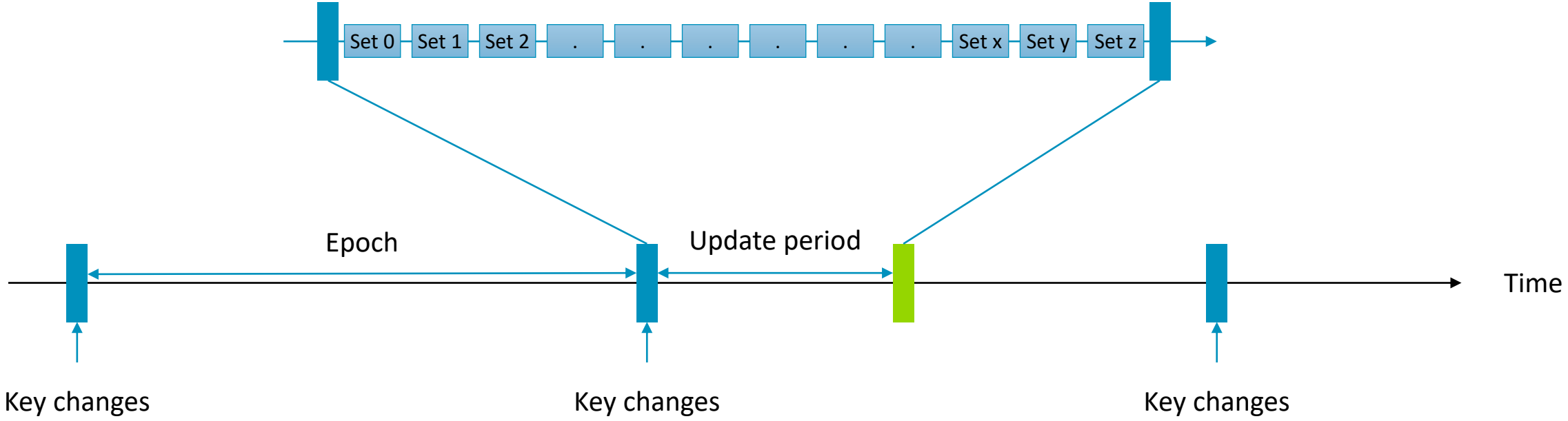
Secret key
0x032a
0x7bd1

New Set
0x03ea
0x0b11

Set	Tag	RGID	Entry
0			
1			
2			
3			
810		0	0x800003f2
2833		1	0xc0007b35
N-2			
N-1			

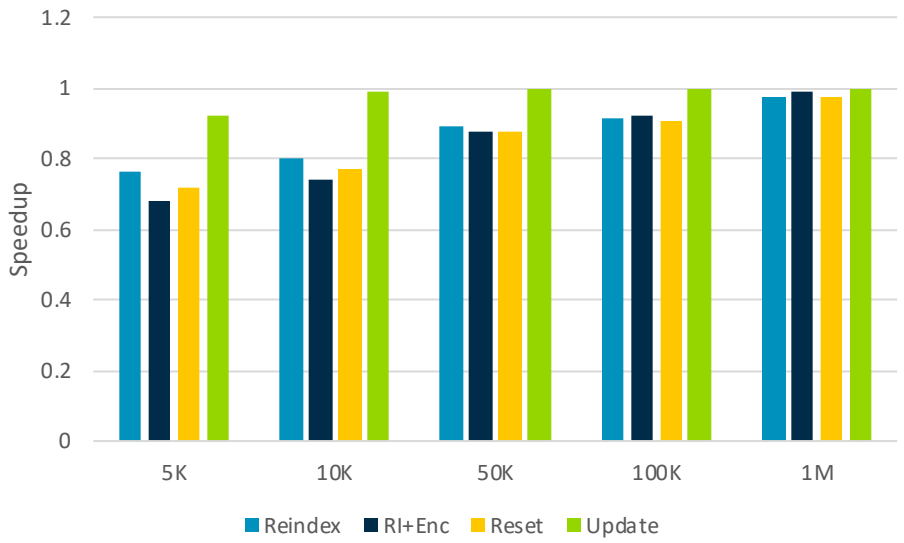
How to Update Branch Predictor?

- Epoch: for each context, the same secret key is used for this execution duration.
- After key changes, we update branch predictor entries
  - Gradually to minimize performance penalty.
  - Deterministically to know which key to use to access (due to gradual update)
  - Bank-level parallelism aware to minimize the update latency.



Results

- ACME with (old) MTH configuration, SPEC suites.
- Epoch length: 5K to 1M
- Mechanism:
  - Reindex (do nothing)
  - Reindex with branch information encryption
  - Reset
  - Update
- Even with a very short 5K-cycle epoch, 7.5% degradation, while others show > 20%.
- 10K-cycle is sufficient to perform the update process.



