

# 2023-03-28

Created: 2023-11-08 12:40:14

## Chats

- OP-TEE.0327.md
  - [09:12] What is OP-TEE? (OP-TEE.0327.md)
  - [09:13] What is the prefix OP of OP-TEE? (OP-TEE.0327.md)
  - [09:13] Are there alternative technologies of OP-TEE? (OP-TEE.0327.md)
  - [09:14] Does OP-TEE a software or a hardware solution? (OP-TEE.0327.md)
  - [09:15] Is OP-TEE a firmware? (OP-TEE.0327.md)
  - [09:15] Are there successful cases of OP-TEE? (OP-TEE.0327.md)
  - [09:16] What are different between ARM OP-TEE and Intel SMM? Please also summarize in a table. (OP-TEE.0327.md)
  - [09:17] What is RPMB? (OP-TEE.0327.md)
  - [09:18] What is HS in OP-TEE? (OP-TEE.0327.md)
  - [09:19] What is KEK in OP-TEE? (OP-TEE.0327.md)
  - [13:45] Does OP-TEE support multi-thread? (OP-TEE.0327.md)
  - [13:51] What is secure context switching in TrustZone? (OP-TEE.0327.md)
  - [13:53] Does OP-TEE support multi-process? (OP-TEE.0327.md)
  - [13:56] Can secure kernel and trusted applications in OP-TEE can be multi-thread? (OP-TEE.0327.md)
  - [13:57] Does OP-TEE support IPC? (OP-TEE.0327.md)
  - [13:59] Does OP-TEE support shared memory? (OP-TEE.0327.md)
  - [14:03] Please summarize multi-process, IPC, and shared memory of OP-TEE. (OP-TEE.0327.md)
  - [14:06] Can secure world can access memory in normal world? (OP-TEE.0327.md)
  - [14:07] Can normal world access memory in secure world? (OP-TEE.0327.md)
  - [14:08] Does OP-TEE support network communication in secure world? (OP-TEE.0327.md)
  - [15:19] <https://github.com/OP-TEE> (OP-TEE.0327.md)
- TrustZone.0329.md
  - [15:30] <https://genode.org/documentation/articles/trustzone> (TrustZone.0329.md)
  - [15:31] Please summarize the following articles in short. (TrustZone.0329.md)
  - [15:32] An Exploration of ARM TrustZone Technology ... (TrustZone.0329.md)
  - [15:33] Starting point ... (TrustZone.0329.md)
  - [15:34] The design and genesis of our custom base-hw kernel platform ... (TrustZone.0329.md)
  - [15:37] Hypervisor managing the non-secure world ... (TrustZone.0329.md)
  - [15:38] World switch between non-secure world and secure world ... (TrustZone.0329.md)
  - [17:26] User-level Virtual Machine Monitor (VMM) ... (TrustZone.0329.md)
  - [17:26] Simple test kernel for the non-secure world ... (TrustZone.0329.md)
  - [17:27] Interrupts ... (TrustZone.0329.md)
  - [17:28] Booting Linux in the non-secure world ... (TrustZone.0329.md)
  - [17:28] Device emulation ... (TrustZone.0329.md)
  - [17:29] TrustZone demonstrated in color ... (TrustZone.0329.md)
  - [17:30] Additional device drivers ... (TrustZone.0329.md)
  - [17:30] Demonstration scenario ... (TrustZone.0329.md)
  - [17:31] Common questions, answered ... (TrustZone.0329.md)
  - [17:32] Does it make sense to run commodity Oses like Android in the secure world? ... (TrustZone.0329.md)

(TrustZone.0329.md)

- [17:37] Please compare TrustZone and TPM in a table. (TrustZone.0329.md)
- [17:40] What is bootstrap code? (TrustZone.0329.md)
- [17:43] What are two lines of experimentation? Please summarize them in short. (TrustZone.0329.md)
- [17:44] What are two lines in the section "Starting point" in the article? (TrustZone.0329.md)

---

## Q & A

---

### OP-TEE.0327.md

---

**Question:** What is OP-TEE?

**Answer:**

OP-TEE (Open Portable Trusted Execution Environment) is an open-source project that aims to provide a secure and trusted execution environment for software applications on devices. OP-TEE is primarily designed for Arm TrustZone technology, which is used in various processors to create a secure area (trusted execution environment) separate from the normal operating system. This separation helps protect sensitive data and code from unauthorized access or tampering.

OP-TEE is made up of several components, including:

1. Secure kernel: This is the core component of OP-TEE that runs in the secure world and manages trusted applications (TAs) and their execution.
2. Trusted applications: These are the secure applications that run within the secure world, providing security services to normal applications.
3. Client API: This is the interface that normal applications use to communicate with trusted applications.
4. TEE Internal Core API: This is the interface used by trusted applications to access TEE services provided by the secure kernel.

OP-TEE is widely used in industries like mobile devices, IoT, automotive, and more, where security is of paramount importance. The project is maintained by Linaro, a collaborative organization that focuses on improving open-source software for Arm-based platforms.

---

**Question:** What is the prefix OP of OP-TEE?

**Answer:**

The prefix "OP" in OP-TEE stands for "Open Portable." It signifies that the project is an open-source implementation of a Trusted Execution Environment (TEE) that can be portably used across various platforms, mainly built around Arm TrustZone technology. The name highlights the goals of providing both transparency and flexibility to developers and users who want to create secure applications and services on a wide range of devices.

---

**Question:** Are there alternative technologies of OP-TEE?

**Answer:**

Yes, there are alternative technologies to OP-TEE that provide trusted execution environments or similar security features. Some of them include:

1. Intel SGX (Software Guard Extensions): Intel SGX is a set of security-related instruction codes that are built into some Intel CPUs. It allows developers to create secure enclaves, which are isolated memory regions

into some Intel or ARM architecture developers to create secure enclaves, which are isolated memory regions that protect sensitive data and code from unauthorized access, even from the host operating system or a hypervisor.

2. AMD SEV (Secure Encrypted Virtualization): AMD SEV is a hardware feature found in some AMD processors that encrypts the memory of virtual machines to protect data from unauthorized access. It aims to provide a secure execution environment for applications running in virtualized environments.
3. ARM TrustZone-M: TrustZone-M is a security extension for ARM Cortex-M processors, targeting microcontroller devices used in IoT, industrial control, and other embedded systems. While OP-TEE targets TrustZone technology in application processors, TrustZone-M is tailored for microcontrollers.
4. Google Titan M: Titan M is a custom security chip developed by Google to provide a secure environment for storing sensitive data and performing cryptographic operations on its Pixel smartphones. It is designed to protect the integrity of the bootloader, verify the operating system, and secure user data.
5. Apple Secure Enclave: The Secure Enclave is a coprocessor found in Apple devices, such as iPhones and iPads, that provides a secure environment for cryptographic operations and storage of sensitive data like encryption keys, biometric information, and device passcodes.

