Microsoft Azure

Confidential Computing

Cédric Fournet

Azure Research

Swiss Joint Research Centre | Spring Workshop

Zürich
April 2024

What is Confidential Computing?

Existing Encryption		Confidential Computing

Data at rest

Encrypt inactive data when stored in blob storage, database, etc.

The industry moved from disks in the clear to **encrypted disks, with managed keys**

Data in transit

Encrypt data that is flowing between untrusted public or private networks

Evolved from browsing/moving data in the clear (HTTP), to **encrypting data** (HTTPS/TLS)

Data in use

Protect/Encrypt data that is in use, while in RAM and during computation

Evolving from computing in the clear to **Trusted Execution Environments**, like Intel SGX, TDX, or AMD SEV-SNP

see also [Toward Confidential Cloud Computing – Communications of the ACM]()

What is Confidential Computing?

Data in use

Protect/encrypt data that is in use, while in RAM, and during computation

Protection from cloud threats

Malicious admins

Hackers

Unauthorized access, control plane exploits

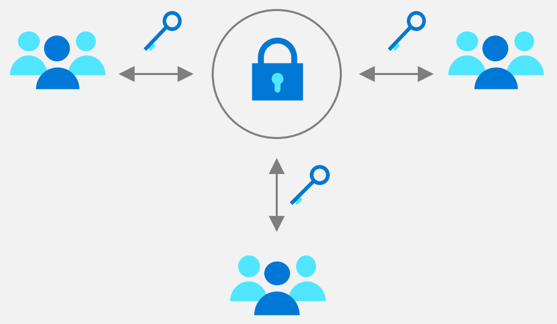
Protect end-user data from CSP & tenant

Guest OS
Host OS

VM admin
Host admin

Hypervisor

Share data with multi-party securely



see also [Toward Confidential Cloud Computing – Communications of the ACM]

10 years of Confidential Computing

Looking back: theory (TTPs, information flow) and early experiments
Now: mainstream hardware support (CVMs) & deployment in production
Next: secure default for the public cloud?

This talk: three ongoing research projects:

1. *How to deploy confidential ML workloads?*
   Azure Confidential GPU VMs with NVIDIA H100 Tensor Core GPUs
2. *How to keep track of attested code?*
   Transparent software supply chain for confidential computing
3. *How to prevent side channels?*
   Principled partitioning and scheduling of microarchitectural resources

Inside Azure Confidential GPU VMs with NVIDIA H100 Tensor Core GPUs

Microsoft + Nvidia, 2020—2024

AI Accelerators is the Next Frontier of Confidential Computing

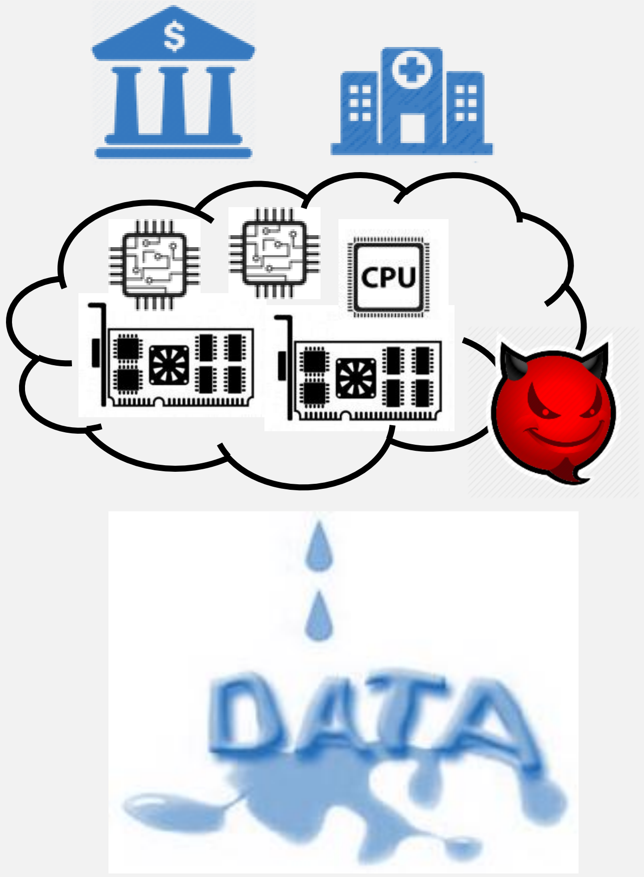
How to get the most value out of sensitive data?

Cloud accelerators enable AI at scale

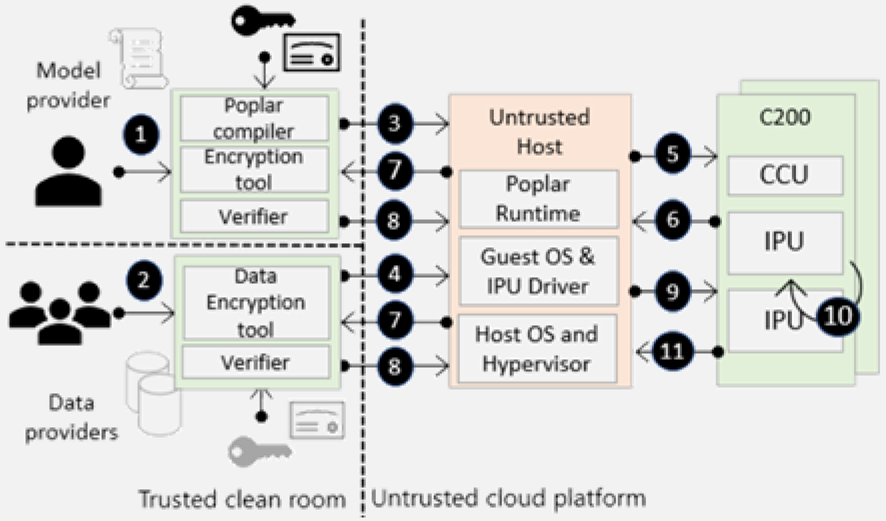
- Medical diagnostics, financial forecasting, generative AI
- Large models require 10^N scarce, high-end GPUs

Ever-growing confidentiality & privacy concerns

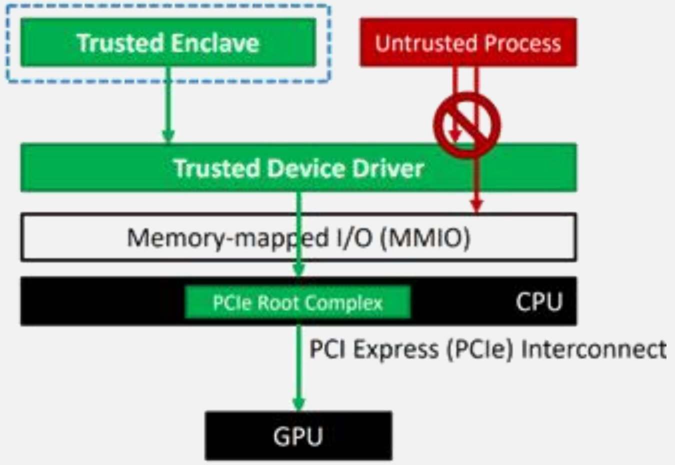
- Privacy-sensitive data (e.g., medical history, transactions)
- Proprietary AI models (e.g. API access to GPT4)



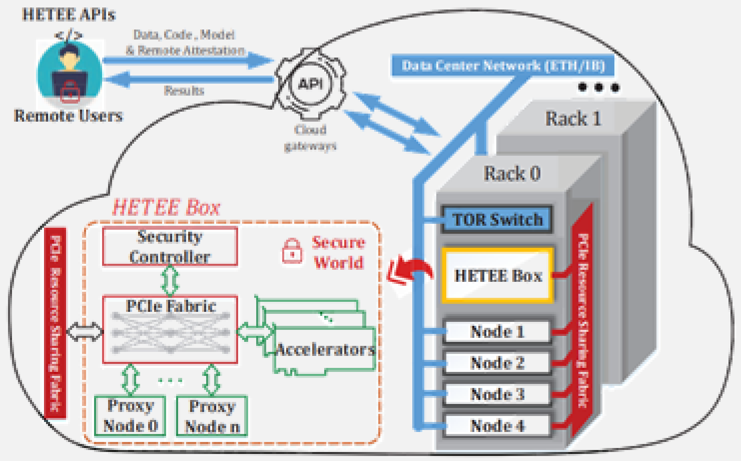
AI Accelerators is the Next Frontier of Confidential Computing



Graphcore [S&P22]: fully desaggregated CC

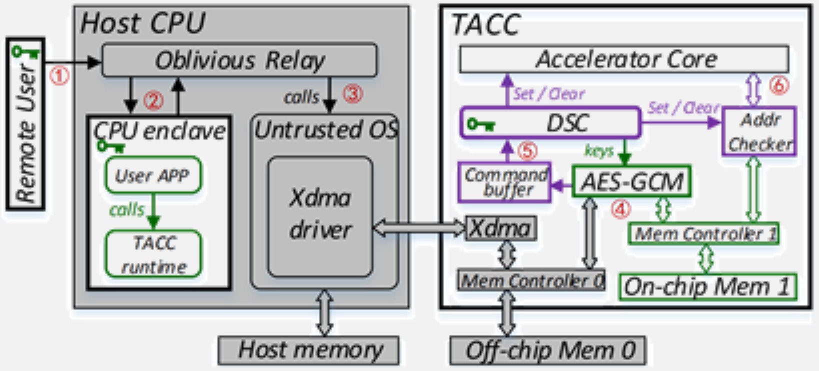


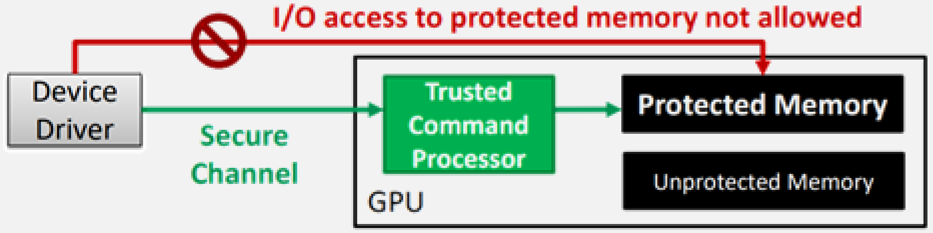
HIX [ASPLOS'19]: Heterogenous TEEs



Rack-scale heterogenous TEE [S&P20]

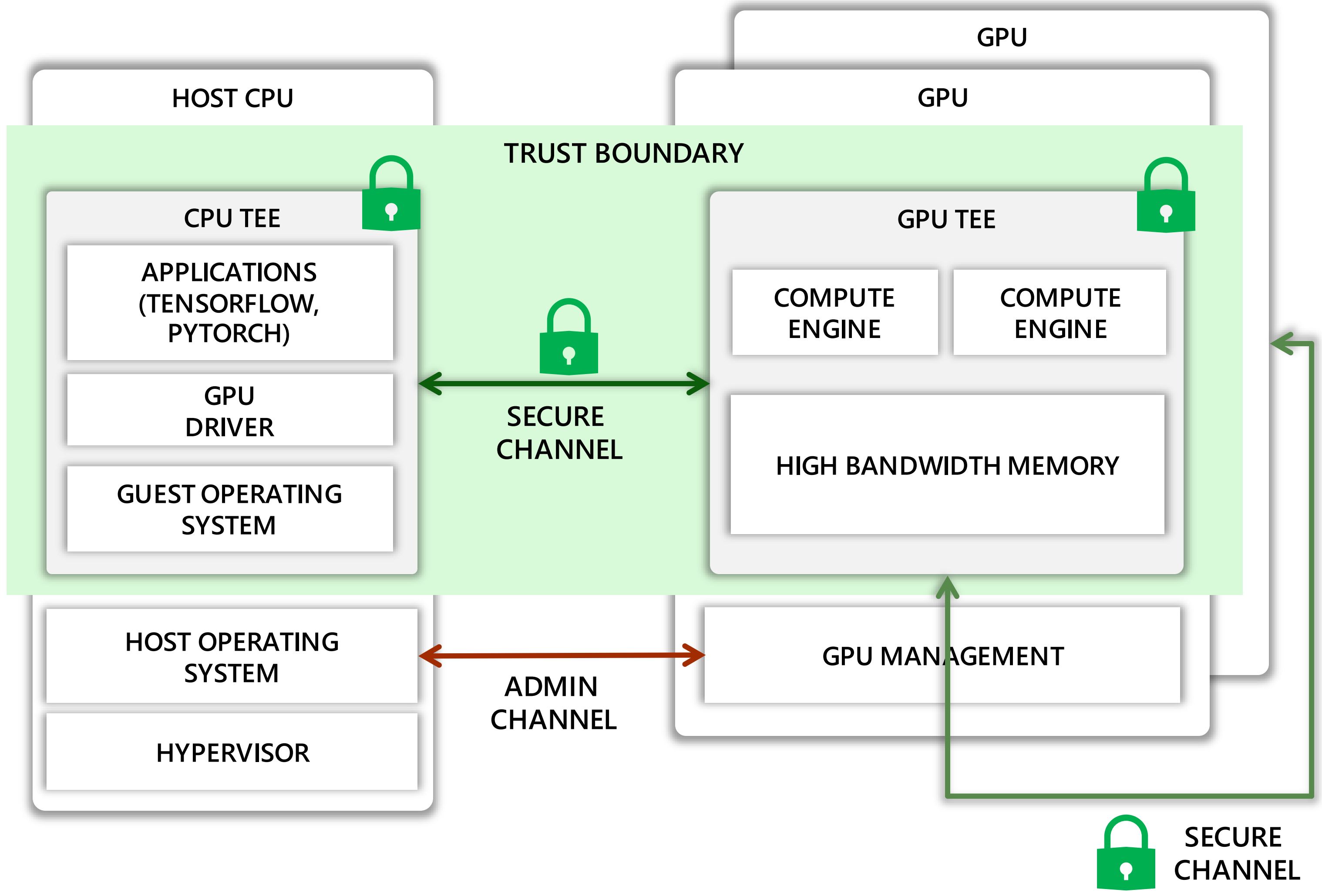
TACC [SYSTOR22]: Secure Accelerator Enclave