Next Steps

- New side-channel attacks revealed every month/quarter, but also various mitigation techniques are proposed.
- Doors are closing quicker than I originally thought.
- I want to move on to
  - Conventional system security.
    - Memory (heap/stack) protection, JOP/ROP, code-reuse attack.
  - How to improve the security of SMT processors (which is in consideration in future Arm cores).
  - How to efficiently use Arm security features (pointer authentication, MTE, Newmore/Towmore).

arm

Securing Memory in untrusted environments

Arm Research Summit

Prakash Ramrakhyani

09/16/2019

Securing Memory at the Edge: Motivation

Data Center Nodes

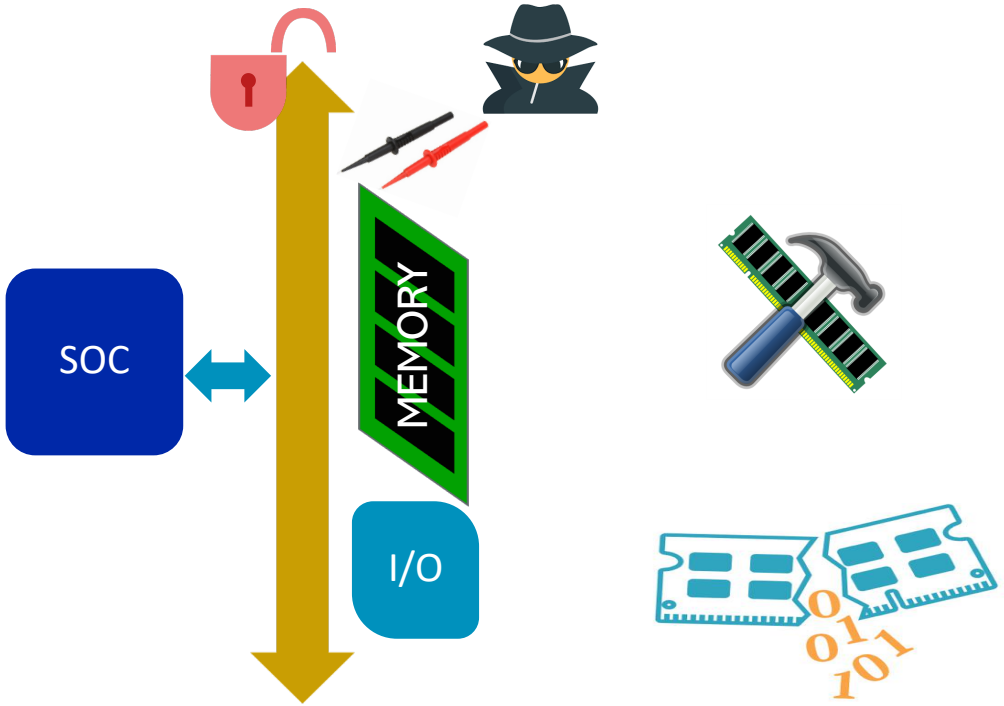
Thousands deployed in secure facilities

Edge Infrastructure Nodes

Millions deployed in the field

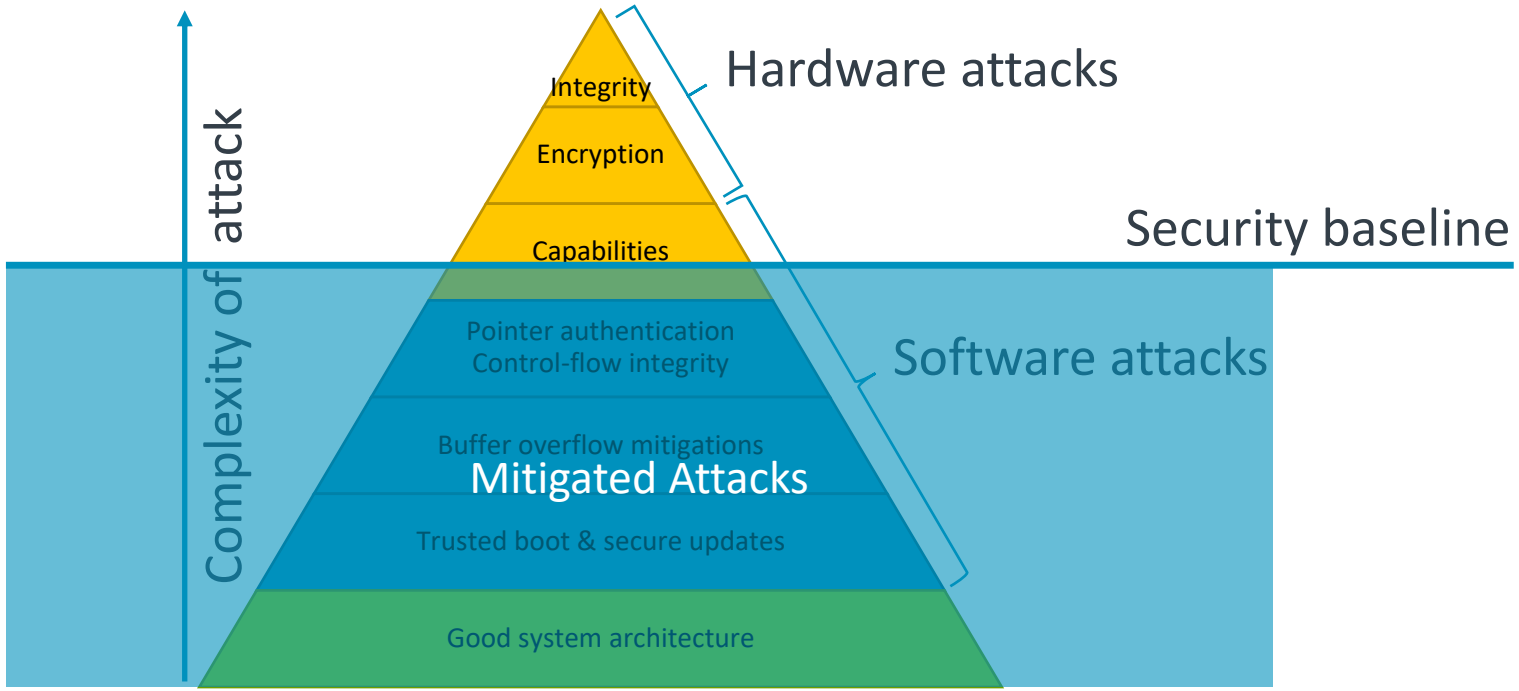
Smart Devices

Billions deployed in the field



- Advent of 5G and edge computing implies third party handling of sensitive data may no longer be limited to physically secure data centers
- Physical access can render the system vulnerable to cold boot attacks. Insecure interface implies that malicious agents can circumvent software based access control mechanisms and modify data/programs residing in memory
- As attacks get more sophisticated (eg: ECCploit) existing detection/defense mechanisms (ECC) are rendered ineffective.