These technologies may have different implementations, features, and target applications, but they all aim to enhance security by providing isolated execution environments and protecting sensitive data from unauthorized access.

---

**Question:** Does OP-TEE a software or a hardware solution?

**Answer:**

OP-TEE is primarily a software solution that leverages underlying hardware capabilities, specifically Arm TrustZone technology, to provide a secure and trusted execution environment for software applications on devices.

OP-TEE includes a secure kernel, trusted applications, and APIs that enable secure communication between the normal and secure worlds. While OP-TEE is a software implementation, it relies on the hardware-based security features provided by Arm TrustZone to create a separate, secure area for sensitive data and code execution.

In summary, OP-TEE is a software solution that takes advantage of hardware-based security technologies to achieve its goals of providing a secure environment for running trusted applications.

---

**Question:** Is OP-TEE a firmware?

**Answer:**

OP-TEE can be considered a firmware component, as it is a low-level software that interacts directly with the hardware and provides security services to the system. It serves as the trusted execution environment for devices using Arm TrustZone technology.

Firmware is a type of software that is typically stored in non-volatile memory and provides basic functionality or services to a device's hardware. In the case of OP-TEE, it provides the secure kernel, trusted applications, and APIs for managing and interacting with the secure environment.

OP-TEE is often integrated into the boot process of a device, alongside other firmware components like the bootloader, and it initializes the secure environment before the main operating system starts. This ensures that the secure services and trusted applications provided by OP-TEE are available to the system throughout its operation.

---

**Question:** Are there successful cases of OP-TEE?

**Answer:**

Yes, there have been successful use cases of OP-TEE across various industries and applications that require a secure and trusted execution environment. Some of these include:

- 1. Mobile devices: OP-TEE is used in mobile devices like smartphones and tablets to provide a secure environment for handling sensitive data, such as biometric information, cryptographic keys, and DRM (Digital Rights Management) protection.
- 2. IoT devices: Internet of Things (IoT) devices, such as smart home appliances, wearables, and industrial sensors, benefit from the security provided by OP-TEE. It enables these devices to handle sensitive data securely, execute cryptographic operations, and maintain the confidentiality and integrity of the data being transmitted.
- 3. Automotive systems: OP-TEE can be used in automotive systems to secure in-vehicle communication, protect sensitive data, and enable secure over-the-air updates.
- 4. Payment systems: OP-TEE can provide a secure environment for payment applications, ensuring the confidentiality and integrity of financial transactions and sensitive user data.
- 5. Healthcare devices: Medical devices and healthcare systems can utilize OP-TEE to securely store and process patient data, protecting it from unauthorized access and tampering.

While specific implementations may vary, these examples demonstrate the versatility and effectiveness of OP-TEE in providing a secure and trusted environment for a wide range of applications. The open-source nature of OP-TEE also makes it possible for developers and organizations to customize and adapt the software to fit their unique security requirements.

**Question:** What are different between ARM OP-TEE and Intel SMM? Please also summarize in a table.

**Answer:**

ARM OP-TEE and Intel SMM are different security technologies that provide secure and isolated execution environments for sensitive operations. Here's a comparison of the two:

- 1. ARM OP-TEE:
  - 2. Primarily designed for ARM-based platforms using TrustZone technology.
  - 3. Open-source implementation, maintained by Linaro.
  - 4. Provides a secure kernel, trusted applications, and APIs for communication between the normal and secure worlds.
  - 5. Focuses on creating a Trusted Execution Environment (TEE) for secure application execution.
  - 6. Targets a variety of industries, including mobile devices, IoT, automotive systems, and more.
- 7. Intel SMM (System Management Mode):
  - 8. Designed for Intel x86 processors.
  - 9. A proprietary implementation by Intel.
  - 10. Provides a separate CPU mode (SMM) with its own address space and execution environment.
  - 11. Focuses on handling low-level system management tasks, such as power management, hardware monitoring, and firmware updates.
  - 12. Primarily used for system management tasks on PCs, servers, and other x86-based systems.

Summary in a table:

Feature/Aspect	ARM OP-TEE	Intel SMM
----------------	------------	-----------

| Platform | ARM-based systems | Intel x86 processors | | Implementation | Open-source | Proprietary | | Primary Focus | Trusted Execution Environment (TEE) | System Management Tasks | | Typical Applications | Mobile, IoT, automotive, etc. | PCs, servers, x86-based systems | | Architecture | ARM TrustZone | System Management Mode |

Please note that while both technologies provide a secure and isolated execution environment, they serve different purposes and target different platforms. ARM OP-TEE is designed to enable secure applications on ARM-based devices, while Intel SMM is used for low-level system management tasks on Intel x86-based systems.

---

**Question:** What is RPMB?

**Answer:**

RPMB, or Replay Protected Memory Block, is a security feature found in some eMMC (embedded MultiMediaCard) and UFS (Universal Flash Storage) devices. It is designed to provide a secure, tamper-resistant, and write-protected storage area for sensitive data, like cryptographic keys or device-specific information.

The main characteristics of RPMB include:

1. **Replay protection:** RPMB uses a counter mechanism to prevent replay attacks. Each write operation to the RPMB partition must include a counter value greater than the current value stored in the RPMB. This ensures that old data cannot be replayed or re-written, protecting against unauthorized modifications.
2. **Authentication:** RPMB requires that write and read operations be authenticated using a shared secret key, usually derived from a hardware unique key (HUK) or device-specific key. This ensures that only authorized entities can access or modify the data stored in the RPMB partition.
3. **Write protection:** RPMB partitions are usually write-protected, meaning that once data is written, it cannot be modified or deleted. This ensures the integrity of the stored data and prevents unauthorized tampering.
4. **Limited storage capacity:** RPMB partitions are typically small in size, as they are intended for storing only critical security-related data.

RPMB is commonly used in devices like smartphones, tablets, and other embedded systems to store sensitive data that requires a higher level of security and protection against tampering or unauthorized access.

---

**Question:** What is HS in OP-TEE?

**Answer:**

In the context of OP-TEE, "HS" typically refers to the "Hardware Secure" world or environment. OP-TEE leverages Arm TrustZone technology, which partitions the system into two separate environments: the Normal world (also known as Non-Secure or NS) and the Secure world (also known as Hardware Secure or HS).

The Hardware Secure world is an isolated execution environment where the OP-TEE secure kernel and trusted applications run. This environment provides protection for sensitive data and secure operations, ensuring they are not accessible by the Normal world or vulnerable to tampering. The secure kernel manages the trusted applications, resources, and communication with the Normal world through APIs like the OP-TEE Client API.

In summary, "HS" in OP-TEE refers to the Hardware Secure world, an isolated environment within the Arm TrustZone technology where sensitive data and secure operations are protected from unauthorized access or tampering.