I/O access to protected memory not allowed
Device Driver
Secure Channel
Trusted Command Processor
Protected Memory
Unprotected Memory
GPU

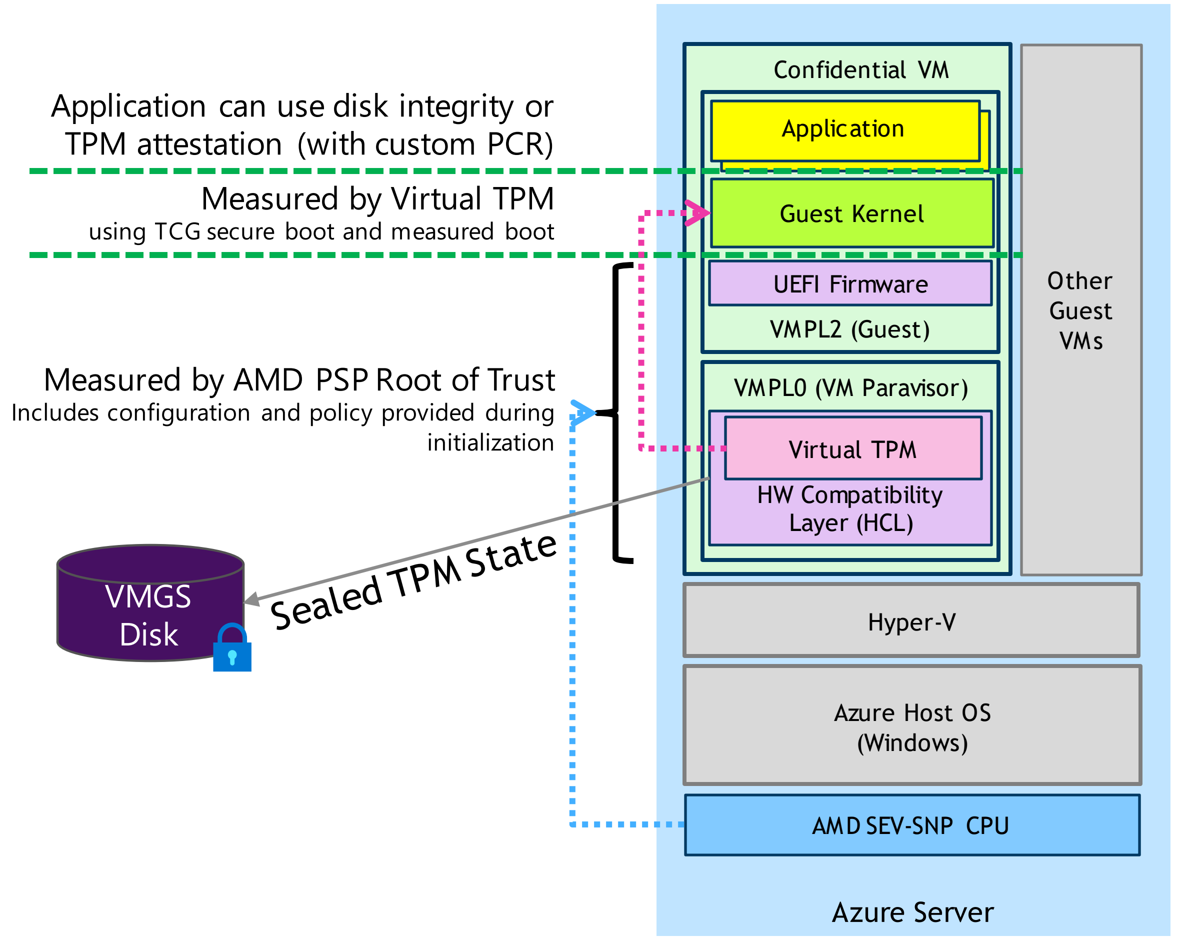
Graviton [SOSP'18]: TEEs on GPUs

Confidential Computing on NVIDIA GPUs: Requirements



- **Compatibility**: should run unmodified CUDA applications
- **Uniformity**: use same GPU driver codebase for CC mode
  - Use VM-based TEE to run GPU driver and CUDA runtime libraries
- **Host & Hypervisor isolation**
  - Memory protection
  - Encrypted commands
  - Encrypted DMA
- **Remote attestation**
- **High performance**

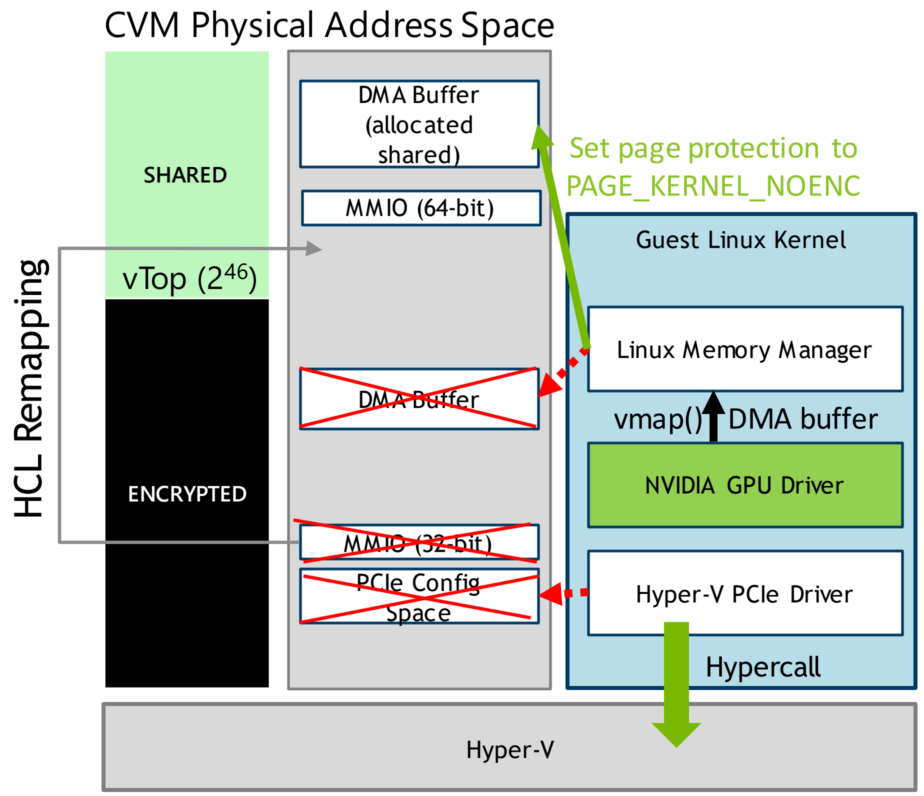
Overview of Azure Confidential VMs (AMD SEV-SNP)


Application can use disk integrity or TPM attestation (with custom PCR)
Measured by Virtual TPM
using TCG secure boot and measured boot
Measured by AMD PSP Root of Trust
Includes configuration and policy provided during initialization
VMGS Disk
Sealed TPM State
Confidential VM
Application
Guest Kernel
UEFI Firmware
VMPL2 (Guest)
VMPL0 (VM Paravisor)
Virtual TPM
HW Compatibility Layer (HCL)
Other Guest VMs
Hyper-V
Azure Host OS (Windows)
AMD SEV-SNP CPU
Azure Server

- Azure Confidential VMs aim to run stock distributions (Ubuntu, RHEL), despite some guest enlightenments (up-streamed / backported by Microsoft).
- Critically, C-bit page encryption and RMP management are handled in HCL rather than in the guest Kernel
- Kernel attestation relies on vTPM for TCG measured boot. Guest applications can be implicitly attested through disk integrity or explicitly with TPM quote & PCR
- We also rely on HCL to provide persistence of the vTPM state (required for guest image encryption & integrity)
- Why should the guest trust the CVM's virtual TPM?
  - The TPM is implemented in a small hypervisor that is measured by the AMD PSP Root of Trust
  - We use the VM Privilege Level of AMD SEV-SNP to offer transparent devices (TPM, disks) to the guest
  - The AMD PSP attestation of the CVM firmware components (HCL, TPM, UEFI) is exposed via the TPM

Microsoft Azure

Assigning PCIe Devices to vTOM Confidential VMs



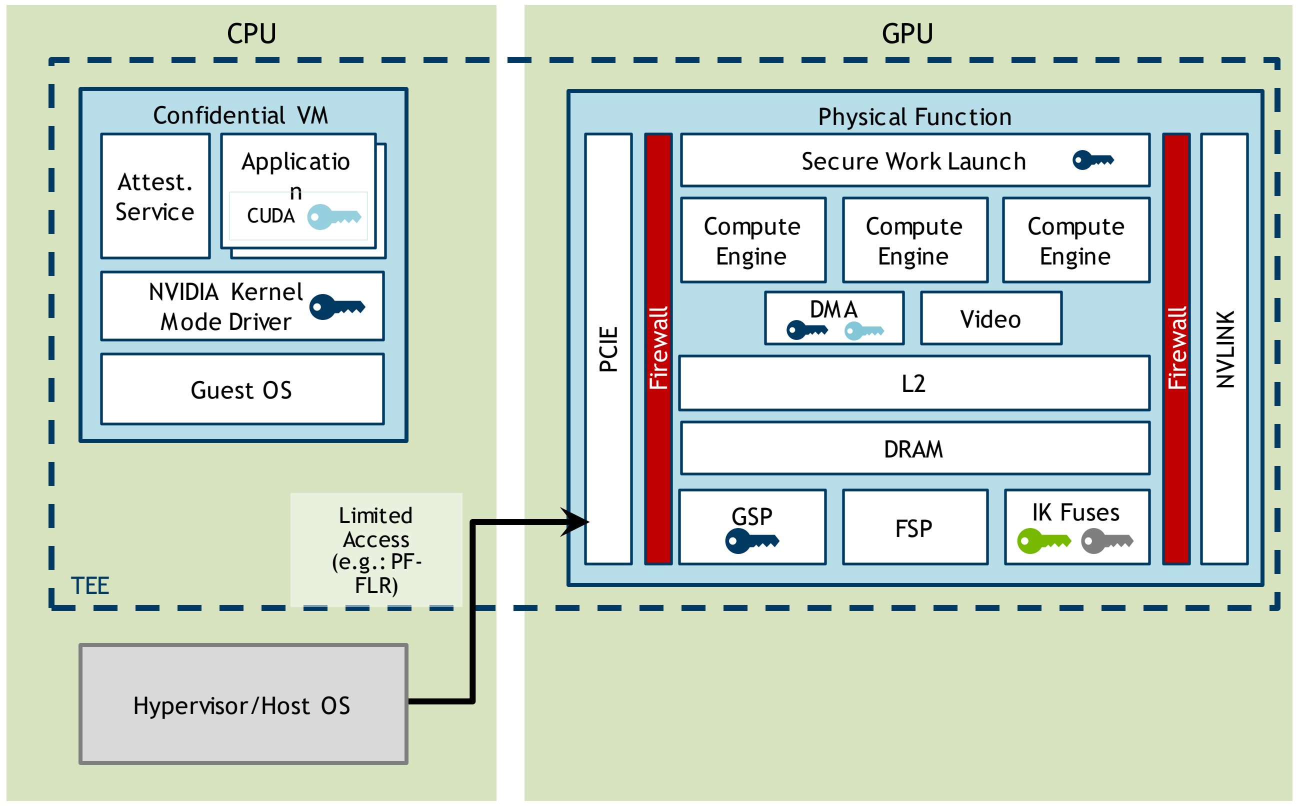
CVMs split the physical address space between encrypted pages (address < vTOM) and shared pages (address >= vTOM).

1. Only shared pages can be accessed outside the CVM. DMA buffers must be mapped above vTOM.
2. PCIe config space access (used to enumerate devices) requires Hyper-V emulation through hypercalls

For drivers: no change when using standard Linux DMA APIs. Other APIs like vmap() require the 'NOENC' flag.

Linux patches are up-streamed to Linux 6.3 and backported to Ubuntu 22.04.

GPU Isolation Features in CC Mode



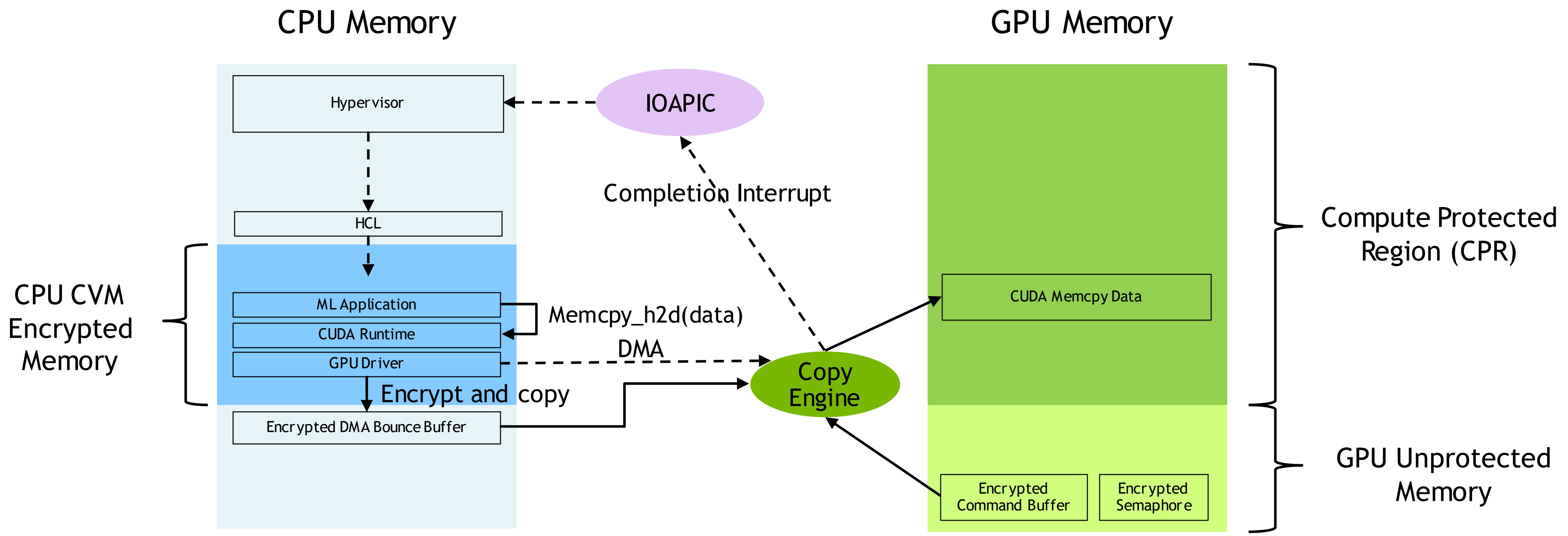
When Nvidia's Hopper GPU boots in confidential mode, it blocks ingress and egress for the Compute Protected Region (CPR) of GPU Memory

- The PCIe Firewall blocks access by the CPU to most registers and all of the GPU CPR Memory
- NVLINK Firewall blocks access by NVLINK peer GPUs to GPU CPR Memory.
- DMA engines can only read or write outside of CPR with encryption enabled
- All other engines (e.g. Compute SMs) are blocked from reading or writing outside of CPR.