Physical memory attacks are next

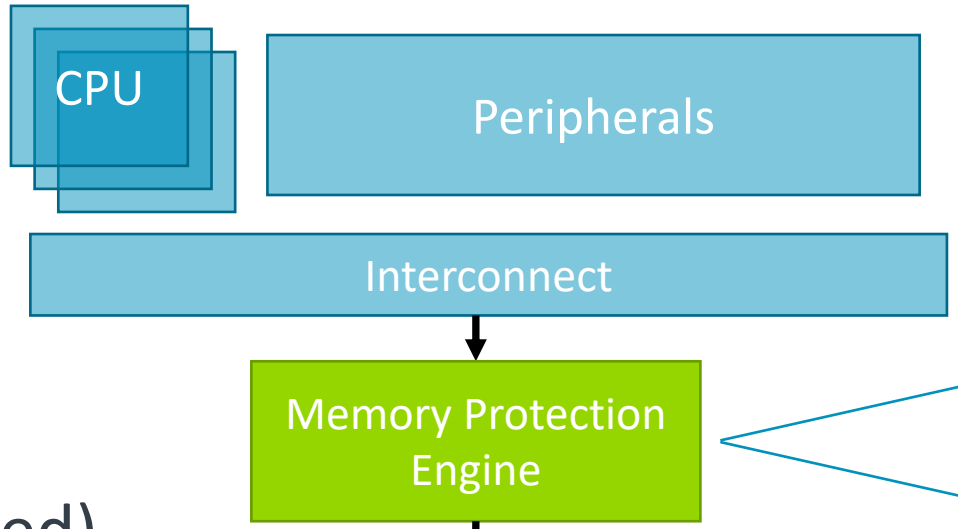


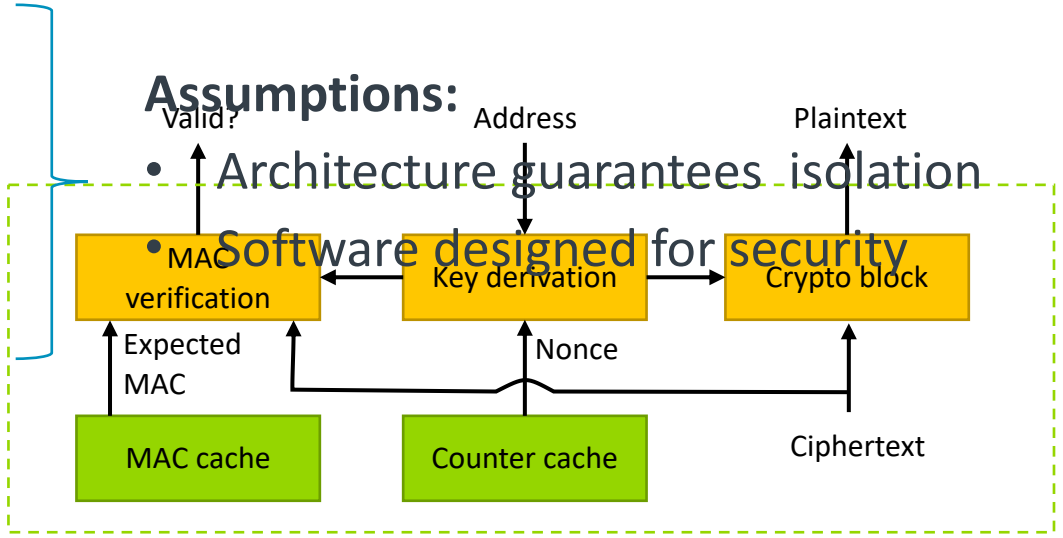
New technologies raise the baseline level of security

- PSA
- Pointer authentication
- Memory Tagging

Physical attacks on off-chip memory can now be the next cheap attack vector

High-level Architecture



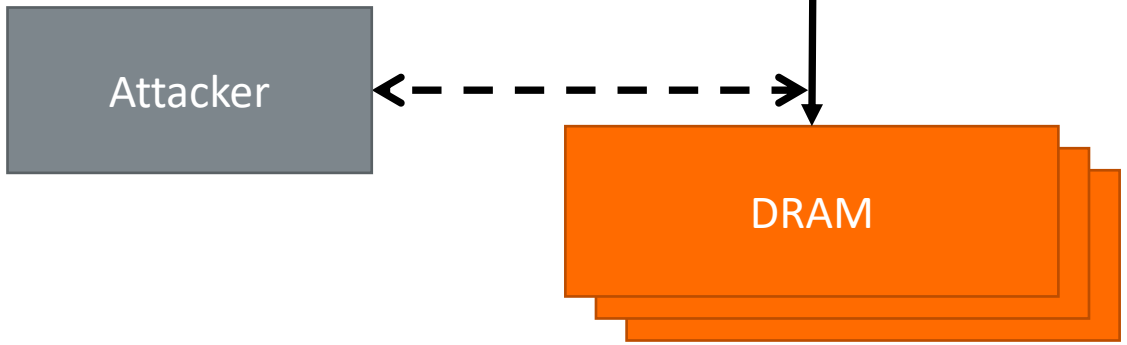


Assumptions:

- Architecture guarantees isolation
- Software designed for security

On chip (trusted)

Off chip (untrusted)



Attack modes:

1. Cold-boot readout
2. Passive listening
3. Fault injection
4. Targeted patching
5. Replay
6. Address bus side-channels

Encryption

MACs

Integrity trees

Address Obfuscation

Cost of Memory Protection

Complexity of attack

Full Integrity

- Protects: Memory replay attacks
- Storage: MACs (12.5%), tree (2%-14%)
- Worst-case perf.: +9 memory accesses
  +9 MAC calculations
  (for 8 GB memory)

Limited Integrity

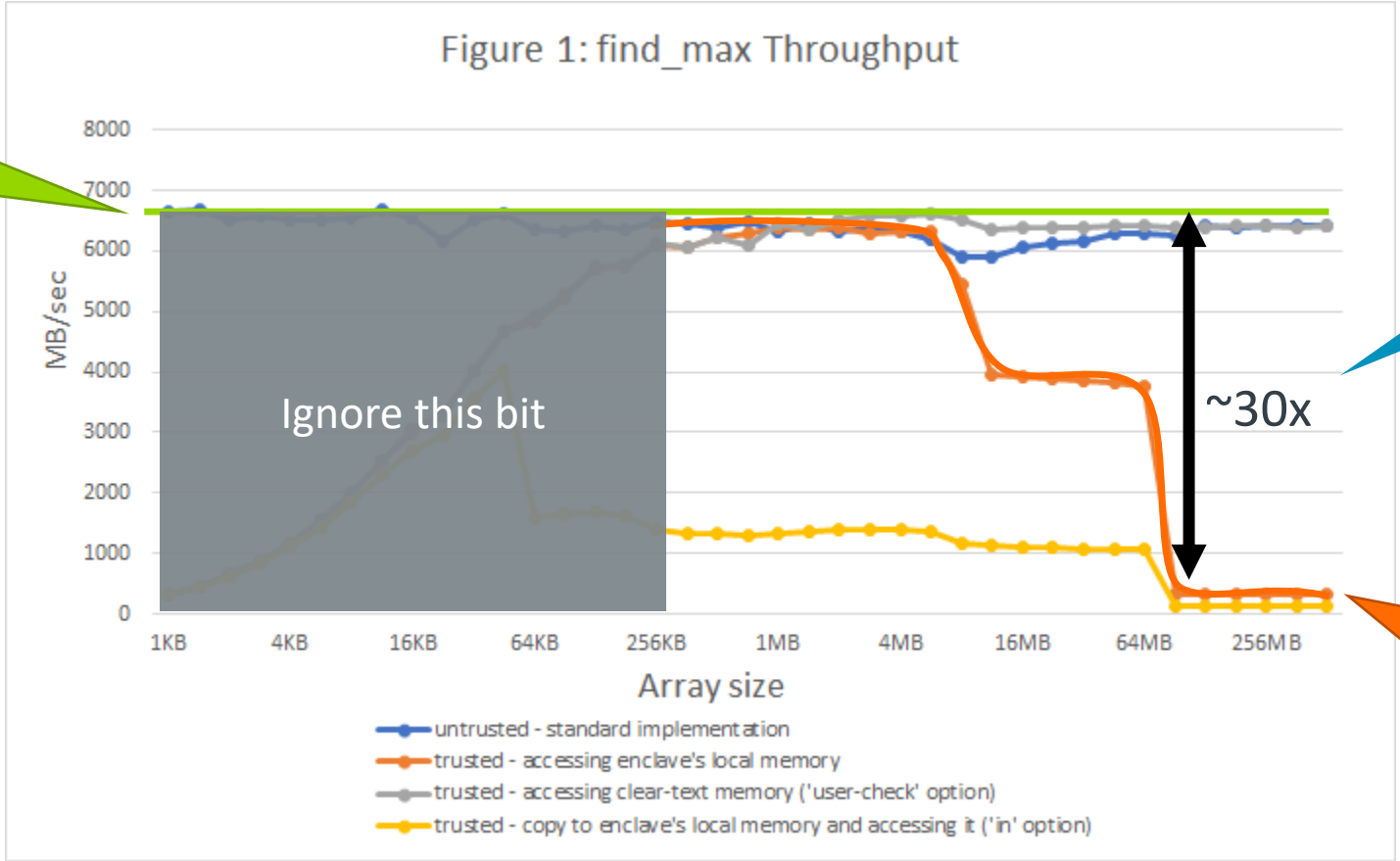
- Protects: Detects data corruption attacks
- Storage: MACs (12.5%)
- Worst-case perf.: +1 memory access
  +1 MAC calculation

Encryption

- Protects: Confidentiality
- Storage: none
- Performance: 5-15ns additional latency