---

**Question:** What is KEK in OP-TEE?

**Answer:**

In the context of OP-TEE and security, KEK stands for Key Encryption Key. A Key Encryption Key is a cryptographic key used to encrypt and protect other keys, typically data encryption keys, ensuring their confidentiality and integrity. By using a KEK, sensitive keys can be securely stored, transmitted, or shared among different parties or systems without the risk of exposure.

In OP-TEE, Key Encryption Keys may be used to protect keys that are stored in the device or used by trusted applications for various security purposes, such as encryption, authentication, or digital signatures. The KEK itself is usually stored in a secure location, like an RPMB partition or within the secure environment provided by OP-TEE, to prevent unauthorized access or tampering.

To summarize, a KEK in OP-TEE is a cryptographic key used to protect other sensitive keys, ensuring their confidentiality and integrity when stored, transmitted, or shared.

---

**Question:** Does OP-TEE support multi-thread?

**Answer:**

OP-TEE, in its current design, supports multi-threading with certain limitations. The secure world in OP-TEE, which includes the secure kernel and trusted applications, is primarily single-threaded. This means that only one thread can execute at a time within the secure environment, regardless of the number of CPU cores available.

However, OP-TEE does support parallelism in the sense that it allows multiple trusted applications to be loaded and executed concurrently. This is achieved through the use of secure context switching between different trusted applications, ensuring that only one trusted application is active at a time.

It's worth noting that OP-TEE also supports concurrency in the normal world. Multiple normal world applications can communicate with different trusted applications simultaneously, and OP-TEE will manage the context switching between them.

In summary, while OP-TEE's secure environment is primarily single-threaded, it supports parallelism through the use of context switching between multiple trusted applications and concurrent communication with normal world applications.

---

**Question:** What is secure context switching in TrustZone?

**Answer:**

Secure context switching in TrustZone refers to the process of transitioning between different secure contexts (trusted applications) within the secure world while maintaining isolation and security. ARM TrustZone technology provides a hardware-based separation between the normal world and the secure world, allowing both to run independently and securely.

In the context of OP-TEE and TrustZone, when multiple trusted applications are running or when a normal world application requests a service from a trusted application, the secure kernel manages the execution of these trusted applications by performing secure context switching. This involves:

1. Saving the current context: The secure kernel saves the current state of the active trusted application, including its register values and memory, in a secure location.
2. Restoring the new context: The secure kernel restores the state of the trusted application that is being switched to by loading its previously saved register values and memory.
3. Switching execution: The secure kernel transfers control to the new trusted application, allowing it to continue executing where it left off.

Secure context switching ensures that each trusted application's state is kept separate and secure from others, maintaining isolation and preventing unauthorized access or tampering. Additionally, it allows the secure kernel to manage the concurrent execution of multiple trusted applications and handle requests from normal world applications efficiently.

In summary, secure context switching in TrustZone is the process of managing and transitioning between different trusted applications within the secure world while maintaining isolation, security, and efficient execution.

---

**Question:** Does OP-TEE support multi-process?

**Answer:**

OP-TEE supports multiple trusted applications (TAs) running concurrently within the secure environment provided by the Arm TrustZone. While the secure world in OP-TEE is primarily single-threaded, it can handle multiple processes in the form of trusted applications.

When multiple trusted applications are loaded and running, OP-TEE manages their execution using secure context switching. This ensures that only one trusted application is active at a time, while maintaining isolation and security between different trusted applications. The secure kernel in OP-TEE is responsible for managing resources, handling requests from the normal world, and switching between the trusted applications as needed.

On the normal world side, multiple processes can communicate with different trusted applications simultaneously, and OP-TEE will manage the context switching between them.

In summary, OP-TEE supports multi-process execution in the form of multiple trusted applications running concurrently within the secure environment, while also allowing multiple normal world processes to communicate with these trusted applications.

---

**Question:** Can secure kernel and trusted applications in OP-TEE can be multi-thread?

**Answer:**

As of now, the secure kernel and trusted applications in OP-TEE are primarily single-threaded. The secure world in OP-TEE does not support multi-threading within a single trusted application, meaning each trusted application can only execute one thread at a time.

However, OP-TEE does support parallelism in the sense that it allows multiple trusted applications to be loaded and executed concurrently. This is achieved through the use of secure context switching between different trusted applications, ensuring that only one trusted application is active at a time.

Although the secure environment itself is single-threaded, OP-TEE still enables concurrent communication with normal world applications. Multiple normal world applications can communicate with different trusted applications simultaneously, and OP-TEE will manage the context switching between them.

In summary, the secure kernel and trusted applications in OP-TEE are primarily single-threaded, but parallelism is achieved through secure context switching between multiple trusted applications and concurrent communication with normal world applications.

---

**Question:** Does OP-TEE support IPC?

**Answer:**

Yes, OP-TEE supports Inter-Process Communication (IPC) between trusted applications (TAs) within the secure world and between normal world applications and trusted applications.

1. IPC between trusted applications: Trusted applications in OP-TEE can share data and communicate with each other by invoking other trusted applications. However, the secure kernel must intermediate such communication to ensure isolation and maintain security. Trusted applications use the TEE\_InvokeTACommand function to invoke another trusted application and pass data as parameters.
2. IPC between normal world applications and trusted applications: OP-TEE supports communication between normal world applications and trusted applications through the GlobalPlatform TEE Client API. Normal world applications can open a session with a trusted application, invoke commands, and share data with the trusted application while maintaining security.

In both cases, OP-TEE manages the IPC to ensure isolation between different trusted applications and maintain the security of the secure environment. While the communication mechanisms may vary, OP-TEE provides the necessary APIs and infrastructure to support IPC between trusted applications and between normal world applications and trusted applications.

---

**Question:** Does OP-TEE support shared memory?

**Answer:**

Yes, OP-TEE supports shared memory, which allows normal world applications to share data with trusted applications (TAs) within the secure environment. Shared memory is used for exchanging data between the normal world and the secure world while maintaining security and isolation.

In OP-TEE, shared memory is managed by the GlobalPlatform TEE Client API, which provides functions for allocating, registering, and releasing shared memory. When a normal world application wants to share data with a trusted application, it creates a shared memory buffer using these API functions. The shared memory buffer is then passed as a parameter to the trusted application when invoking a command.

The secure kernel in OP-TEE ensures that the shared memory buffer is accessible by both the normal world application and the trusted application while preventing unauthorized access or tampering. The secure kernel also maintains the isolation between different trusted applications by limiting access to the shared memory buffers associated with each trusted application.

Please note that shared memory in OP-TEE is primarily used for communication between normal world applications and trusted applications. Sharing memory directly between different trusted applications is generally not supported, as it can compromise the isolation and security provided by the secure environment. Instead, trusted applications communicate with each other using the TEE\_InvokeTACommand function, as mentioned in a previous answer regarding IPC support.