The Compute Protected Region of memory is secured so that the GPU can process data at full speed in its High Bandwidth Memory

All GPU performance counters are disabled, to protect against side channels.

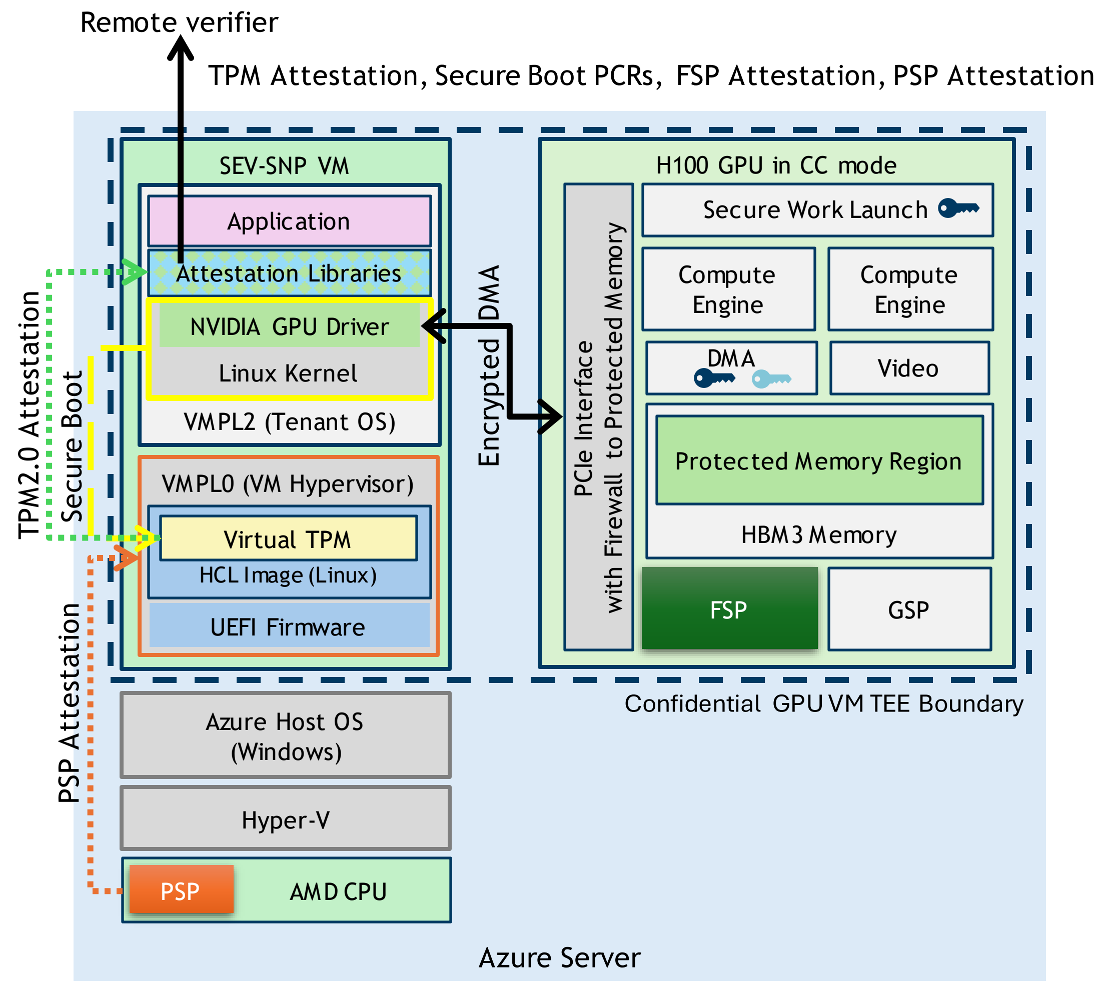
How Memory is Managed in Confidential Mode

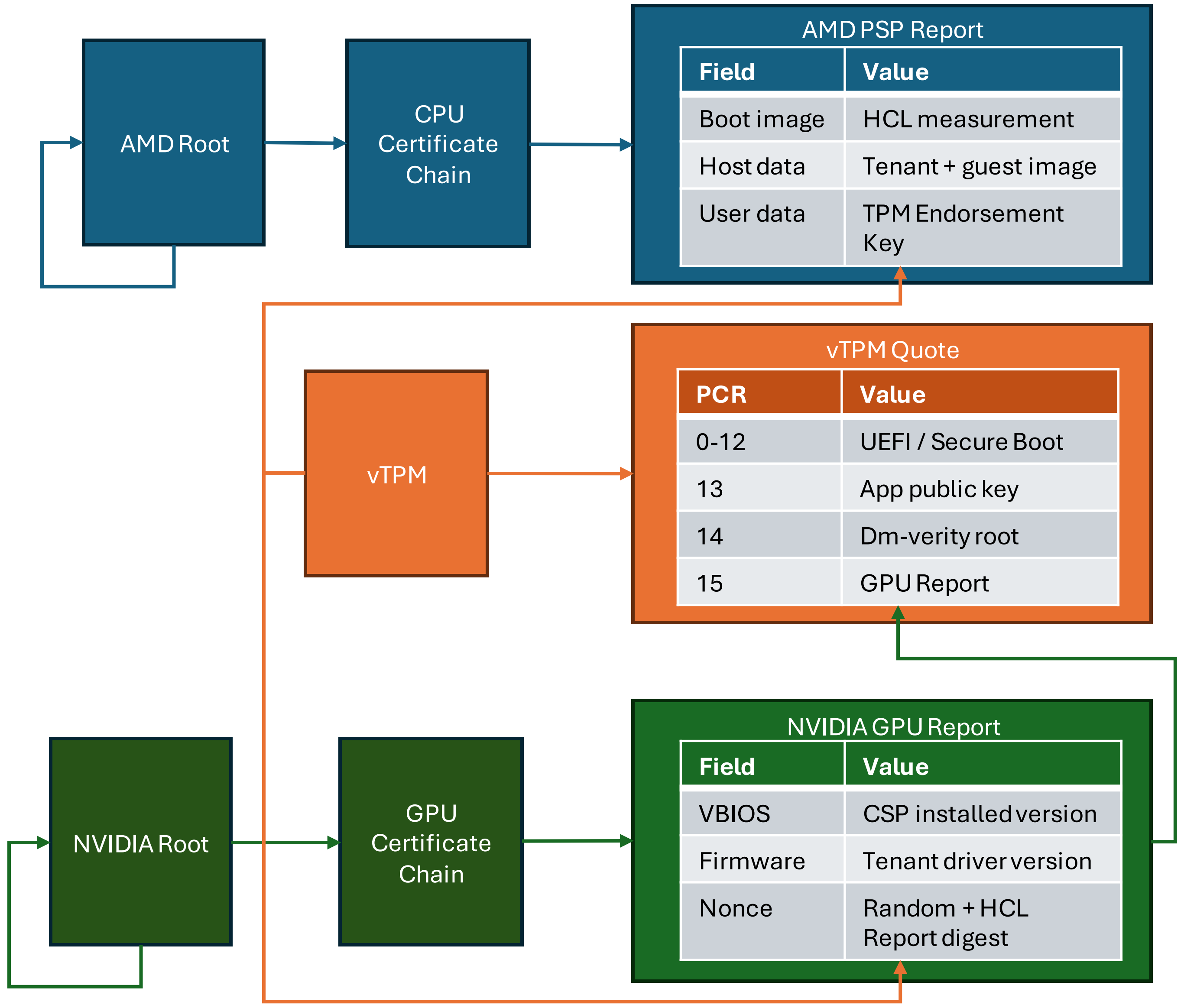


- By default, all Guest VM memory is encrypted by SEV
- To perform a DMA the GPU driver must encrypt and copy to data to a bounce buffer in shared memory page
- The interrupt for DMA must also be re-injected by the Hypervisor. We use HCL to do this efficiently.

- Most of the GPU memory is configured as Compute Protected Region (CPR), protected by hardware firewalls
- A small portion of GPU memory is outside of the CPR and is used for:
  - Encrypted CUDA Command Buffers & Semaphores
  - Bounce Buffers for NVLINK Peer to Peer

Attestation of Confidential GPU VM Applications





AMD PSP Report

Field	Value
Boot image	HCL measurement
Host data	Tenant + guest image
User data	TPM Endorsement Key

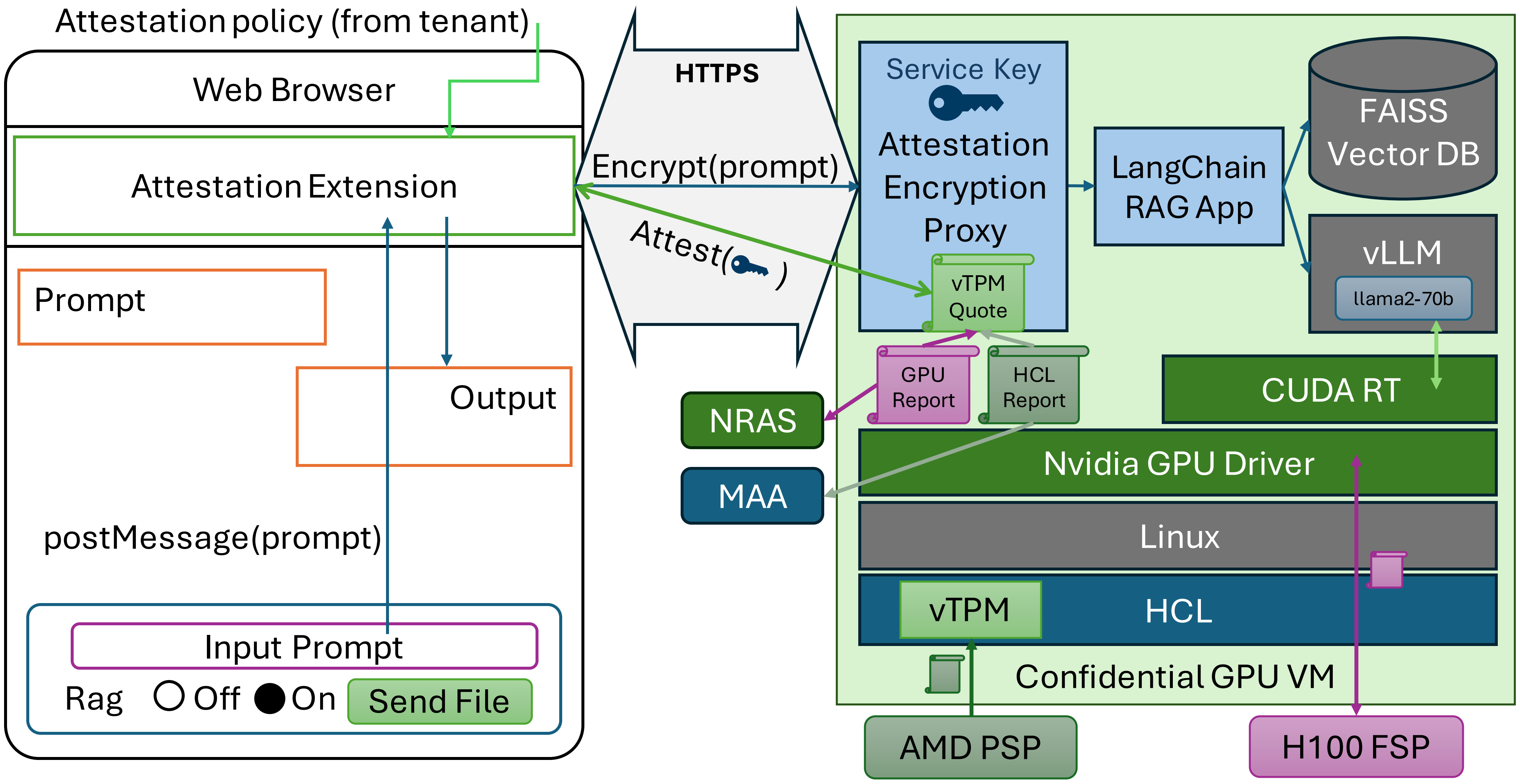
vTPM Quote

PCR	Value
0-12	UEFI / Secure Boot
13	App public key
14	Dm-verity root
15	GPU Report