Memory bandwidth in SGX

Reference bandwidth

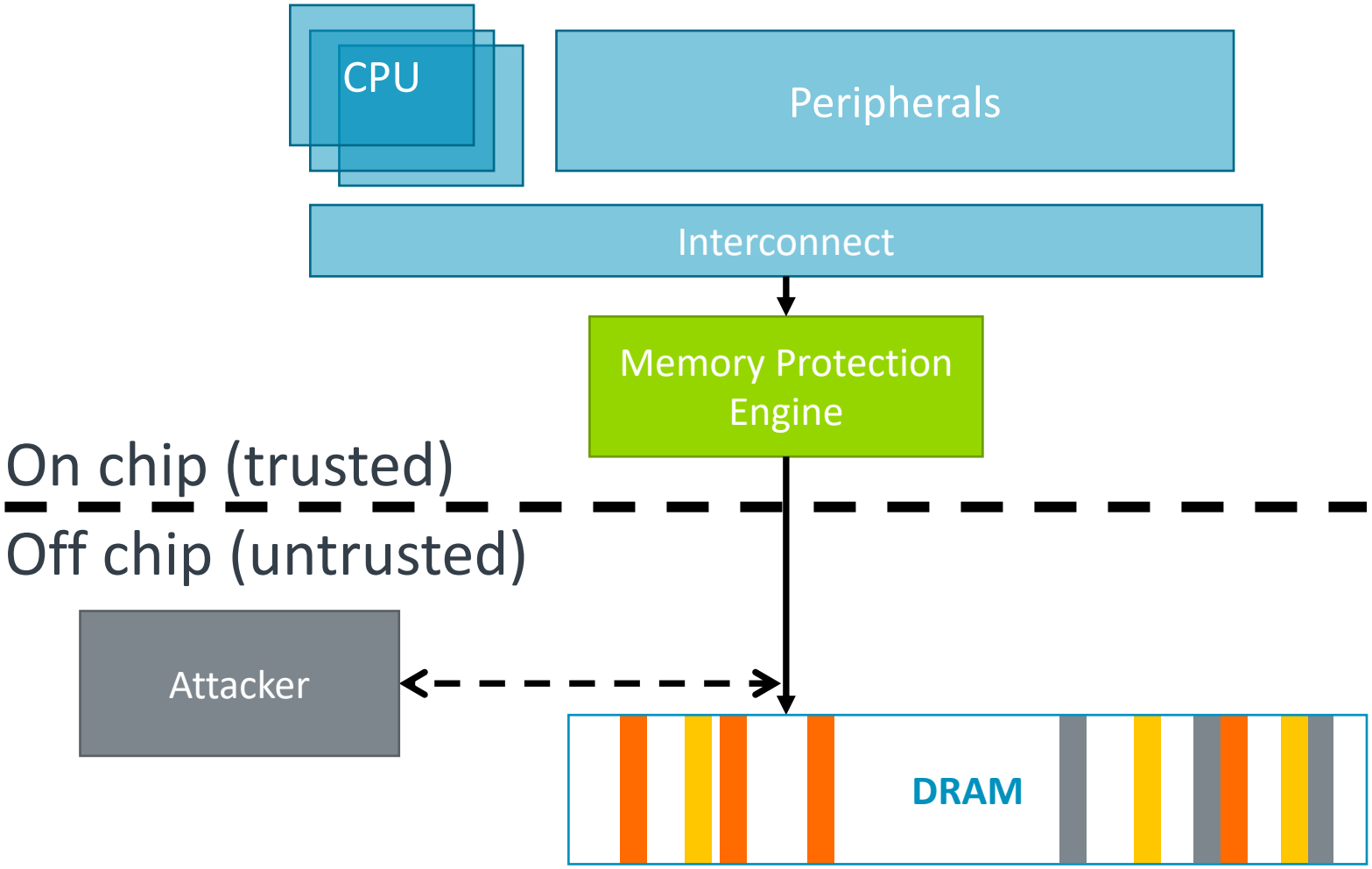


How can this be justified?

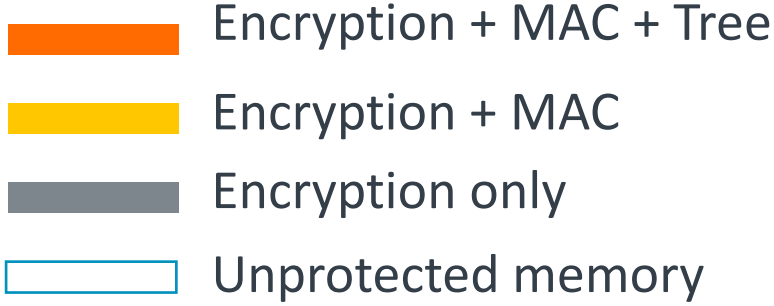
SGX bandwidth

Danny Harnik & Eliad Tsfadia. Impressions of Intel® SGX performance.

In an Ideal World: Scalable Selective Memory Protection



Absorb the greatest cost of protection only for the most sensitive regions of memory