In summary, OP-TEE supports shared memory for exchanging data between normal world applications and trusted applications while maintaining security and isolation within the secure environment.

---

**Question:** Please summarize multi-process, IPC, and shared memory of OP-TEE.

**Answer:**

In OP-TEE, multi-process, IPC, and shared memory are features that enable concurrent execution, communication, and data sharing between normal world applications and trusted applications, as well as between trusted applications themselves.

1. Multi-process: OP-TEE supports multiple trusted applications running concurrently within the secure environment. The secure kernel manages the execution of these applications using secure context switching, ensuring that only one trusted application is active at a time while maintaining isolation and security.
2. IPC (Inter-Process Communication): OP-TEE supports communication between trusted applications and



2. **IPC (Inter-Process Communication):** OP-TEE supports communication between trusted applications and between normal world applications and trusted applications. For communication between trusted applications, the TEE\_InvokeTACommand function is used. For communication between normal world applications and trusted applications, the GlobalPlatform TEE Client API is used. In both cases, OP-TEE manages the IPC to ensure isolation and maintain security.
3. **Shared memory:** OP-TEE supports shared memory to allow normal world applications to exchange data with trusted applications securely. The GlobalPlatform TEE Client API manages shared memory buffers, which can be passed as parameters when invoking commands in trusted applications. The secure kernel ensures that the shared memory buffer is accessible by both parties while preventing unauthorized access or tampering.

In summary, OP-TEE provides multi-process support for concurrent execution of multiple trusted applications, IPC for communication between applications, and shared memory for secure data exchange. These features enable efficient and secure interaction between normal world and trusted applications while maintaining isolation and security within the secure environment.

---

**Question:** Can secure world can access memory in normal world?

**Answer:**

In general, the secure world in OP-TEE can access memory in the normal world, but it must do so with caution. The secure world has a higher level of privilege, allowing it to access both secure and normal world memory. However, accessing normal world memory from the secure world can introduce potential security risks, such as data leaks or manipulation of sensitive information.

To minimize such risks, OP-TEE employs a set of precautions and guidelines:

1. **Use shared memory:** When secure world applications need to access data from the normal world, it is recommended to use shared memory, which is explicitly allocated and managed by the GlobalPlatform TEE Client API. This ensures proper isolation and access control for the shared data.
2. **Validate memory addresses:** If the secure world must access normal world memory directly, it should first validate the memory address range to ensure it doesn't overlap with secure memory or any other protected memory region.
3. **Be cautious of data manipulation:** The secure world should be aware that normal world memory can be manipulated or tampered with by potentially malicious software running in the normal world. Sensitive operations should not rely solely on data stored in normal world memory without additional validation or protection.
4. **Minimize normal world memory access:** The secure world should minimize direct access to normal world memory whenever possible and instead rely on secure memory or shared memory buffers for data exchange.

While the secure world in OP-TEE can access normal world memory, it should be done with caution and following proper guidelines to maintain the security and isolation provided by the ARM TrustZone technology.

---

**Question:** Can normal world access memory in secure world?

**Answer:**

In the context of OP-TEE and Arm TrustZone technology, the normal world cannot directly access the memory in the secure world. The TrustZone technology enforces hardware-based isolation between the normal and secure worlds, which prevents normal world software from accessing or tampering with secure world memory.

The secure world memory, which includes the secure kernel, trusted applications, and their associated data, is

The secure world memory, which includes the secure kernel, trusted applications, and their associated data, is protected from unauthorized access by the normal world. This ensures that sensitive data and operations within the secure world remain confidential and secure.

If a normal world application needs to exchange data with a trusted application in the secure world, it must use shared memory, which is explicitly allocated and managed by the GlobalPlatform TEE Client API. This provides a controlled mechanism for data exchange between the normal and secure worlds, maintaining security and isolation.

In summary, the normal world cannot directly access memory in the secure world, thanks to the hardware-based isolation provided by Arm TrustZone technology. Data exchange between the two worlds is facilitated through shared memory, which is managed by the GlobalPlatform TEE Client API.

---

**Question:** Does OP-TEE support network communication in secure world?

**Answer:**

OP-TEE does not provide built-in support for direct network communication from within the secure world. The main reason for this is to maintain the security and isolation of the secure environment, as exposing networking capabilities directly to the secure world could introduce potential vulnerabilities or attack surfaces.

In most use cases, network communication is handled by normal world applications. If there is a need for secure communication involving trusted applications in the secure world, it is usually done indirectly:

1. Normal world applications handle network communication and exchange data with trusted applications in the secure world using the GlobalPlatform TEE Client API and shared memory.
2. Secure operations, such as encryption, decryption, or authentication, can be performed by trusted applications in the secure world. The normal world application can then send or receive the processed data over the network.
3. If end-to-end security is required, the normal world application can use additional encryption or authentication protocols, in conjunction with secure operations performed by trusted applications.

By handling network communication in the normal world and only exchanging data with the secure world as needed, OP-TEE maintains the isolation and security of the secure environment while still enabling secure network communication indirectly.

In summary, OP-TEE does not support direct network communication within the secure world. Instead, network communication is typically handled by normal world applications, which can securely exchange data with trusted applications in the secure world using the GlobalPlatform TEE Client API and shared memory.

---

**Question:** <https://github.com/OP-TEE>

**Answer:**

The link you provided, <https://github.com/OP-TEE>, is the official GitHub repository for the OP-TEE project. This repository contains the source code, documentation, and other resources related to OP-TEE, an open-source Trusted Execution Environment (TEE) based on the Arm TrustZone technology.

The repository is organized into several sub-projects, including:

1. `optee_os`: The core component, which includes the secure kernel and related components that run within the secure world. It manages trusted applications, resources, and secure context switching.
2. `optee_client`: The library implementing the GlobalPlatform TEE Client API, which enables normal world applications to communicate with trusted applications in the secure world.

3. `optee_test`: A set of tests and examples to verify the functionality of the OP-TEE system, including trusted application examples and test suites for various cryptographic operations.
4. `build`: A set of scripts and configuration files to help users build and compile the OP-TEE components for different target platforms.

The GitHub repository is an excellent starting point for developers interested in working with OP-TEE, as it provides the necessary resources and documentation to understand, build, and deploy OP-TEE on supported platforms. If you are interested in learning more or contributing to the project, you can explore the repository, read the documentation, and follow the guidelines provided.

---

## TrustZone.0329.md

---

**Question:** <https://genode.org/documentation/articles/trustzone>

**Answer:**

The link you've shared (<https://genode.org/documentation/articles/trustzone>) directs to an article on Genode's website that discusses TrustZone technology. Genode is an open-source operating system framework focused on creating a secure and scalable environment for running multiple instances of software. TrustZone is a security technology developed by ARM, which is widely used in modern processors.