NVIDIA GPU Report

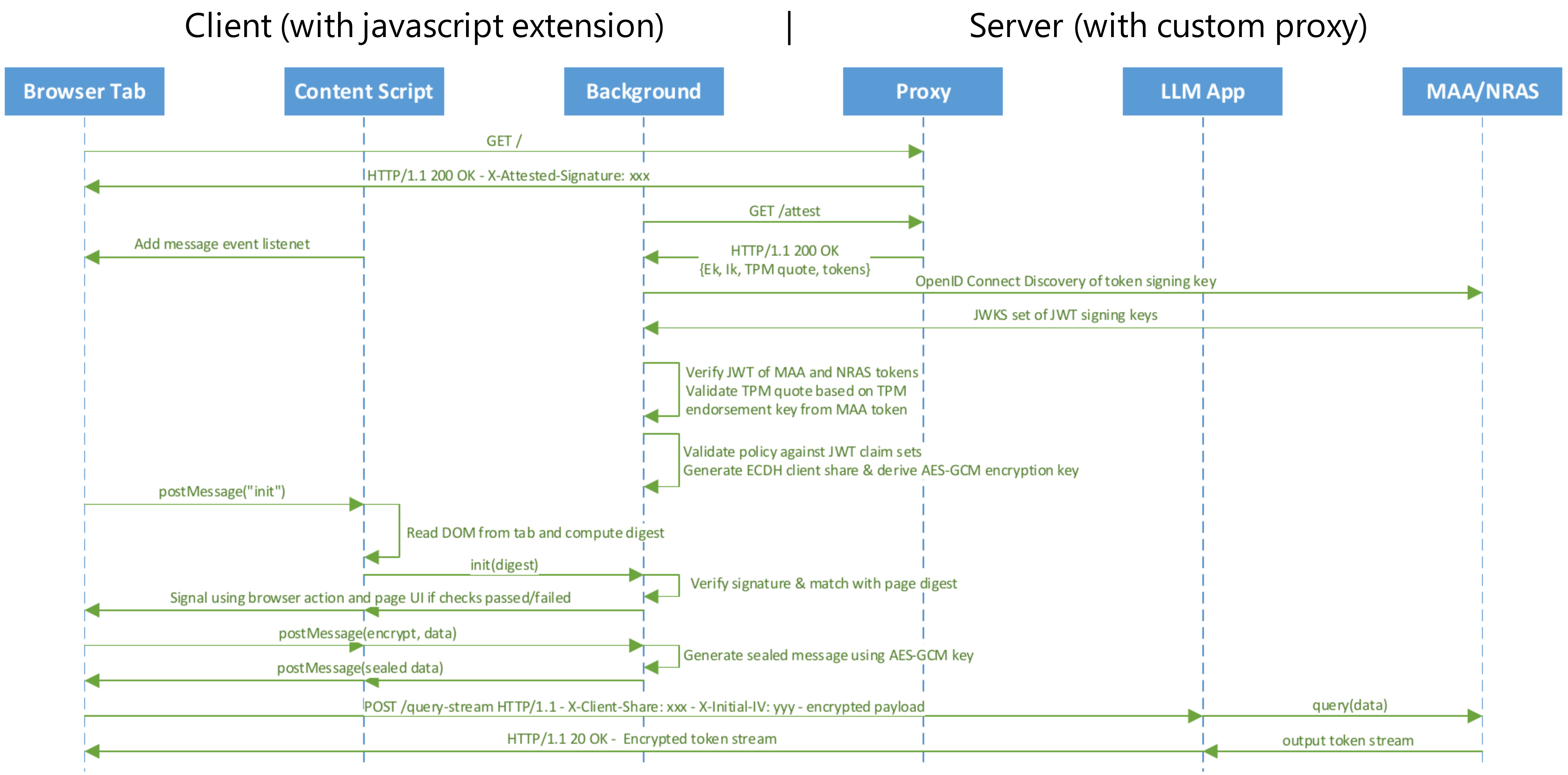
Field	Value
VBIOS	CSP installed version
Firmware	Tenant driver version
Nonce	Random + HCL Report digest

Demo: Sample Confidential Retrieval-Augmented Generation (RAG)



see also Mark Russinovich's demo

Application-level Attestation and Encryption Protocol


Client (with javascript extension)
|
Server (with custom proxy)
Browser Tab
Content Script
Background
Proxy
LLM App
MAA/NRAS
GET /
HTTP/1.1 200 OK - X-Attested-Signature: xxx
GET /attest
Add message event listenet
HTTP/1.1 200 OK
{Ek, Ik, TPM quote, tokens}
OpenID Connect Discovery of token signing key
JWKS set of JWT signing keys
Verify JWT of MAA and NRAS tokens
Validate TPM quote based on TPM
endorsement key from MAA token
Validate policy against JWT claim sets
Generate ECDH client share & derive AES-GCM encryption key
postMessage("init")
Read DOM from tab and compute digest
init(digest)
Verify signature & match with page digest
Signal using browser action and page UI if checks passed/failed
postMessage(encrypt, data)
Generate sealed message using AES-GCM key
postMessage(sealed data)
POST /query-stream HTTP/1.1 - X-Client-Share: xxx - X-Initial-IV: yyy - encrypted payload
query(data)
HTTP/1.1 20 OK - Encrypted token stream
output token stream

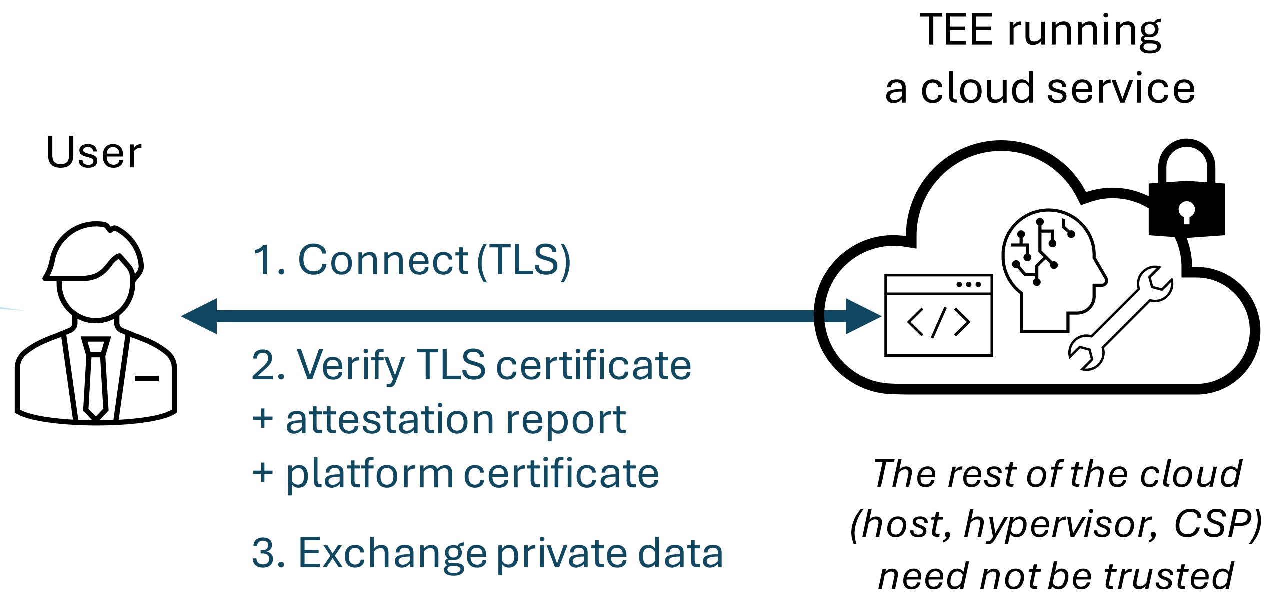
Why Should I Trust Your Code?

Transparent Updates for Confidential Computing

See also Why Should I Trust Your Code? – CACM

The Attested Code Update Problem

Which code hash should
I trust for this service ??



Cloud services are frequently updated, to add functionality, fix bugs, or patch CVEs.

- Code reviews take time & effort, and they are not perfect.
- Most users can't review source updates and rebuild attested binaries
- Most service providers can't wait
- The "attested TCB" for the service includes code from multiple providers (firmware, system, runtime, apps, libraries, containers) which all require authorization & updating

Transparency: Core Intuition

We cannot stop supply chain actors from making false claims, but we can make them accountable by requiring all claims be registered in verifiable **transparency ledgers**.

This ensures that malicious actors that make contradictory claims to different entities (customers, auditors, regulators) can be detected and held accountable.

Examples of Transparency Systems

Certificate Transparency [RRC 6962] Adam Langley, Emilia Kasper, Ben Laurie (Google)

CONIKS: bringing key transparency to end users , M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman (USENIX Security'15).

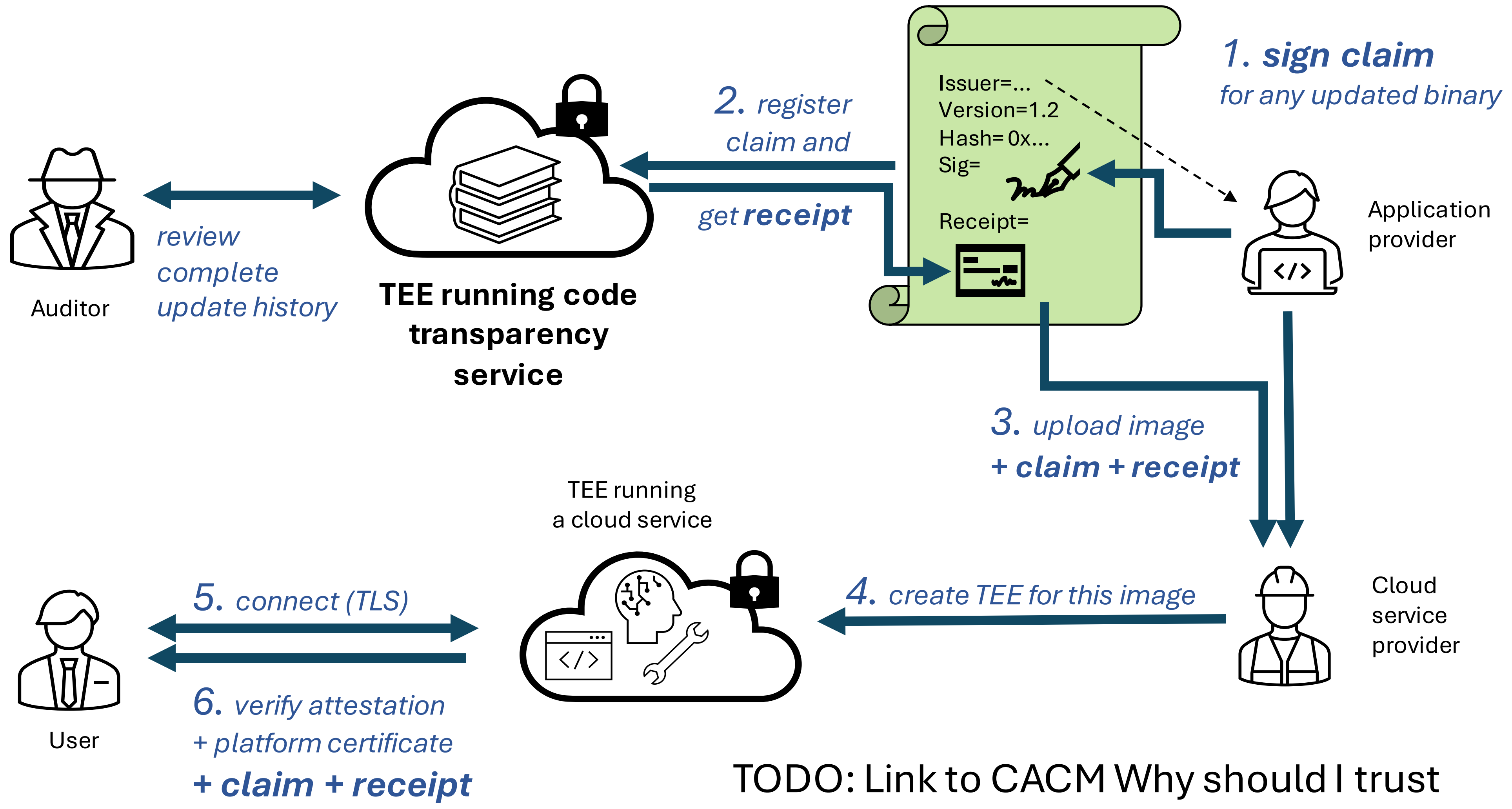
Keeping authorities "honest or bust" based on large-scale decentralized witness cosigning (IEEE S&P '16)

CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds (Usenix'17, EPFL)

Contour: A practical system for binary transparency logging on bitcoin the latest authorized binary version.
M. Al-Bassam, S. Meiklejohn (Data Privacy Management, Cryptocurrencies and Blockchain Technology, 2018).

Proposal:

A Transparency Service (TS) for Attested Code


1. sign claim
for any updated binary
Issuer=...
Version=1.2
Hash= 0x...
Sig=
Receipt=
2. register claim and
get receipt
Application provider
Auditor
review complete update history
TEE running code transparency service
3. upload image + claim + receipt
TEE running a cloud service
4. create TEE for this image
Cloud service provider
5. connect (TLS)
User
6. verify attestation + platform certificate + claim + receipt

TODO: Link to CACM Why should I trust your code. Link to CCF paper.