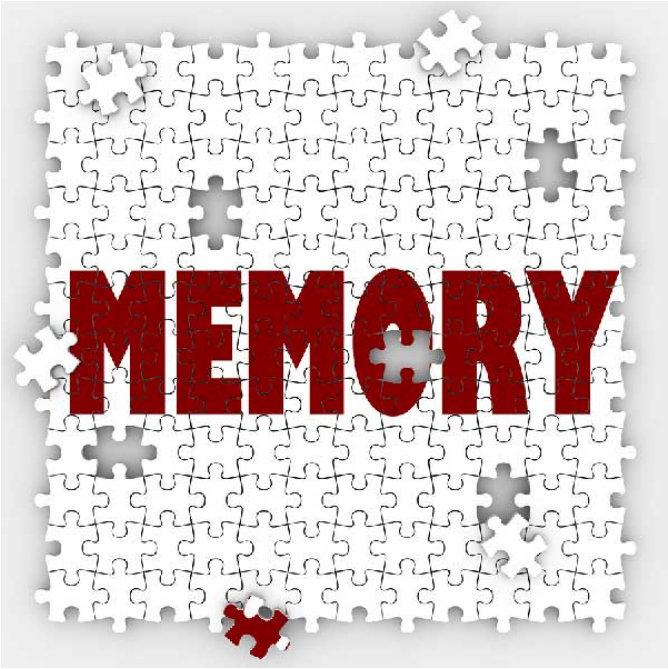


Address based attacks are on the horizon

- Typical secure enclave implementations specify encryption and integrity protection as requirements, but do not require securing the address bus.
- Pattern of accesses made to different regions of memory can reveal adequate side-channel information to impact security
- Controlled channel attacks involve untrusted privileged software inferring information by observing paging behavior of applications
- Research group at Cornell University has demonstrated ability to reverse engineer a CNN simply by observing the pattern of memory accesses made by the CNN accelerator

Re-piecing the puzzle

- Rethink memory management
  - Move either all or some functions for virtual memory management out of the OS
  - Eg: Barrelfish OS
- Newer interfaces
  - Encrypt address and command buses in addition to data bus
  - Point to point intent based interfaces lend themselves naturally to such requirements Eg: Gen-Z
  - Performance overheads of current implementation of the Gen-Z are a hindrance to immediate adoption



References

- Gerd Zellweger, Simon Gerber, Kornilios Kourtis, Timothy Roscoe. *Decoupling Cores, Kernels, and Operating Systems*. 11th USENIX Symposium on Operating Systems Design and Implementation **OSDI 2014**
- Yuanzhong Xu, Weidong Cui, and Marcus Peinado. 2015. *Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems*. - 2015 IEEE Symposium on Security and Privacy **SP '15**.
- Shaizeen Aga and Satish Narayanasamy. 2017. *InvisiMem: Smart Memory Defenses for Memory Bus Side Channel.* 44th Annual International Symposium on Computer Architecture **ISCA '17**
- Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. *VAULT: Reducing Paging Overheads in SGX with Efficient Integrity Verification Structures* - 23rd International Conference on Architectural Support for Programming Languages and Operating Systems **ASPLOS '18**
- Saileshwar G, Nair P, Ramrakhyani P, Elsasser W, Qureshi M. *SYNERGY: Rethinking Secure-Memory Design for Error-Correcting Memories* - 24th IEEE International Symposium on High-Performance Computer Architecture, **HPCA-2018**.
- Saileshwar G**,** Nair P, Ramrakhyani P, Elsasser W, Joao J, Qureshi M. *Morphable Counters: Enabling Compact Integrity-Trees for Low-Overhead Secure-Memories* - 51st Annual IEEE/ACM International Symposium on Microarchitecture, **MICRO-2018**
- Weizhe Hua, Zhiru Zhang, and G. Edward Suh. 2018. *Reverse engineering convolutional neural networks through side-channel information leaks*. - 55th Annual Design Automation Conference **DAC '18**.
- Cojocar, Lucian, Kaveh Razavi, Cristiano Giuffrida and Herbert Bos. *Exploiting Correcting Codes : On the Effectiveness of ECC Memory Against Rowhammer Attacks* - 2019 IEEE Symposium on Security & Privacy **SP '19**.
- Kwong, Andrew and Genkin, Daniel and Gruss, Daniel and Yarom, Yuval *RAMBleed: Reading Bits in Memory Without Accessing Them*. IEEE Symposium on Security and Privacy in **SP '20.**
- https://genzconsortium.org/white-papers/

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

arm

Security Meetup Lightning Talk

Veracruz
Privacy-preserving compute using trusted hardware

Derek Miller, **Dominic Mulligan**, Hugo Vincent, Shale Xiong

Veracruz

Veracruz is a privacy-preserving compute infrastructure. In the *most general* setting:

- **Secret data** can be fed into,
- **secret programs**, which are offloaded to,
- third parties who **host** the computation, and produce
- **secret results**…

In this setting, everybody is *mutually distrusting* and has individual concerns:

- The data and program owners want to retain their secrets,
- The host does not want their machine to be damaged by programs they cannot audit or monitor

THE TIMES

Log in Subscribe

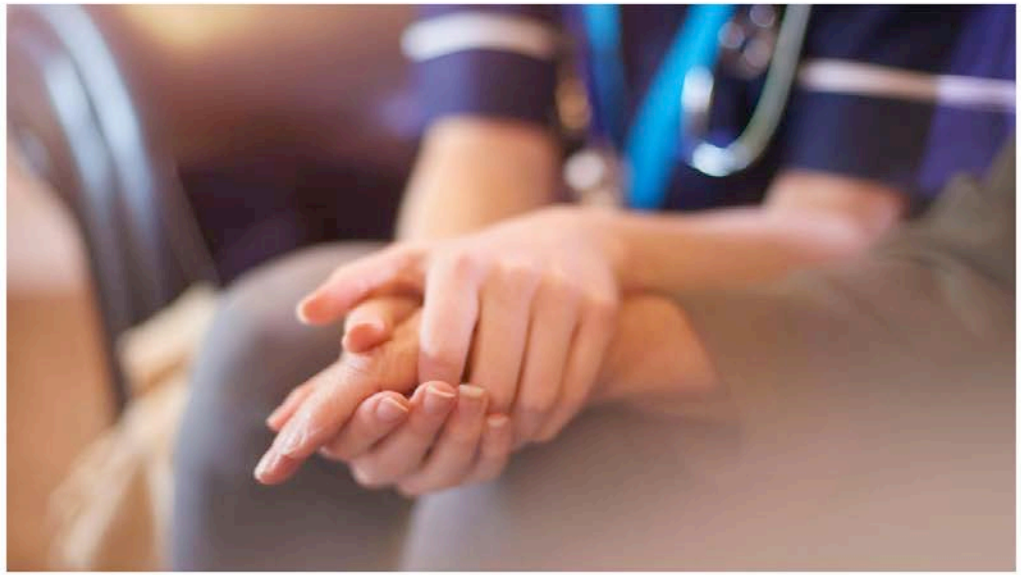
Simon Segars, boss of tech giant ARM Holdings, pleads with NHS to share patients' data