TrustZone technology aims to provide a secure execution environment for trusted applications, protecting them from potential security threats posed by untrusted applications. It achieves this by dividing the system into two distinct worlds: the "Secure World" and the "Normal World." These two worlds run concurrently on the same hardware, but they are isolated from each other, ensuring that sensitive data and operations remain protected.

The "Secure World" is designed to handle security-sensitive tasks, while the "Normal World" handles general-purpose tasks. To switch between the two worlds, ARM provides a secure monitor call (SMC) instruction that manages the transitions.

Genode leverages TrustZone technology to enhance the security of its operating system framework. By using TrustZone, Genode can isolate critical parts of the system and applications from the rest of the system, reducing the risk of sensitive data leaks or unauthorized access.

If you have any specific questions about TrustZone or Genode, feel free to ask, and I'll do my best to help.

---

**Question:** Please summarize the following articles in short.

**Answer:**

It appears you forgot to provide the articles that you'd like me to summarize. Please provide the links or titles of the articles, and I'll be happy to provide summaries for you.

---

**Question:**

An Exploration of ARM TrustZone Technology

ARM TrustZone technology has been around for almost a decade. It was introduced at a time when

The figure above illustrates the concept of the two worlds. The normal world is active (non-se

As former university researchers in the field of OS security, ARM TrustZone was deemed as rele

These questions prompted us to dive right into the world of TrustZone. We spent two and half y

Section Starting point describes the starting point of our journey and lays out the possible r

### Answer:

Title: An Exploration of ARM TrustZone Technology

Summary: ARM TrustZone technology is a security solution for ARM-based platforms that introduces a "secure mode" alongside the regular "normal mode." The secure mode establishes a "secure world" and a "normal world" that run concurrently but are isolated from each other, ensuring that sensitive data and operations remain protected. TrustZone is more flexible than TPM chips because its functionality is defined by system software rather than being hard-wired. Despite initial skepticism, TrustZone has gained attention as a marketing term for SoC vendors to promote confidence in their chip's security. The article delves into various aspects of TrustZone, including its rationale, design, implementation, and common questions related to the technology.

### Question:

Starting point

The first question on the table when starting a series of experiments was the one for a suitab

At the time we started our investigations, we were most interested in the ARM Cortex-A9-based

We conducted two lines of experimentation: Prototyping using an existing microkernel and the c

For both lines of work, we used the Genode OS Framework (Genode) as foundation. Genode is a co

For the first line of work, the enablement of the Versatile Express platform, one of the kerne

In our second line of work, we supplemented Genode with a new base platform that we call "base

### Answer:

Title: Starting Point and Experimentation with ARM TrustZone Technology

Summary: The article discusses the process of choosing a suitable development platform for exploring ARM TrustZone technology. It narrows down to the ARM Versatile Express Cortex-A9 board, an official reference board by ARM. Two lines of experimentation are conducted: prototyping using an existing microkernel and creating a custom kernel platform. Genode OS Framework is used as the foundation for both lines of work, due to its low trusted computing base complexity and high flexibility. The first line of work uses the combination of Genode with Fiasco.OC to enable the principal use of Genode on the platform, while the second line of work supplements Genode with a new base platform called "base-hw" to reduce the TCB complexity of the base system and gain insights into kernel-level problems when using Cortex-A9 and TrustZone.

### Question:

The design and genesis of our custom base-hw kernel platform

In contrast to classical L4 microkernels where Genode's core process runs as user-level rootta

The figure illustrates the effect of the new design on the TCB of the root of Genode's process

For realizing the base-hw kernel platform, we undertook the following steps. First, we designe

The kernel entry and exit code paths had been implemented and tested on both Qemu and the refe

The next step was to add support for interrupt handling and preemptive multi-threading. So we

The main thread that initializes the platform and spawns the first user-level process called i

A pager thread that receives and handles page faults,

One thread per device interrupt,

A so called RPC endpoint that handles requests to the services provided by core, and

The parent thread of the init process that responds to requests of the init process via the pa

Because the correct functioning and scheduling of those threads is fundamental for bringing-up

The user-level sender marshals its payload into its UTCB and invokes the kernel,

The kernel transfers the payload from the sender's UTCB to the receiver's UTCB and schedules t

The receiver retrieves the incoming message from its UTCB.

Because all UTCBs are always mapped in the kernel, no page faults can occur during the second

In addition to IPC, threads interact via the synchronization primitives provided by the Genode

With the entity of core running on the reference platform, it was time for spawning the first

A CPU session with the main thread of the init process,

A region-manager (RM) session describing the address space of the process,

A protection-domain (PD) session representing the encapsulation boundary of the process,

A RAM session containing the physical memory resources available to the init process.

To accommodate the implementation of these services, the kernel was enhanced with system calls

At the startup of the init process, its first life sign is a page fault produced by the main t

All further functionalities needed to bring up the init process such as the ELF loading were r

The last step towards executing real-world application scenarios on our custom kernel was the

With those functionalities in place, the path was cleared to execute user-level device drivers

For more sophisticated work loads that require asynchronous communication, we subsequently imp

## Answer:

Title: Design and Genesis of Custom Base-HW Kernel Platform for Genode

Summary: The custom base-hw kernel platform for Genode features a unique hybrid kernel/userland design that simplifies the kernel part, side-steps hard kernel-level problems, and reduces redundancies between kernel and roottask, ultimately resulting in a lower trusted computing base (TCB) complexity. The platform was developed in several steps, including designing and implementing the bootstrapping concept, kernel entry and exit code paths, support for interrupt handling and preemptive multi-threading, core-local context switching, inter-process communication, and user-level device driver mechanisms. This custom kernel platform enabled the execution of real-world application scenarios and provided full coverage of the Genode API.

## Question:

Hypervisor managing the non-secure world

This section describes our experiences while creating a TrustZone-aware hypervisor that is abl

---

**Answer:**

Title: TrustZone-Aware Hypervisor for Managing the Non-Secure World

Summary: A TrustZone-aware hypervisor was developed for managing one non-secure virtual machine (VM) and multiple tasks running unprivileged in secure mode. The hypervisor was built upon the custom base-hw kernel platform and involved several steps, including making the kernel TrustZone-aware and constructing a virtual machine monitor (VMM) as a user-level component on top of the kernel. The hypervisor was designed to seamlessly schedule between the non-secure VM and secure tasks, providing a robust and secure foundation for managing different security domains in a TrustZone-enabled system.