Proposal:

A Transparency Service (TS) for Attested Code

Building blocks:

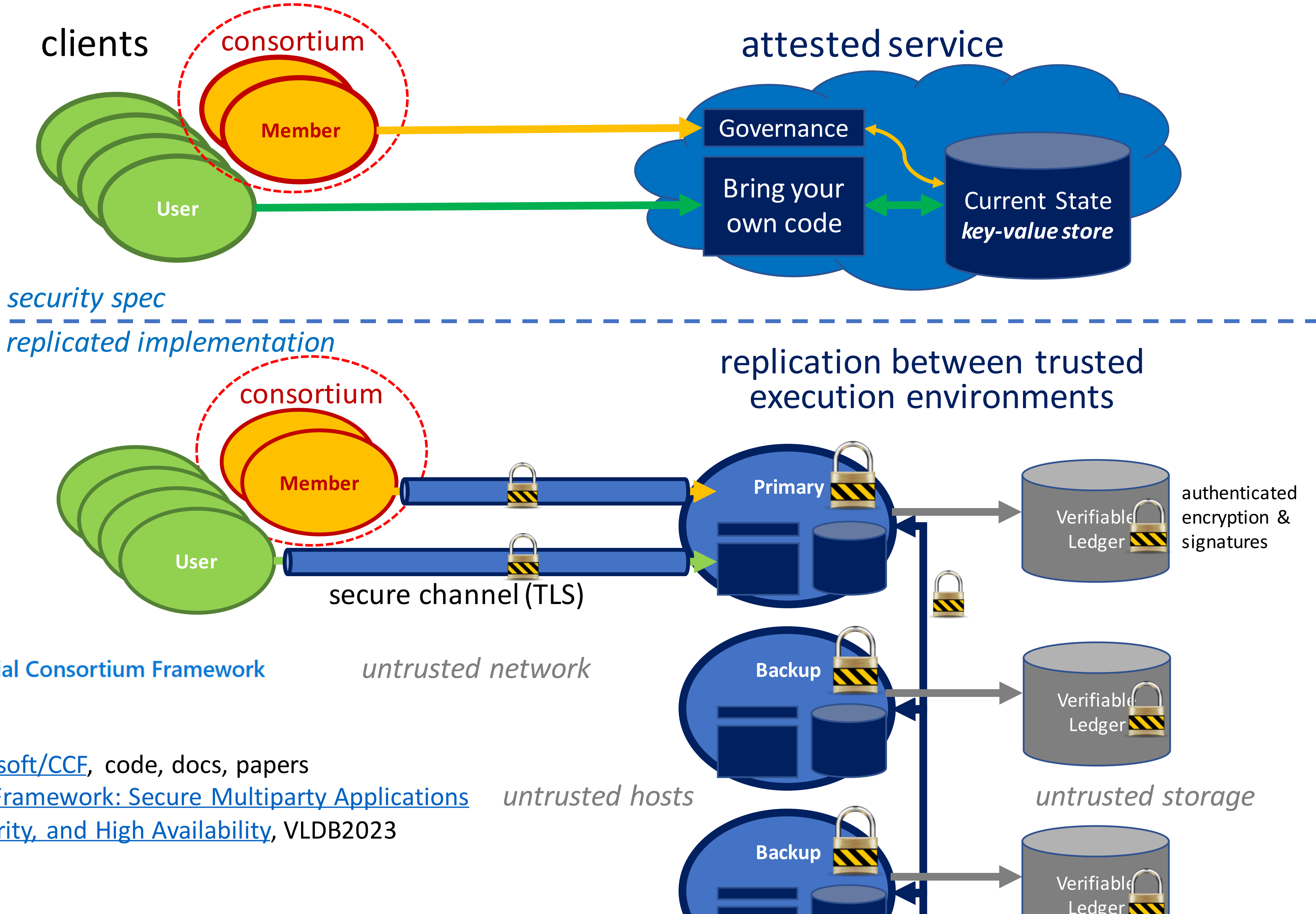
1. IETF SCITT architecture for transparent claims & protocols
2. CCF as an attested transparency service & append-only log
3. Delegation policies and confidential containers
   to automate our software supply chain for confidential services
   (source-code release, build, deploy)

Supply Chain Integrity, Transparency, and Trust (SCITT)

- Interoperable transparency support for recording statements in (generic) supply chains
- Claim formats (CBOR)
  - standardized headers
  - standardized proofs of registration (receipts)
  - opaque payloads
- Issuer identification and signing (COSE)

```
             Artifact
                |
                v                      +---------------+
Issuer ->   Statement    Envelope      | DID Document |
                \           /          +---------------+
                 \         /              |     |
                  \ ______/               |     |
                      |                   |     |
                      v        Signature  |     |
                    Claim  <--------------/     |
                      |                         |
                      |   Receipt   +---------+ |
Transparency ->       +-------------| Ledger |  /
   Service            |             +---------+ X
                      v                       / \
                 Transparent                 /   \
                    Claim                   /    |
                      |\                   /     |
                      | \                 /      |
                      |  \               /       |
Verifier    ->        |    Verify Claim          |
                      |                          |
Auditor    ->     Collect Receipts        Replay Ledger
```

https://datatracker.ietf.org/group/scitt/



CCF Confidential Consortium Framework

see also:

https://github.com/Microsoft/CCF, code, docs, papers

Confidential Consortium Framework: Secure Multiparty Applications with Confidentiality, Integrity, and High Availability, VLDB2023

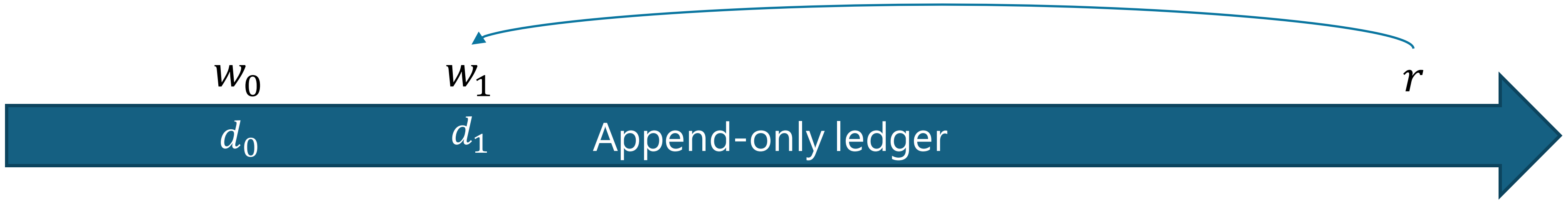
Receipts: Proofs of Registration & Freshness

A claim $k = (issuer, feed) \mapsto d$
may have been registered 6 months ago…

…and still be the latest
for this k, as of yesterday

A **write receipt**
proves that
$T[k] \leftarrow d_1$
at index w

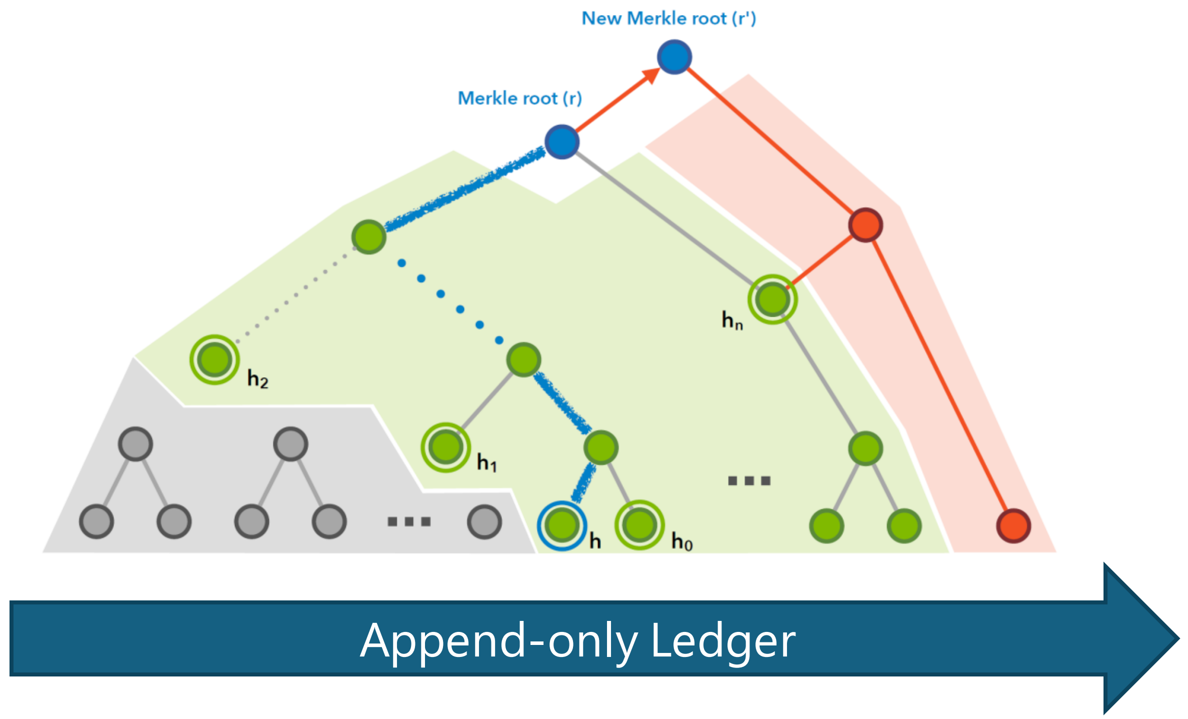
A **read receipt**
proves that
$T[k] == d_1$
at index r



prior writes to k

latest state

Receipts: Proofs of Registration & Freshness



Writes receipts are implemented by signing the root of the plain binary Merkle tree over the whole ledger contents

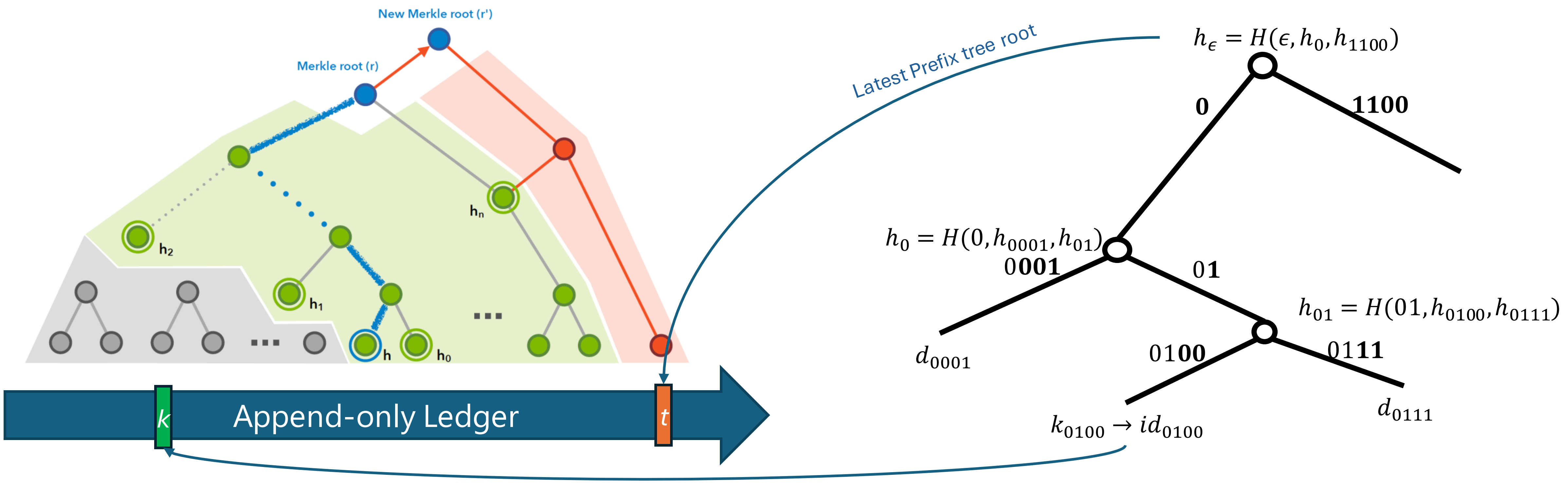
They can be issued efficiently:

- One hash per transaction
- One signature per transaction batch

The signing key is supported by attestation reports and governance transactions, also recorded in the ledger

```
COSE_CounterSignature = {
    "serviceId"     => bstr            ; Hash of public key of CCF service
    "transactionId" => tstr            ; CCF transaction id
    "alg" => int                       ; Signature algorithm
    "signature" => bstr                ; Signature over tree root
    "proof" => [+ ProofElement]        ; Intermediate hashes (Merkle path)
}
```

Receipts: Proofs of Registration & Freshness



Read receipts are implemented using a separate prefix tree (indexed by issuer + feed) pointing to the latest write index.

The prefix tree root is frequently timestamped and committed to the ledger

Read receipts can be attached to a Write receipt for that index.
Read receipts can be efficiently refreshed from the ledger.

Registration Policies

Receipt verification ensures the transparency service
has successfully applied all (transparent) registration policies:

- Policies can prevent many common supply chain attacks
  (by verifying identifiers, signatures, release tags, version numbers,...)
- Policies can ensure that sufficient metadata is recorded
  to enable independent auditing against more advanced attacks,
  and thus deter/blame bad actors.

Simple policies are directly enforced by the transparency service

- In our prototype, scripted verification of crypto evidence
  (certificate, signatures, receipts, attestations)

Advanced policies are enforced by custom TEEs
(themselves verifiable using simple policies)

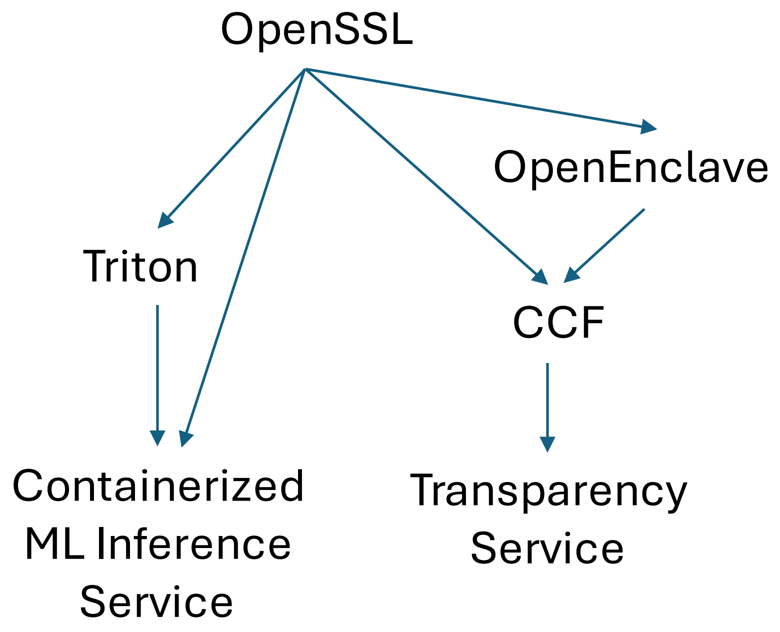
- In our prototype, containers for source release,
  for building binary packages, containers, and enclaves

Transparent Attested Build

1. Building a sample confidential ML inference service based on Triton
2. Bootstrapping our transparency service

Large complex build steps *but* making them transparent and attested only requires superficial changes (a few lines in scripts and dockerfiles)

Open-Source Projects & Dependencies



Project	LOC	Output Image (GB)	Layers	Build time (sec)			URLs	Output claim size (KB)
				Baseline	Proxy	TEE		
OpenSSL	24	-	-	464	618	736	155	34
OpenEnclave-base	16	2.01	7	387	678	1046	514	98
OpenEnclave	32	2.20	9	1870	2046	2390	81	23
CCF	43	4.39	14	2901	2911	3310	545	117
TS	29	0.54	15	115	200	293	66	11
Triton	16	1.78	15	1260	1432	1744	572	128

Principled Side-channel Protection

Boris Koepf, Stavros Volos,
Oleksii Oleksenko, Jana Hofmann,
Cédric Fournet

See also Project Venice for papers, details, etc

Isolation is core to Confidential Computing

- Smaller, delimited TCBs
- End-to-end encryption, during computation: **no direct leakage**

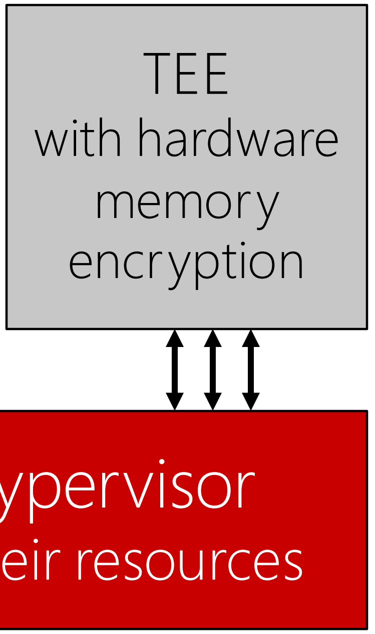
Sharing is core to Cloud Computing

- Amortizing cores, caches, buffers, memory, disk, network across many tenant workloads.
- Example: Growing core count sharing large expensive DRAM

Side-channels are an essential issue

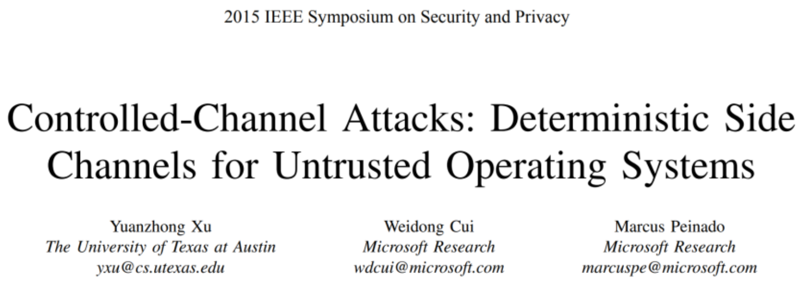
- Largely ignored in early hardware implementations
- Largely exploited in attacks papers (in the lab)

Controlled software channels



The cloud provider is a powerful attacker that allocates all resources and observes their use at a fine granularity

- Initial attacks targeted jpeg and spellchecking libraries, by invalidating code pages to infer data-dependent control flows
- Recent attacks exploit one-stepping and zero-stepping of target TEE.



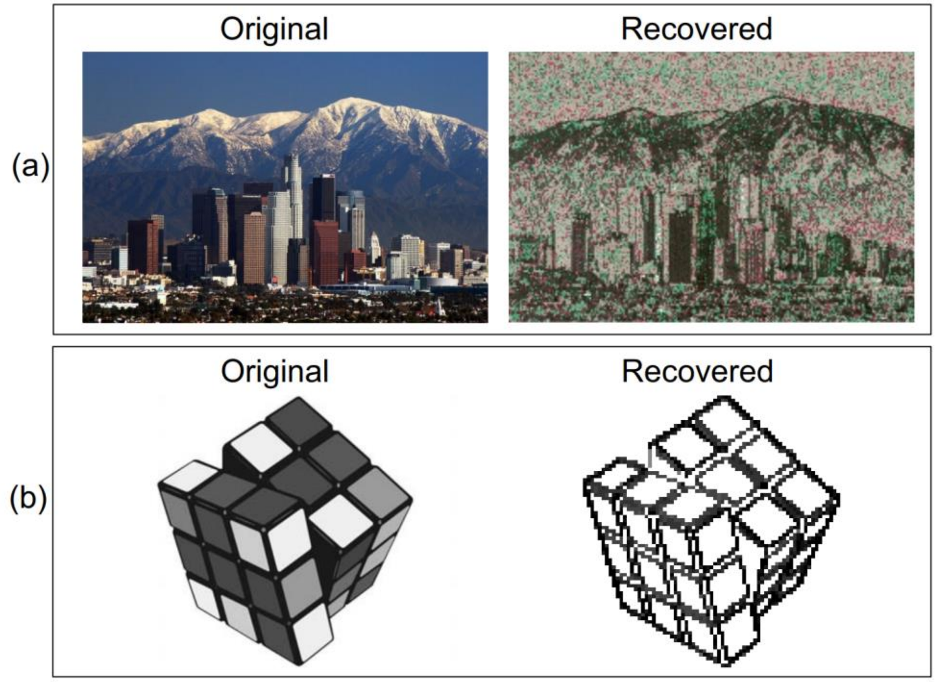
2015 IEEE Symposium on Security and Privacy

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems

Yuanzhong Xu
The University of Texas at Austin
yxu@cs.utexas.edu

Weidong Cui
Microsoft Research
wdcui@microsoft.com

Marcus Peinado
Microsoft Research
marcuspe@microsoft.com



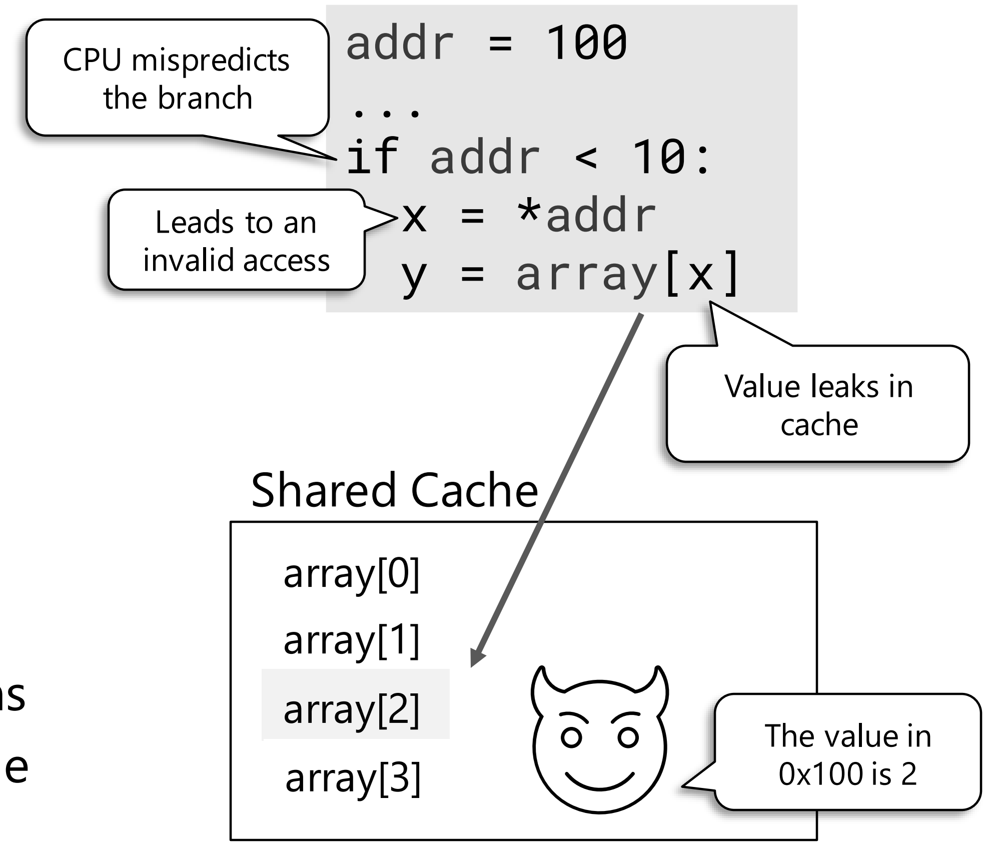
Microarchitectural Side-channels

Cache side-channels:

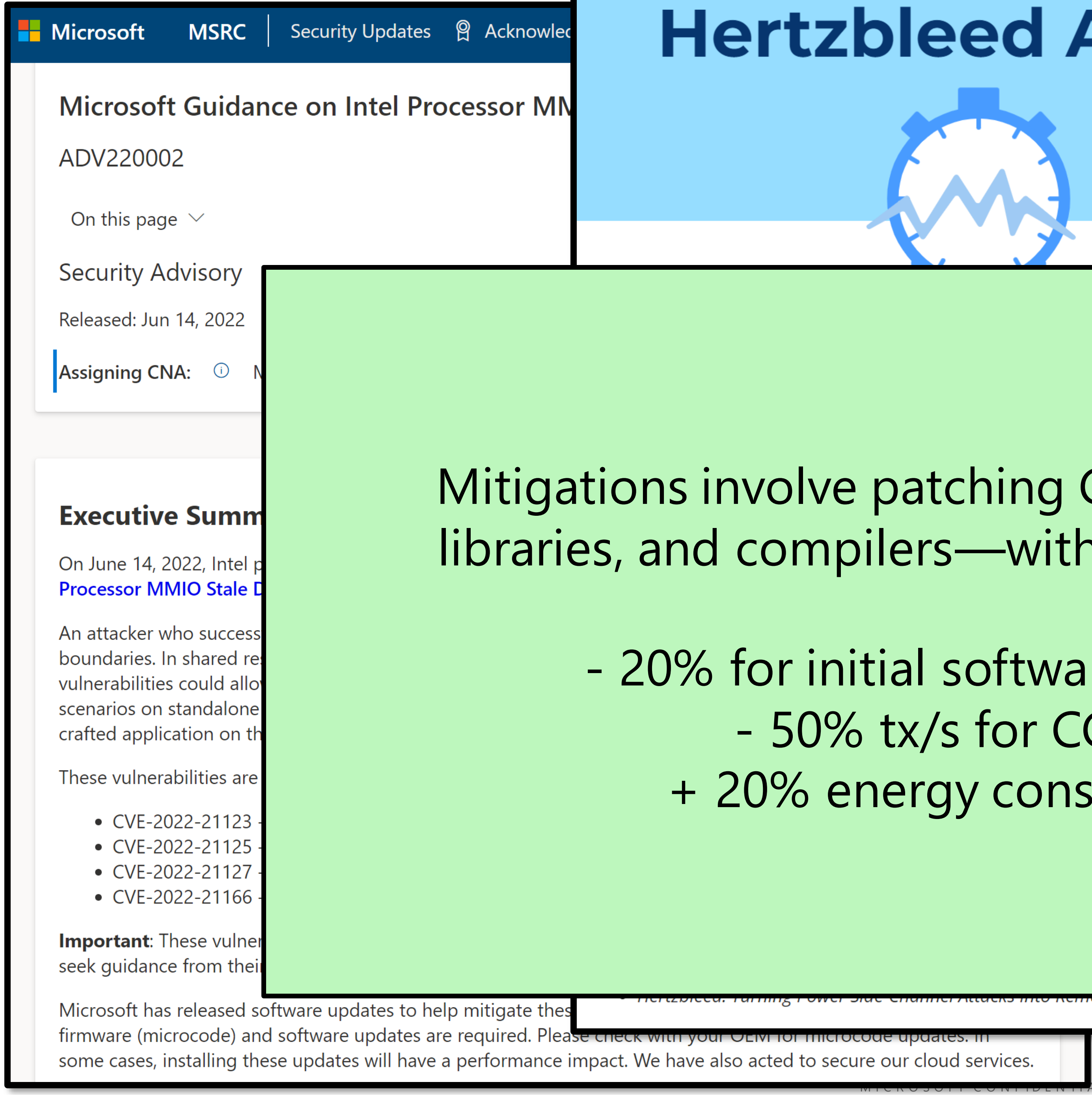
- Cache state depends on program secrets
- Attacker can observe cache changes, and thus infer the secrets

Speculative side-channel attacks:

- Speculative execution can violate security assumptions (e.g., bypass bounds checks)
- Leaks speculatively results via uArch state
- Depends on details of proprietary hardware implementations
- Many practical attacks, sometimes irrespective of target code



```
addr = 100
...
if addr < 10:
  x = *addr
  y = array[x]
```



Microsoft Guidance on Intel Processor MM

ADV220002

On this page

Security Advisory

Released: Jun 14, 2022

Assigning CNA: M

Executive Summ

On June 14, 2022, Intel p
Processor MMIO Stale D

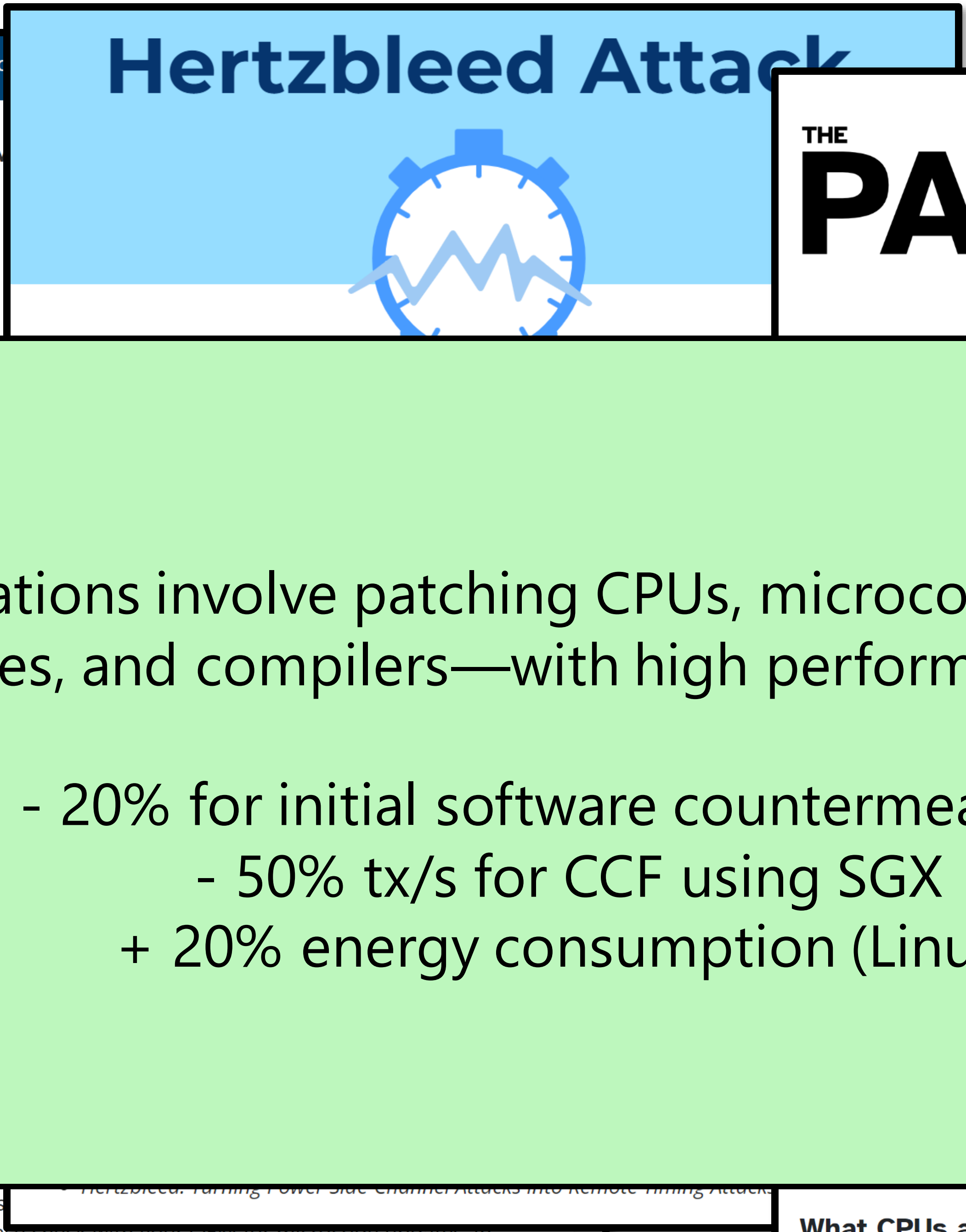
An attacker who success
boundaries. In shared re
vulnerabilities could allo
scenarios on standalone
crafted application on th

These vulnerabilities are

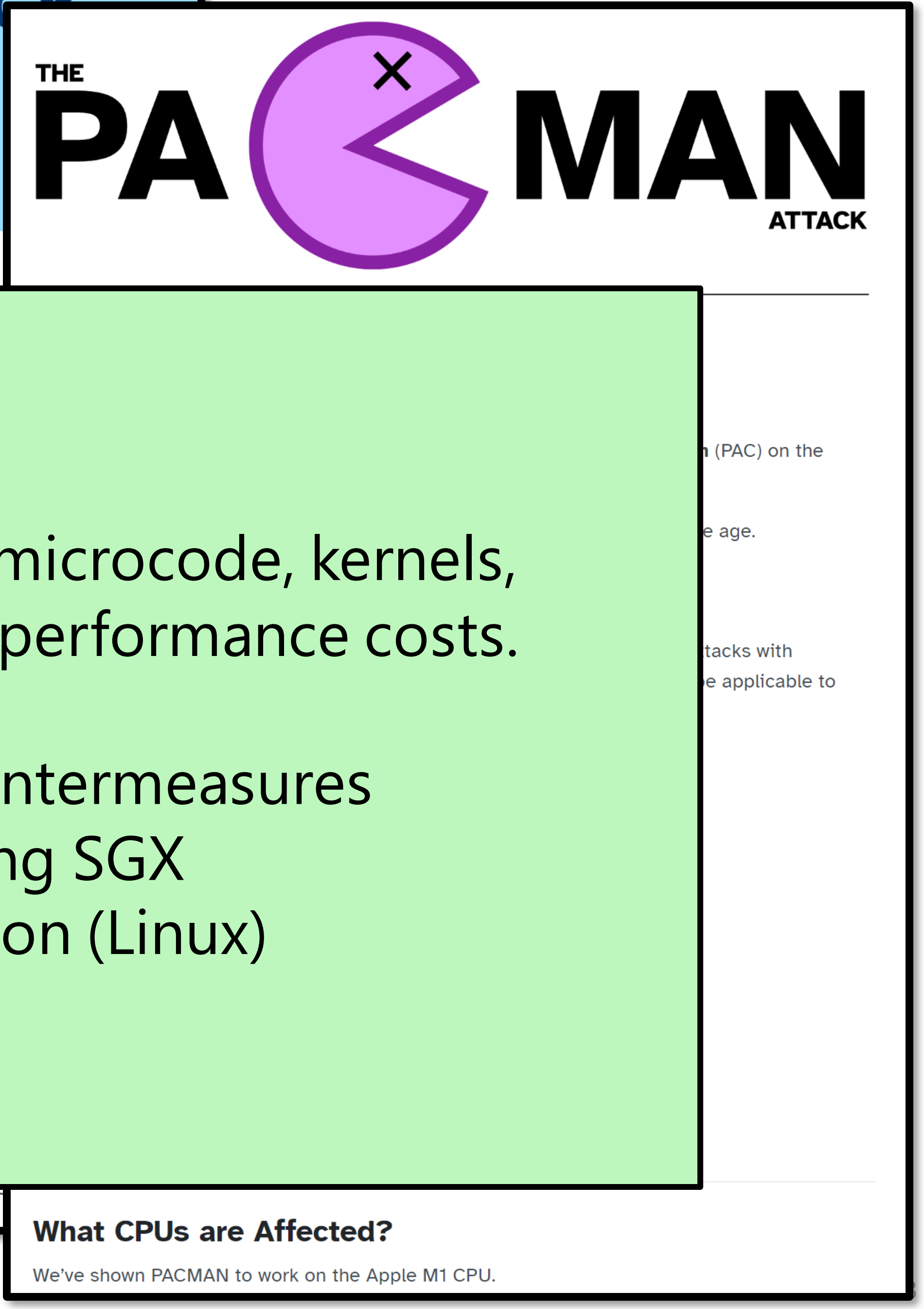
- CVE-2022-21123
- CVE-2022-21125
- CVE-2022-21127
- CVE-2022-21166

Important: These vulne
seek guidance from thei

Microsoft has released software updates to help mitigate thes
firmware (microcode) and software updates are required. Please check with your OEM for microcode updates. In
some cases, installing these updates will have a performance impact. We have also acted to secure our cloud services.



- Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks



n (PAC) on the

e age.

tacks with
e applicable to

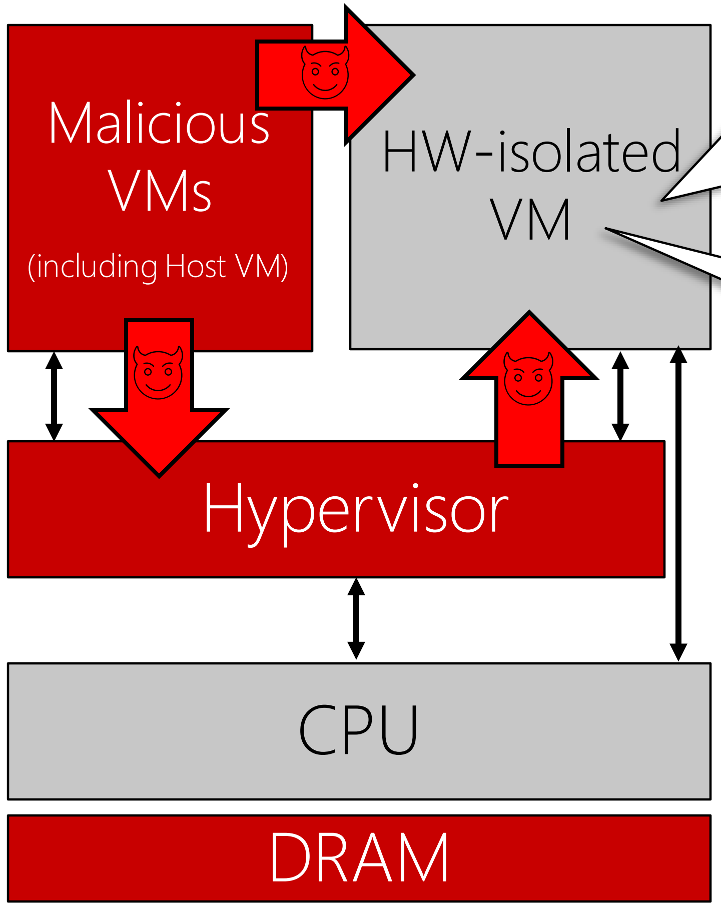
What CPUs are Affected?

We've shown PACMAN to work on the Apple M1 CPU.

Mitigations involve patching CPUs, microcode, kernels, libraries, and compilers—with high performance costs.

- 20% for initial software countermeasures
- 50% tx/s for CCF using SGX
+ 20% energy consumption (Linux)

Threat Model (CC)

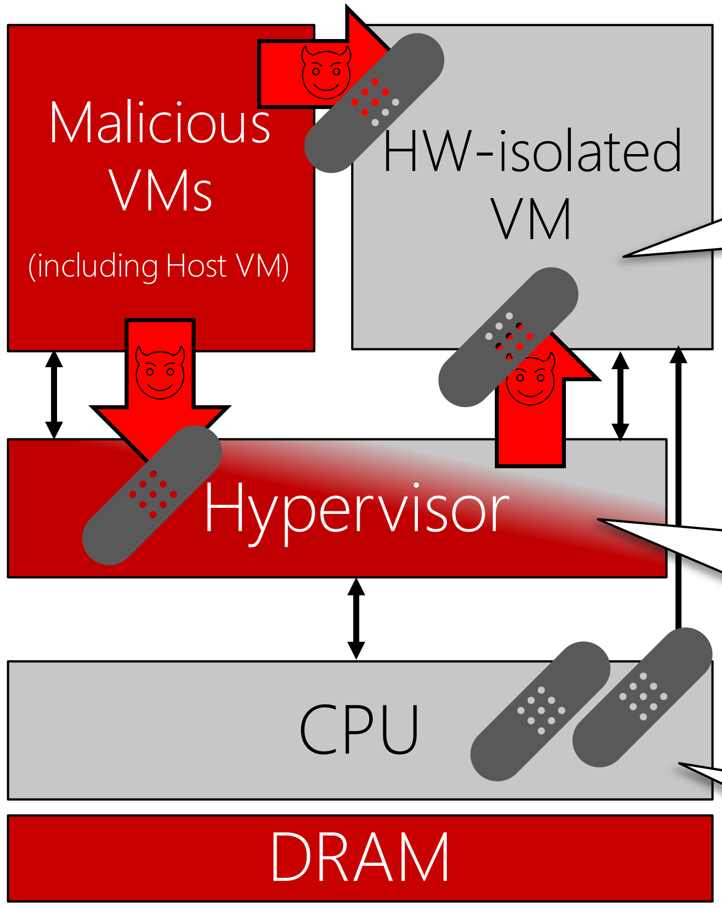


Can we protect confidential workloads from side-channels?

Can we convince tenants that we do?

Side-Channels Today:
Ad Hoc Attacks & Countermeasures

many attack-specific patches across all abstraction levels

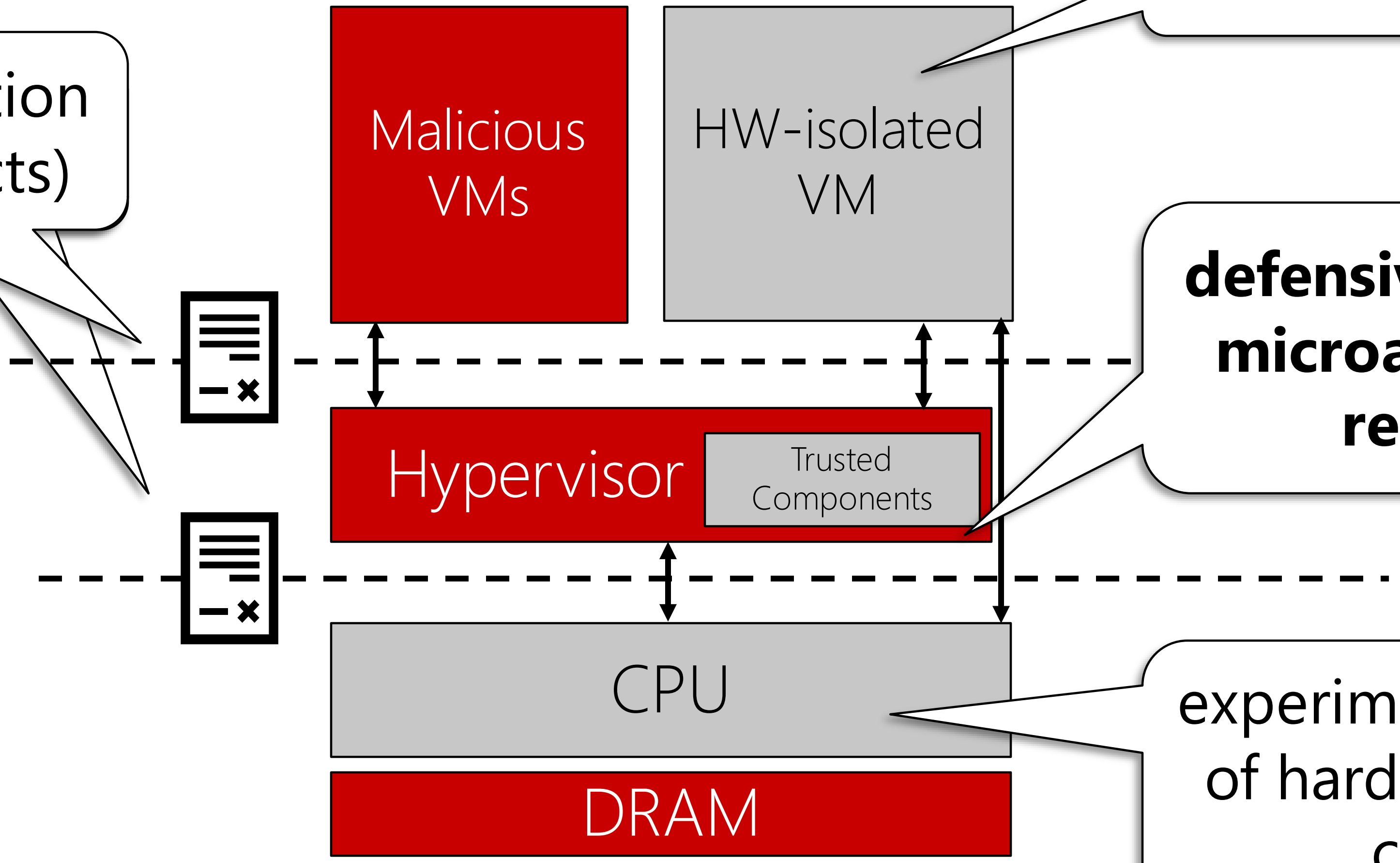


Except for selected libraries (crypto), we'd rather not change application code.

How to share resources? Fine-grained sharing aggravates attacks: hyperthreading, deduplication.

Any unknown microarchitectural details we should worry about?

Project Venice (Ongoing)



robust abstraction layers (contracts)

unmodified* use cases

defensive sharing of microarchitectural resources

experimental validation of hardware/software contracts

Microarchitectural (uArch) Isolation

Security Properties

1. **Spatial Isolation.** A VM is assigned resources whose uArch state cannot be observed or altered by other VMs.
2. **Temporal Isolation.** A VM is assigned resources whose initial uArch state does not depend on previous VMs and cannot be observed by future VMs.

Resources

- Core uArch (e.g., L1/L2 caches, TLBs)
  - Targeted by various Hyper-V defences, such as *Core Scheduling and HyperClear*
- Uncore uArch (e.g., L3 cache, directory for cache coherence)

How to jointly partition core and uncore resources?

Marghera: System Design for uArch Isolation

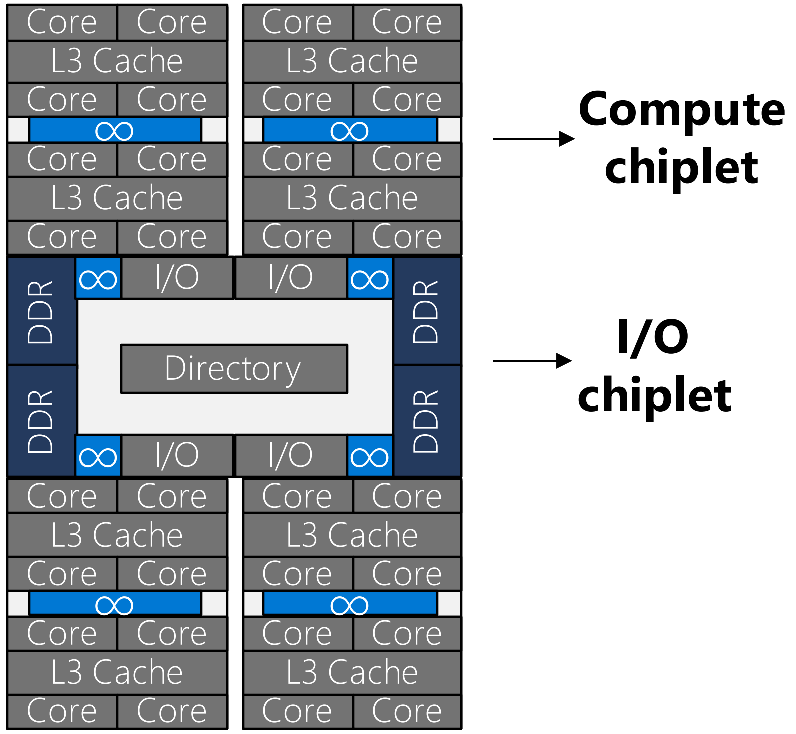
Memory manager

- Implements memory partitions via coloring
- Each color is exclusively assigned to one VM

Resource scheduler

- Implements compute partitions via chiplet scheduling

Chiplet-based Isolation on AMD Milan



Source: AMD Milan
(basis of Confidential Containers)

L3 cache is private to the chiplet's cores

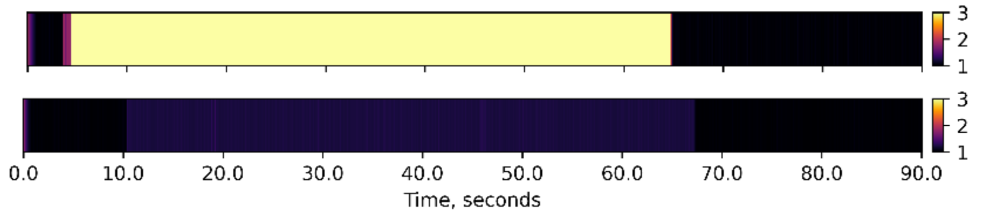
Cross-chiplet cache coherence via directory

Access to memory & I/O via shared I/O chiplet

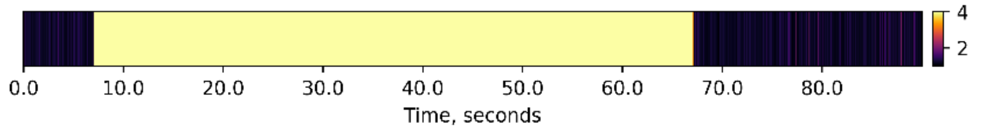
Chiplet-based Isolation: Leakage

L3 cache leakage

- Eliminated with chiplet-based scheduling



Cross-chiplet directory leakage



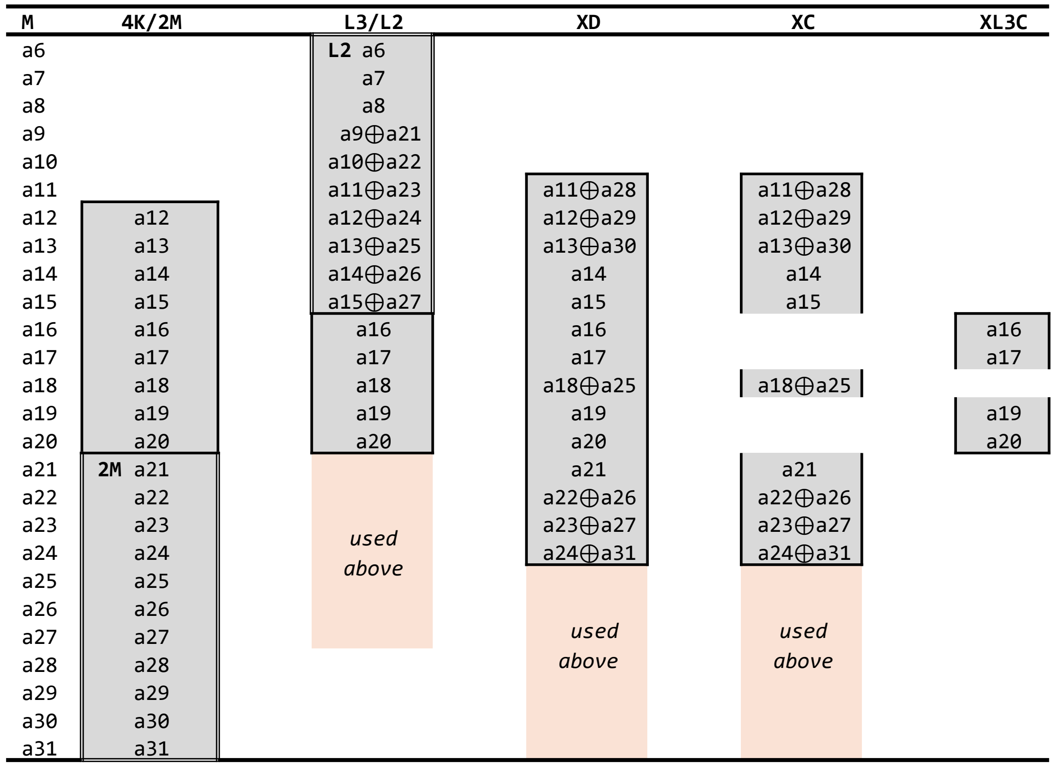
Memory Coloring for uArch Isolation

Challenges

- Identify indexing functions for all uArch resources
- Identify coloring function that simultaneously partitions *shared* resources, while not partitioning *private* resources

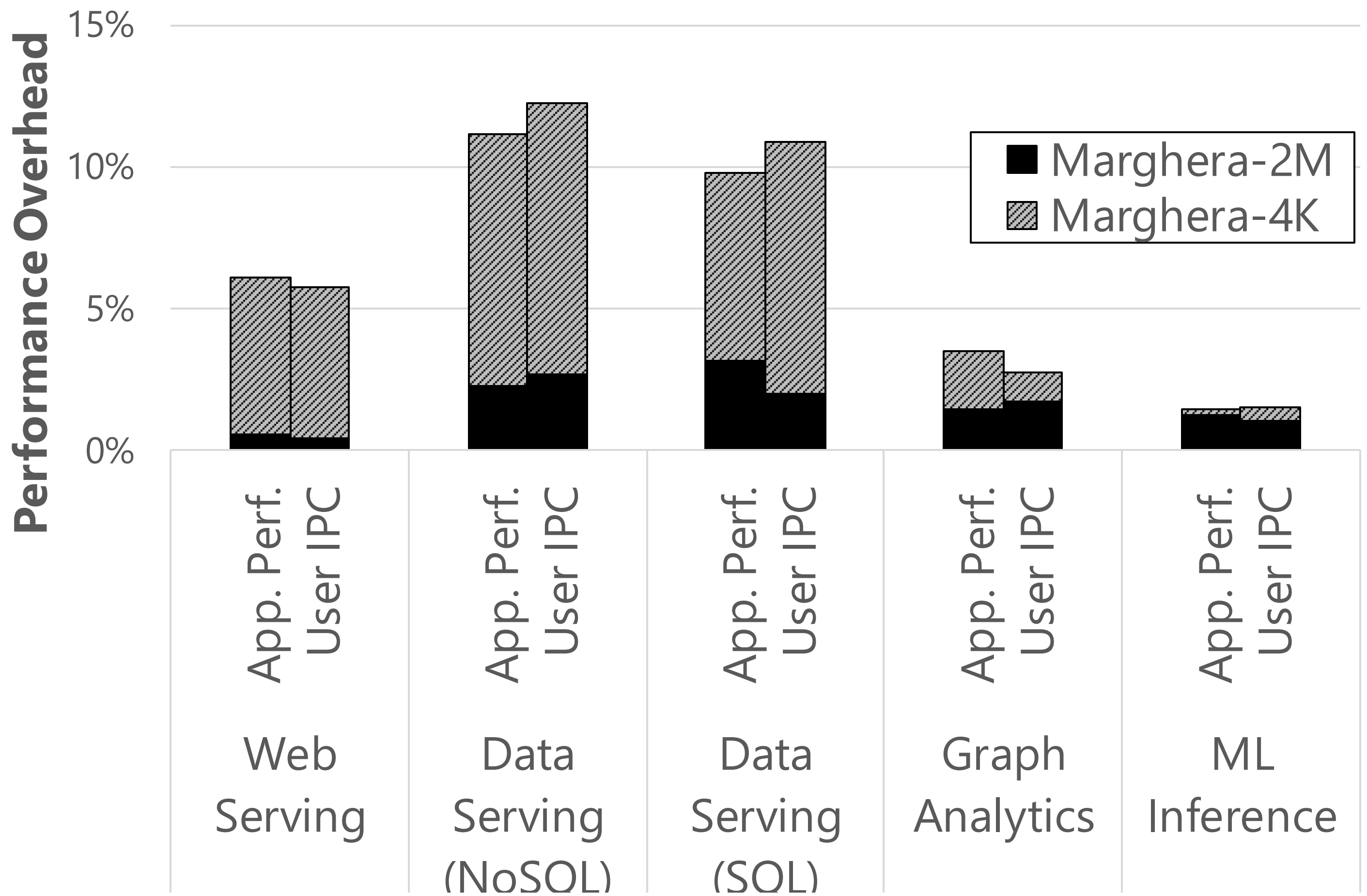
Algebraic tools for partitioning

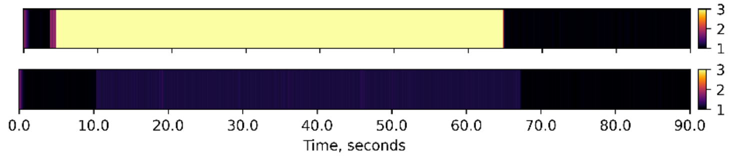
- Work for linear indexing functions (bits combined with XOR)
- Compose indexing functions to yield optimal trade-offs between security and performance



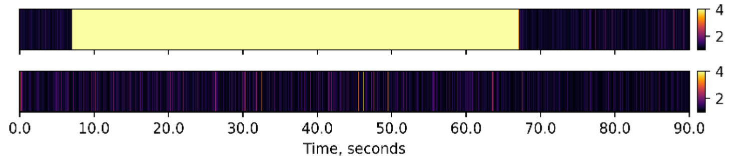
M	4K/2M	L3/L2	XD	XC	XL3C
a6		**L2** a6			
a7		a7			
a8		a8			
a9		a9⊕a21			
a10		a10⊕a22			
a11		a11⊕a23	a11⊕a28	a11⊕a28	
a12	a12	a12⊕a24	a12⊕a29	a12⊕a29	
a13	a13	a13⊕a25	a13⊕a30	a13⊕a30	
a14	a14	a14⊕a26	a14	a14	
a15	a15	a15⊕a27	a15	a15	
a16	a16	a16	a16		a16
a17	a17	a17	a17		a17
a18	a18	a18	a18⊕a25	a18⊕a25	
a19	a19	a19	a19		a19
a20	a20	a20	a20		a20
a21	**2M** a21		a21	a21	
a22	a22		a22⊕a26	a22⊕a26	
a23	a23	*used*	a23⊕a27	a23⊕a27	
a24	a24	*above*	a24⊕a31	a24⊕a31	
a25	a25				
a26	a26				
a27	a27		*used*	*used*	
a28	a28		*above*	*above*	
a29	a29				
a30	a30				
a31	a31				

Evaluation Highlights





L3 Cache



Cross-chiplet Directory

All identified microarchitectural side-channels are prevented with a small performance overhead (<3%)

Summary

Confidential computing lets users take control of their TCB

- Makes explicit the hardware, software, and services they need to trust
- Provides strong guarantees against the rest—even against the cloud provider.

Trusted Execution Environments will be pervasive in the cloud

- Concerted industry effort towards standardized capabilities.
- Ubiquitous hardware support makes them cheap (much like network/storage encryption)
- Defensive software (re)engineering is still required to reap all security benefits.

Many open issues:

- Application security (specs, safe programming, automated verification, auditing)
- Protocols for attestation, key-release, provisioning
- Transparency for hardware and software supply chains
- Side channels!

Microsoft Azure

We are hiring!

- 2-year Postdoc Researchers in Security and Privacy and Security and Systems
- Research interns

Azure Research