Ben Woods

March 10 2019, 12:01am,
The Sunday Times

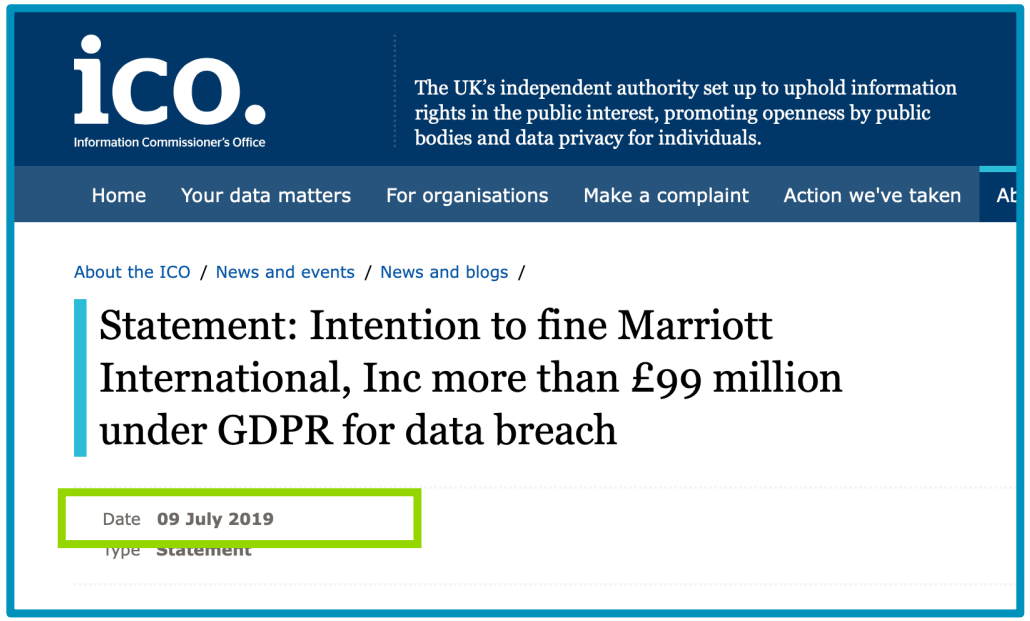
Health NHS Apple

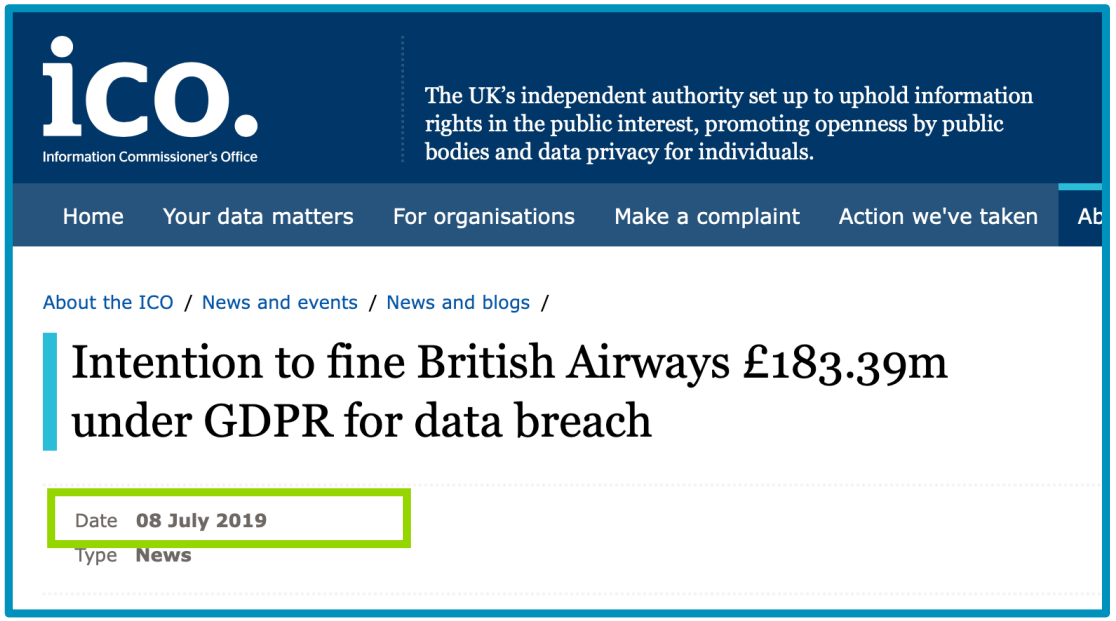
Technology



The government last month announced a new code of conduct for the use of artificial intelligence in the NHS
GETTY IMAGES

The chief executive of the iPhone chip designer ARM Holdings has urged the NHS to give artificial intelligence (AI) developers greater access to its "goldmine"





From trawling through ICO website, these seem to be largest fines they've ever levied