---

**Question:**

World switch between non-secure world and secure world

We started investigating TrustZone by bringing the so called monitor-mode in the Genode/ARM-ke

In addition to interrupt-triggered mode switches, the non-secure world is able to explicitly e

Thereby, our consideration was that all exceptions that enter the monitor mode should be trigg

---

**Answer:**

Title: World Switch Between Non-Secure and Secure Worlds

Summary: A world switch mechanism was implemented to facilitate transitions between the non-secure and secure worlds in a TrustZone-enabled system. The monitor mode was utilized, which is an execution mode add-on in ARM CPUs with Security Extensions (TrustZone). The system is designed so that all exceptions entering the monitor mode should be triggered only by the non-secure world, simplifying the assembler world-switch routine. The secure world can switch back to the non-secure world through a system call with a pointer to the VM's CPU state, pushing the provided state to the non-secure world.

---

**Question:**

User-level Virtual Machine Monitor (VMM)

To be able to use the newly introduced system call that switches to the non-secure-world, we i

The image above illustrates the relationship between the user-level VMM (TZ VMM) and Genode's

Given the world-switch routine in the kernel and the VM-session interface of core, all prerequ

---

**Answer:**

Title: User-Level Virtual Machine Monitor (VMM)

Summary: A new VM-session interface was introduced as a service in Genode's core to enable a client (VMM) to manage the CPU state of a VM, initiate a world switch to the non-secure world, and obtain the VM's state. This allows the creation of a simple VMM that acquires a portion of RAM, loads the VM's ELF binary to physical RAM, and sets the program counter of the VM's CPU state. The VMM enters an endless loop, executing the VM and dumping its CPU state to the debugging console after each exceptional return.

---

**Question:**

Simple test kernel for the non-secure world

To test the newly implemented world-switch routine, we created a simple test kernel. It is

Setup exception vectors for data-aborts, pre-fetch aborts, interrupts, fast-interrupts, superv

Initialize interrupt controller to receive UART0 interrupts,

Initialize UART0 to be able to print simple messages, and detect interrupt reception (RX),

Switch to user-mode and idle,

Whenever an interrupt occurs print some short message.

TrustZone Platform Controller (TZPC) and Address-Space Controller (TZASC)

After successfully testing the VMM setup and the world-switch routine, it was time to investig

TrustZone Protection Controller (TZPC)

TrustZone Address Space Controller (TZASC)

The Cortex A9 MPCore internal Interrupt Controller

The first and easiest part seemed to be the configuration of the physical memory areas that sh

Due to experimentation, we were able to deduct the following insights. The TZASC controller on

The TZPC is used to protect on-chip peripherals (e.g., the TZPC and TZASC themselves) as well

Given those findings, the Versatile Express platform apparently does not allow the partitionin

---

**Answer:**

Title: Simple Test Kernel for Non-Secure World & TrustZone Platform and Address-Space Controller

Summary: A simple test kernel was created to test the VMM setup and the world-switch routine. TrustZone mechanisms were investigated to protect the secure world from the non-secure world. The Versatile Express platform features TrustZone Protection Controller (TZPC) and TrustZone Address Space Controller (TZASC). The TZASC controller is used to secure physical address ranges addressable through the Static Memory Controller (SMC), and the TZPC protects on-chip peripherals and bus accesses to external subsystems. The platform does not allow the partitioning of DDR RAM into secure and non-secure address ranges, but it does host 32 MiB of SRAM that can be preserved for exclusive use by the secure world.

---

**Question:**

Interrupts

In the previous section, we identified how to partition the physical resources of the referenc

Unfortunately, the assignment of device interrupts to either the secure or non-secure world ca

Each type has a unique exception vector so that FIQs enter the kernel using a different code p

On the occurrence of an IRQ, the CPU automatically saves a more comprehensive register state c

The most distinctive characteristic is that an FIQ can preempt any exception handler except fo

For a device interrupt, it is possible to define whether to take the IRQ or FIQ exception entr

---

To summarize the above, FIQs are exclusively used by the device drivers of the secure world wh

---

**Answer:**

Title: Interrupts in Secure and Non-Secure Worlds

Summary: Device interrupts can be indirectly partitioned between the secure and non-secure worlds using the ARM architecture's two types of interrupts: normal interrupts (IRQ) and fast interrupts (FIQ). The custom kernel can handle FIQs for the secure world and IRQs for the non-secure world. By defining whether a device's interrupts are delivered as FIQs or IRQs, it is possible to assign individual device interrupts to either the secure or non-secure world. This setup allows secure world device drivers to use FIQs exclusively while IRQs are used by the non-secure world, as expected by the Linux kernel.

---

**Question:**

Booting Linux in the non-secure world

Having the principal separation between the secure and non-secure world in place, we can now r

Loading the kernel. The kernel has to be complemented with additional boot-time information ca

In addition to the kernel, a RAM disk must be loaded into the memory of the non-secure world.

To test the loading procedure, we started Linux with all access rights to the peripherals gran

SP810\_ENABLE

Enable the timer.

CPU\_ID:

Request the CPU ID, which is the hard-coded value 0xc000191 for the reference platform.

SYS\_COUNTER

Returns the value of the Sys\_24MHz system register.

MISC\_FLAGS

Returns the value of the Sys\_misc system register.

SYS\_CTRL

Perform one of the system configuration control operations OSC1, DVI\_SRC, and DVI\_MODE.

MCI\_STATUS

Returns the value of the Sys\_mci system register.

We would have preferred to employ a trap-and-execute emulation scheme for those register acces

---

**Answer:**

Title: Booting Linux in the Non-Secure World with TrustZone

Summary: To boot Linux in the non-secure world using TrustZone, a Linux kernel and a RAM disk must be loaded into memory. The kernel is complemented with ATAGs data structure, which contains essential boot-time information. To ensure security, access to peripherals is granted or denied based on their criticality for the secure world. For devices that must be preserved for the secure world, the Linux kernel code is modified to issue hypercalls instead of directly accessing device resources. The hypervisor reflects these hypercalls to the user-level VMM, which responds to them accordingly.

---

**Question:**



## Question:

Device emulation

We considered device emulation as a less intrusive alternative to the introduction of hypercalls.

The basic idea of emulating device access is to let the hypervisor pass control to the VMM as

That said, we found that the trap-and-execute emulation model is not possible to implement with

The image above displays the flow of control when the normal-world OS performs an illegal device

We found that the latency of reporting access violations can be reduced by adding memory barriers

The general pattern of changing the original code to using a hypercall looks as follows (ctrl\_r

```
+#ifdef RUNS_IN_SECURE_WORLD
    if (ctr)
        return readl(ctr);
+#else
+ if (ctr) {
+     static u32 ret;
+     asm volatile("mov r1, #3          \n"
+                 "dsb                  \n"
+                 "dmb                  \n"
+                 "smc                  \n"
+                 "mov %[value], r0 \n" : [value] "=r" (ret) :: "r0", "r1");
+     return ret;
+ }
+#endif
```

We replaced the original call to the access function `readl(ctr)` with the snippet above.

A ready-to-use configuration for the Linux kernel accompanied with the needed modification to

Adaptation of the Linux kernel

<https://github.com/skalk/linux/commit/284073c4f6c0e6a77c02ae9d296da9c46f6f5104.patch>

We tried to keep the kernel patch as small as possible. The patch comprises the addition of on

## Answer:

Title: Device Emulation in TrustZone's Non-Secure World

Summary: Device emulation in the non-secure world using TrustZone is considered as an alternative to introducing hypercalls. However, trap-and-execute emulation is not feasible due to the latency in reporting access violations. Instead, modifications to the Linux kernel are made by replacing calls to access functions with hypercalls, allowing Linux to boot completely in the non-secure world. A minimal kernel patch was developed to achieve this, only adding 73 lines of code and enabling the successful boot of Linux in the non-secure world.

## Question:

TrustZone demonstrated in color

The goal of our ARM TrustZone experiments was to push the envelope of this technology beyond t

We identified the Freescale i.MX53 SABRE tablet as a suitable platform for this experiment. It

The practical work consisted of the following steps:

Adaptation of Genode to the i.MX53 platform,

Enabling TrustZone on i.MX53,

Implementing a TrustZone monitor that deals with the specifics of the i.MX53 with the goal of

Display driver running as user-level device driver on Genode

Touchscreen driver running as user-level device driver on Genode

Interfacing Genode's display and user-input drivers with the Android software stack.

Multiplexing the display between Android and Genode using hardware overlays.

In the following, each of those topics is briefly covered.

Genode on i.MX53 SABRE tablet

We conducted our first TrustZone experiments on the ARM Versatile Express platform based on a

TrustZone on i.MX53

As the i.MX53 SoC is based on a Cortex-A8 CPU instead of a Cortex-A9 as used on the Versatile

One particular i.MX-specific difference is the routing of interrupts. Because i.MX does use a

The second distinctive property of i.MX53 compared to the Versatile Express platform is the so

TrustZone monitor

The user-level TrustZone monitor (we call it VMM because it acts very similar to a virtual mac

For our prototype, we partitioned the platform where easily feasible (e.g., for DDR memory, in

The hypercalls implemented by the i.MX53-specific VMM are related to the access of a few platf

---

## Answer:

Title: TrustZone Demonstrated on i.MX53 SABRE Tablet with Genode OS

Summary: The ARM TrustZone technology was utilized to execute a complete Genode-based operating system in the secure world and an unmodified Android OS in the normal world on an i.MX53 SABRE tablet. The secure OS included a graphical user interface with touchscreen interaction, allowing the user to switch between both worlds. Adaptations were made to Genode for i.MX53, and a TrustZone monitor was implemented to deal with the platform's specific requirements. The display was multiplexed between Android and Genode using hardware overlays, successfully demonstrating the capabilities of TrustZone beyond typical TPM-like functionality.

---

## Question:

Additional device drivers

In addition to the low-level drivers outlined above, the demonstration scenario required us to

The task of creating the drivers, in particular the framebuffer driver, turned out to be more

Compared to the framebuffer driver, enabling the touchscreen device was a smooth experience wi

Similar to most embedded graphics devices, the i.MX53 Image Processing Unit (IPU) comes with t

Since we wanted to run Android at almost-native performance in the normal world, we found that

With our custom IPU driver with hardware-overlay support in place, the seemed to be paved to s

Unfortunately, however, the TrustZone implementation of the i.MX53 SoC has a limitation that r

---

## Answer:

Title: Implementing Additional Device Drivers and Addressing TrustZone Limitations on i.MX53 SABRE Tablet

Summary: Additional device drivers were required for the ARM TrustZone demonstration on the i.MX53 SABRE tablet, including a framebuffer driver for the LCD display, a touchscreen driver, and a capacitive sensor driver. While developing the framebuffer driver proved to be more complex than anticipated, the touchscreen driver was implemented without major issues. Hardware overlays were employed to allow Android to run at near-native performance in the normal world, but TrustZone implementation limitations regarding the isolation between the Image Processing Unit (IPU) and GPU were discovered. To fix the security hole, the SoC would need to separate the DMA policy of both devices or the secure world would need to virtualize the GPU.

---

#### Question:

Demonstration scenario

Architectural overview of the TrustZone demo on i.MX53

Figure 6 presents an overview of the key components of the demonstration scenario. Dashed line

At boot time, Genode's core sets up the CSU configuration and thereby defines the permissions

For the demonstration scenario, the VMM provides a virtual framebuffer and a virtual touchscreen

User input is always received by the secure world via the touchscreen driver and the Nitpicker

The demonstration scenario is available at the following branch of Genode.

[https://github.com/skalk/genode/tree/i.MX53\\_tablet\\_demo](https://github.com/skalk/genode/tree/i.MX53_tablet_demo)

Genode as secure OS on a i.MX53 SABRE tablet.

Detailed instructions about reproducing it are provided by the README file at the `os/src/serve`

---

#### Answer:

Title: TrustZone Demonstration Scenario on i.MX53 SABRE Tablet

Summary: The demonstration scenario for ARM TrustZone on i.MX53 SABRE tablet consists of key components like Genode's core, user-level device drivers, and a user-level TZ VMM, which manages the normal world. The VMM provides a virtual framebuffer and virtual touchscreen device to the normal world, with Android OS having direct access to the GPU. This setup maintains graphics performance and ensures the secure world retains control of the display and user input routing. The demonstration scenario can be accessed through the provided GitHub repository, with detailed instructions for reproducing it available in the README file.

---

#### Question:

Common questions, answered

In this Section, we attempt to answer common questions that we were repeatedly asked. Note tha

What are the capabilities of TrustZone?

TrustZone technology can be viewed from two angles, as virtualization solution and as mechanis

The number of virtual machines is limited to two, one running in the secure world and one runn

There is no support to virtualize MMIO resources via the trap-and-execute model. As described

To guarantee that both worlds access distinct device resources only, certain device drivers of

---

There is no way to virtualize the physical memory as used by the non-secure OS. The guest-phys  
Despite those limitations, we identified a single advantage of TrustZone compared to other vir  
We ultimately reached the conclusion that perceiving TrustZone as a virtualization mechanism i  
What are the limitations of TrustZone for compartmentalization?  
There are only two "worlds".

Access by the non-secure side to the physical address space can be restricted but physical add  
The granularity of access restricts depends on the SoC. For example, the Versatile Express pla  
The use of TrustZone is not entirely transparent to the non-secure side because non-accessible  
How to host two different compartments/zones (one trusted, one untrusted)?  
The general approach consists of the following steps:

The system starts in secure mode and boots the hypervisor.  
The hypervisor sets up the non-secure world (i.e., by configuring the TZPC, TZASC, and PIC).  
The hypervisor loads the OS image to be executed in the non-secure world.  
The hypervisor enters the non-secure world at the entry point of the non-secure OS.  
The non-secure OS issues hypercalls for functions that are hidden from the non-secure world. T  
In our particular experiments, the hypervisor is represented by an instance of the Genode OS,  
The individual steps of this process are covered in more detail in section Hypervisor managing

## Answer:

Title: TrustZone Common Questions and Answers

Summary: TrustZone technology is best viewed as an alternative to TPMs rather than as a virtualization solution. It offers versatility and programmability, but it has limitations for compartmentalization, such as only having two worlds, non-virtualized physical addresses, and granularity of access restrictions that depend on the SoC. To host two different compartments or zones (trusted and untrusted), a hypervisor is utilized to set up the non-secure world, load the OS image for the non-secure world, and handle hypercalls from the non-secure OS. In experiments, the Genode OS was used as the hypervisor for this purpose.

## Question:

Does it make sense to run commodity OSes like Android in the secure world?  
Following the rationale of Section What are the capabilities of TrustZone?, the use of TrustZo  
On platforms such as the Versatile Express platform, only small portions of memory can be pres  
How does TrustZone compares to ARM's virtualization extensions?  
The virtualization extensions introduced with Cortex-A15 are orthogonal to TrustZone. There ar  
The virtualization extensions devise the implementation of the virtual machine monitor as part  
Are there any security improvements of the Cortex-A15 over the A9 architecture then?  
The virtualization extensions introduced with Cortex-A15 offer an additional instrument for pa  
That said, the security properties of TrustZone largely depend on the SoC rather than the revi  
What are the capabilities of ARM's virtualization extensions?

There is a new CPU privilege level ("Hyp" mode) below the existing CPU modes, which is only su

Virtualization holes present in previous generations of the ARM architecture have been fixed f

The trap-and-execute model principally allows for executing unmodified guest OSES on top of th

In particular, the handling of virtual interrupts can be accelerated by using a new virtual IR

The Cortex-A15 facilitates the emulation of memory-mapped device registers by providing useful

To support multiple virtual machines, a guest-physical to host-physical address translation ha

There is no virtualization of DMA accesses issued by bus peripherals (i.e., display controller

There is no DMA protection. Additional per-device MMUs would be required. The presence of such

IRQs designated to a device driver running in a guest OS are always handled by the hypervisor,

For more information on ARM virtualization extensions and the procedure of creating a Cortex-A

How does TrustZone help to securely store secrets?

Hiding peripherals and memory from the non-secure world is a key feature of TrustZone. TrustZo

That said, each SoC has different characteristics with regard to assigning or partitioning sto

How does TrustZone facilitate secure booting?

As a precondition for the use of TrustZone for secure booting, the code running in the secure

Given that the secure world is booted in a secure fashion, the loader of the non-secure OS is

If the SoC lacks a way to fix the boot code for the secure world, secure booting cannot be imp

When using Genode as secure OS, the SoC should provide sufficient memory to be preserved for t

## Answer:

Title: TrustZone and Commodity OSES, ARM's Virtualization Extensions, and Secure Booting

Summary: Running commodity OSES like Android in the secure world of TrustZone is not recommended due to increased complexity and limitations on platforms like the Versatile Express platform. ARM's virtualization extensions, introduced with Cortex-A15, are orthogonal to TrustZone and provide additional capabilities for partitioning systems through the Hyp mode. TrustZone can help securely store secrets by hiding peripherals and memory from the non-secure world and can facilitate secure booting if the SoC supports securely bootstrapping the secure world. The implementation of secure booting and secure storage depends on the SoC used.

**Question:** Please compare TrustZone and TPM in a table.

## Answer:

Feature	TrustZone	TPM
Purpose	Hardware-based security solution	Hardware-based security solution
Security functions	Freely programmable	Fixed-function
Execution environment	Secure world on ARM-based SoC	Standalone chip
Isolation	Two worlds: Secure and Non-secure	Separate from CPU
Complexity	Varies, from simple libraries to full OS	Limited, designed for specific functions
Interaction with peripherals	NS bit determines peripheral access control	Communicates via standard bus protocols
Cryptographic capabilities	Software implemented, based on user needs	Built-in cryptographic functions
Secure storage	Depends on SoC implementation	Built-in non-volatile storage
Secure boot	SoC-dependent, can be implemented	Provides support for secure boot process
Virtualization support	Limited, not designed for virtualization	Not applicable
Flexibility	More versatile, programmable functions	Less

versatile, fixed-function | | Performance impact | Depends on implementation complexity | Minimal, as it operates independently |

Please note that this comparison is a high-level overview of TrustZone and TPM. The specific features and capabilities may vary depending on the implementation and the SoC used for TrustZone or the TPM version and vendor.

---

**Question:** What is bootstrap code?

**Answer:**

Bootstrap code, also known as boot code, bootloader, or bootstrapping, is a small program responsible for initializing a computer's hardware components and loading the operating system (OS) into memory. When a computer is powered on or restarted, the bootstrap code is the first code to execute, typically stored in a fixed location, such as a Read-Only Memory (ROM) chip, or firmware like Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI).

The primary function of the bootstrap code is to perform hardware checks, initialize essential hardware components (like CPU, memory, and peripherals), and locate the operating system's kernel. Once the kernel is found, the bootstrap code loads it into memory and transfers control to the kernel, enabling the OS to start and manage system resources.

In summary, bootstrap code is responsible for the initial steps in the boot process, laying the groundwork for the operating system to take over and provide a functional environment for the user.

---

**Question:** What are two lines of experimentation? Please summarize them in short.

**Answer:**

Two lines of experimentation refer to two distinct approaches or methods used to investigate a problem, hypothesis, or research question. In scientific research, these two lines can help validate results, uncover new insights, or compare different strategies. Here's a brief summary of each line:

1. Line 1: This line of experimentation focuses on a particular method, technique, or approach to addressing the research question. It can involve specific experimental designs, data collection methods, or analysis techniques to investigate the hypothesis or problem.
  2. Line 2: This second line of experimentation employs an alternative method or approach to explore the same research question. By utilizing a different technique, researchers can compare the outcomes of both lines of experimentation and validate their findings or reveal new aspects of the research question that were not apparent in the first line of experimentation.
- 

**Question:** What are two lines in the section "Starting point" in the article?

**Answer:**

In the context of the "Starting point" section of the article, two lines refer to two distinct approaches to exploring TrustZone technology:

1. Line 1: Utilize TrustZone as a virtualization solution, exploring its potential to host multiple virtual machines or support multiple operating systems in the secure and non-secure worlds. This line of experimentation seeks to understand TrustZone's capabilities as a virtualization mechanism.
2. Line 2: Use TrustZone as an alternative to Trusted Platform Modules (TPM), focusing on its ability to implement security functions and protect sensitive information within the secure world. This line of

experimentation aims to investigate the versatility and programmability of TrustZone as a security mechanism compared to fixed-function TPMs.

---