Arm Confidential Compute Architecture

Placing confidential compute in the hands of every developer

Mark Knight / Gareth Stockwell
Architecture & Technology Group, Arm
October 2021
Arm Confidential Compute Architecture

- Introduced as supplement to Armv9.2-A
- Driven by the expanding need to ensure privacy and security while harnessing data in ever more powerful ways
- Confidential Compute Architecture (Arm CCA) was announced in March 2021 and first specs publicly released in June 2021.
- Arm CCA designed to protect data and code wherever computing happens
- Protects data in-use by preventing privileged access to the resources, whilst retaining the right to manage them
Confidential Computing is the protection of data \textit{in use}, by performing computation in a hardware-based secure environment, to shield portions of code and data from access or modification, even from privileged software.
Realms: our vision for confidential compute for every workload

- Offer highly trusted execution environments, "Realms", in all markets, which can be used by mainstream workloads (apps on operating systems), developed and deployed using standard development environments.
- Remove the need for software workloads to trust their data to a host OS, Hypervisor, or Trusted OS.
- Support attestation so that every owner of a workload can verify trust in the platform or device.
- Democratize secure compute – so every developer can take advantage of it.

Developers can deploy Arm CCA to any device
Overview of Arm CCA

Realm

- Confidential compute for 3rd parties

Normal

- Hypervisor-based security, owns resources and scheduling

Secure

- TrustZone with dynamic memory 1st party use cases

- TA
- TA
- TOS
- Secure Partition

- Secure Device VM

Hypervisor

Secure Partition Manager

Root

Monitor

 Realm Physical Address Space

Non-Secure Physical Address Space

Secure Physical Address Space

0x000...

0xFFFF....
## Arm CCA threat model

### Confidentiality

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypervisor/Kernel/Secure world reads private Realm memory or register state</td>
<td>Arch</td>
</tr>
<tr>
<td>Device DMA reads private Realm memory or register state</td>
<td>Arch</td>
</tr>
</tbody>
</table>

### Integrity

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypervisor/Kernel/Secure world modifies private Realm memory or register state</td>
<td>Arch</td>
</tr>
<tr>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td>• Modify saved context</td>
<td></td>
</tr>
<tr>
<td>• Writing to Realm pages</td>
<td></td>
</tr>
<tr>
<td>• Memory remapping or aliasing</td>
<td></td>
</tr>
<tr>
<td>Device DMA modifies private Realm memory or register state</td>
<td>Arch</td>
</tr>
</tbody>
</table>

### Availability

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of service to a Realm – it is scheduled by OS/Hyp</td>
<td>Arch</td>
</tr>
<tr>
<td>Realm mounts a DoS attack on the hypervisor</td>
<td>Arch</td>
</tr>
</tbody>
</table>

### Indirect SW attacks

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known SW error injection – E.g.: Rowhammer, CLKSCREW</td>
<td>Arch, Realm</td>
</tr>
<tr>
<td>Known side channels E.g.: Spectre / Meltdown</td>
<td>Arch, Realm</td>
</tr>
</tbody>
</table>

### Direct HW attacks

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical DRAM probe and replay</td>
<td>HW</td>
</tr>
</tbody>
</table>

### Indirect SW attacks

- **Mitigated by**
  - Arm CCA (processor/SMMU/system and FW)
  - Mitigation requires additional HW
  - SW in the Realm has the tools to protect itself
  - Not mitigated
TrustZone in Armv8.4-A

Two security states: Secure, Non-Secure

Two isolation mechanisms:

1. TrustZone isolation boundaries prevent Non-Secure state from accessing Secure physical addresses

2. Page tables - stage 2 isolation boundaries

<table>
<thead>
<tr>
<th>Security State/PA space</th>
<th>Non-Secure PA</th>
<th>Secure PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-secure</td>
<td>Allow</td>
<td>Block</td>
</tr>
<tr>
<td>Secure</td>
<td>Allow</td>
<td>Allow</td>
</tr>
</tbody>
</table>
Arm CCA hardware features
Two new hardware features in the Realm Management Extension (RME)

New isolation boundaries for 3rd party confidential computing

• Realms: New type of protected execution environment

• Data and/or code are protected from any other execution environments:
  • Hypervisors
  • OS Kernels
  • Other Realms
  • Even TrustZone

Dynamic assignment of memory to physical address spaces / worlds

• Supports Realms AND adds dynamic memory support to TrustZone

  • Arm dynamic TrustZone technology
  • Removes boot-time static memory carve-outs
Arm CCA hardware architecture

RME adds another two security states and associated physical address spaces

- **Realm**: A new mutually distrusting space for confidential compute
- **Root**: The Monitor gets its own private address space

HW isolation between address spaces is managed through a new Granule Protection Table (GPT), an extension of MMU page tables that is controlled by the Monitor in EL3

- Invalid accesses raise page faults
Arm CCA is a combination of hardware and firmware

- Use of firmware (RMM and Monitor) simplifies hardware and increases transparency
- Firmware components manage isolation hardware
- The Monitor controls isolation between worlds by programming the Granule Protection Table
- The Realm Management Monitor (RMM)
  - Manages Realm-to-Realm protection using stage 2 page tables
  - Enables Host to manage Realms (create; destroy; schedule; add / remove memory) via the Realm Management Interface (RMI)
  - Enables Realm to request attestation report via the Realm Services Interface (RSI)
- Arm will provide reference implementations of Monitor and RMM
Arm CCA supports attestation

Assuring data is protected; assuring that data and transactions can be trusted
Hardware-backed isolation for all workloads

Mobile / PC / DTV
- Social Media
- Email
- Browsing
- Content Protection
- OS Services
- Enterprise Apps
- Health / Fitness Apps

IoT
- Machine Learning
- Ambient Compute

Cloud
- Tenant 1
- Tenant n

Edge Compute
- Machine Learning
- Multi-Vendor Environment

Automotive
- OEM Apps
- Personal Data
- Autonomous Driving
- Content Protection

High end Wearables
- Health
- SmartKey
- Fitness
- Medical
Arm CCA – developer resources available now

See https://developer.arm.com/armcca

- **Tools**
  - Register XML
  - AC6 EAC asm/disasm support for RME complete
  - GNU Binutils available and upstreaming complete

- **Reference manual supplements**
  - RME Architecture (ARM DDI 0615A.a)
  - SMMU for RME (ARM IHI 0094A.a)
  - MPAM (ARM DDI 0598C.a)

- **Platform design documents released June 2021**
  - Security Model (DEN0096)
  - RME System Architecture (DEN0129)

- **Guides**
  - Overview of the Arm CCA (DEN0125)
  - Arm Realm Management Extension (DEN0126)
  - Arm Confidential Compute Software Stack (DEN0127)

- **Open Source Implementations**
  - TF-A Monitor code branch released June
  - Project Veraison: Software for attestation services

- **FVP model with RME support is available**
- **Blogs on Arm CCA and dynamic TrustZone Technology**
- **Videos from June Linaro event**
  - [https://connect.linaro.org/resources/arm-cca/](https://connect.linaro.org/resources/arm-cca/)
Arm CCA – what’s next
Architecture updates planned for 2022

Software Architecture
- Realm Management Monitor specification under development

Device Assignment
- Device assignment will allow hardware devices to be mapped to Realms
- Focused on the use of accelerators from Realms
- Dependency on an SMMU
- No device may access the memory of a Realm unless the Realm has accepted the device

Memory Encryption Contexts
- Per-Realm encryption key (or tweak)
- Mitigates physical memory replay attacks between Realms
- Defence in depth against attacks on Realm stage-2 translation
Summary

• Security at scale for all classes of device and all workloads – Confidential Compute everywhere
• Simplifying and democratizing the process for every developer
• Continually improving security – Arm CCA builds on earlier innovations in isolation technology
• Hardware specs released publicly June 2021, software specs being developed for publication in 2022
• Start of a journey with the software and toolchain communities
Thank You
Danke
Gracias
謝謝
ありがとうございました
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה
Kiitos