- Prior to this month, fines seemed to be in range £100,000-500,000

Also, first fines for data *breaches*, rather than illegal data access/retention, spam emails

GDPR significantly strengthened regulators, changed business landscape

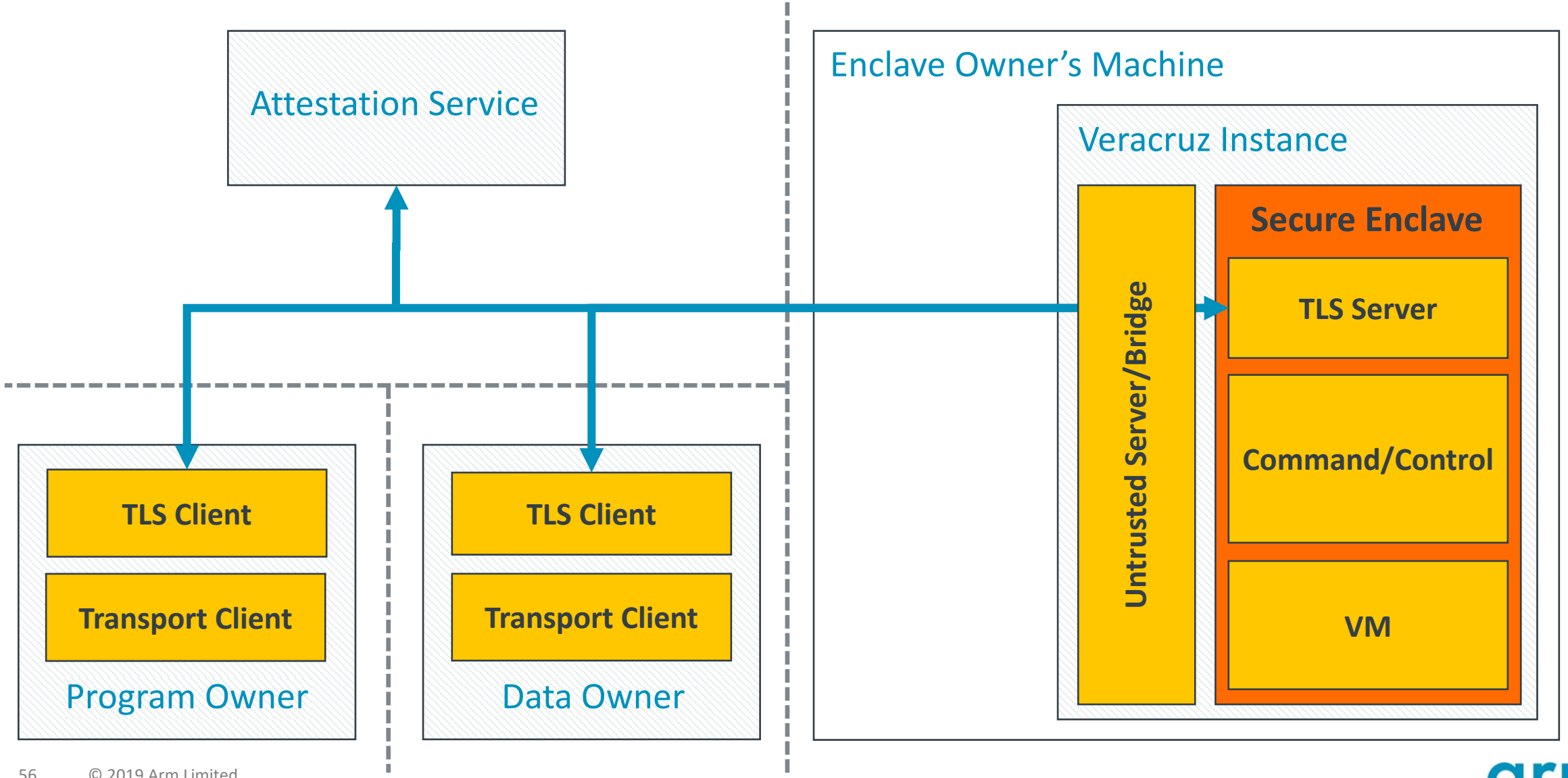
Secure Enclaves

Briefly: Veracruz uses **Secure Enclaves** as a “venue” for performing secret computations

- Secret inputs provisioned into the Enclave securely, via TLS,
- Computation happens in the Enclave, which protects it from prying and interference by the host,
- Result is retrieved by agent according to published policy…

Remote parties can use an **attestation protocol** to ensure Enclave is a legitimate Veracruz instance before they start provisioning secrets

System design



Enclave engineering

Idea: move everything that need not be in the Enclave outside

- Reduces TCB size,
- Reduces possibility of introducing security bugs,
- Increases auditability of Enclaved code

Write *majority* of Enclave contents in Rust

Enclave Engineering

VM prototype implemented in C, executes custom bytecode (“Zocalo”)

- Sandboxes program,
- Insulates host machine from running program,
- Potentially have VM enforce dynamic policies on running program,
- Simple enough (~7,000 LOC) to audit manually

Moreover, use CBMC, the C-bounded model checker, to verify properties of VM:

- **Static analysis**: no memory issues, no undefined behaviours,
- **Verification**: VM fetch-decode behaves as expected, VM (non-floating point) instruction semantics behaves as expected, various reachability properties of code

MVP prototype

Project started in April, have a prototype working with:

- Single data source,
- Single program owner

Capable of executing simple secret machine-learning algorithms in-enclave on secret data

Future work:

- Generalise to N-secret data sources,
- Retain relatively low TCB whilst making Veracruz faster,
- Applications…

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה