2023-04-20

Created: 2023-05-01 22:17:45

Questions

- Attestation `(230418.SGX.yaml)`
  - `[09:22]` Is secure boot a hardware or software attestation? `(SGX.0418.md)`
  - `[09:24]` Do you mean that UEFI Secure Boot requires TPM? `(SGX.0418.md)`
- Enclave `(230418.SGX.yaml)`
  - `[09:36]` What is enclave in Intel SGX? `(SGX.0418.md)`
  - `[09:41]` Which key is encrypt memory encrypted by? `(SGX.0418.md)`
  - `[09:43]` How does CPU run code in the encrypted memory in enclave? `(SGX.0418.md)`
  - `[17:16]` Can a program in an enclave see outside? `(SGX.0418.md)`
- MRSIGNER `(230418.SGX.yaml)`
  - `[09:52]` What is MRSIGNER? `(SGX.0418.md)`
  - `[10:19]` What is enclave identity described in the paper? `[error]` `(Papers.SGX.0419.md)`
  - `[10:34]` Who does generate MRSIGNER? `(SGX.0418.md)`
  - `[10:38]` So the MRSIGNER value is generated in offline not at runtime. Right? `(SGX.0418.md)`
  - `[10:39]` Where is the private key stored? `(SGX.0418.md)`
- SIGSTRUCT `(230418.SGX.yaml)`
  - `[10:59]` Please explain the sentence in the paper: “The enclave builder presents the hardware with an RSA signed enclave certificate (SIGSTRUCT)” and list the steps. `(Papers.SGX.0419.md)`
- Keys `(230418.SGX.yaml)`
  - `[11:19]` What is Sealing Key? `(Papers.SGX.0419.md)`
- Sealing Identity `(230418.SGX.yaml)`
  - `[11:30]` How does “Sealing to the Sealing Identity” work? Please list steps. `(Papers.SGX.0419.md)`
  - `[11:32]` Please compare “Sealing to the Enclave Identity” and “Sealing to the Sealing Identity” in a table. `(Papers.SGX.0419.md)`
- EPID `(230418.SGX.yaml)`
  - `[14:20]` Does Intel EPID require TPM? `(SGX.0418.md)`
  - `[14:26]` How is EPID keys enrolled into a device? `(SGX.0418.md)`
- Slideshow - Intro to Intel SGX `(230420.Slide.SGX.yaml)`
  - `[14:36]` Below slideshow is “Intro to Intel SGX” `(Slides.SGX.0420.md)`
- My Questions `(230420.Slide.SGX.yaml)`
  - `[14:39]` Why is Intel SGX not suited for password managers? `(Slides.SGX.0420.md)`
  - `[15:01]` What does “size of TCB” mean? `(Slides.SGX.0420.md)`
  - `[15:06]` What is “accountable decryption”? `(Slides.SGX.0420.md)`
  - `[16:03]` What is endorsement key? `(Slides.SGX.0420.md)`
- Launch Enclave `(230420.Slide.SGX.yaml)`
  - `[15:10]` What is “Launch Enclave”? `(Slides.SGX.0420.md)`
  - `[15:11]` Is Launch Enclave a software? `(Slides.SGX.0420.md)`
  - `[15:12]` Does Launch Enclave run in OS? `(Slides.SGX.0420.md)`
  - `[15:13]` Does the binary of Launch Enclave reside in BIOS or ME? `[title]` `(Slides.SGX.0420.md)`
  - `[15:16]` Is Launch Enclave provided by another vendor (No Intel)? `(Slides.SGX.0420.md)`
- Privacy CA `(230420.Slide.SGX.yaml)`
  - `[16:14]` How does the Solution 1 “Privacy CA for provisioning ak” work? Please list the steps. `(Slides.SGX.0420.md)`
  - `[16:18]` What is “Privacy CA”? `(Slides.SGX.0420.md)`

Q & A

Attestation `(230418.SGX.yaml)`

Question: `[09:22]` Is secure boot a hardware or software attestation?`(SGX.0418.md)`

Answer:

Secure Boot can be considered a form of hardware-based attestation, as it relies on the hardware components of a system to ensure the integrity of the boot process. It leverages the capabilities of a Trusted Platform Module (TPM) or similar security hardware to verify the digital signatures of firmware, boot loaders, and operating system components. This ensures that only trusted software is executed during the boot process.

However, Secure Boot also involves software components, such as firmware, boot loaders, and digital signature verification routines. So, while it is primarily hardware-based attestation, it also relies on software to perform certain tasks and checks.

Question: `[09:24]` Do you mean that UEFI Secure Boot requires TPM?`(SGX.0418.md)`

Answer:

I apologize for any confusion. UEFI Secure Boot does not strictly require a Trusted Platform Module (TPM). However, a TPM can be used in conjunction with UEFI Secure Boot to enhance security and provide additional functionality.

UEFI Secure Boot primarily focuses on verifying the digital signatures of firmware, boot loaders, and operating system components to ensure that only trusted and authorized software is executed during the boot process. This verification process can be carried out without the need for a TPM.

That said, a TPM can be used to securely store and manage cryptographic keys and attest to the integrity of the boot process, providing a hardware root of trust. When used in combination with UEFI Secure Boot, a TPM can enhance security by offering additional features, such as measuring the boot process, providing a secure way to store cryptographic keys, and supporting remote attestation.

In summary, while UEFI Secure Boot does not require a TPM, the two can be used together to provide a more secure boot environment.

Enclave `(230418.SGX.yaml)`

Question: `[09:36]` What is enclave in Intel SGX?`(SGX.0418.md)`

Answer:

An enclave in the context of Intel Software Guard Extensions (SGX) is a protected region of memory within the application's address space. This region is isolated and secured from other processes running on the same system, including privileged processes and the operating system itself. The primary purpose of an enclave is to provide a secure environment for executing sensitive code and processing sensitive data.

When an application creates an enclave, the SGX hardware creates a secure container that encrypts the enclave's memory. This encrypted memory can only be accessed by the enclave's code, and even the operating system or hypervisor cannot directly access it. Enclaves also have their own isolated execution environment, which helps protect against side-channel attacks and other threats.

Developers can use Intel SGX to create enclaves for various purposes, such as secure key management, encrypted data storage, or privacy-preserving computation. Enclaves are useful for protecting sensitive code and data from potential threats, ensuring the confidentiality and integrity of the data being processed.

Question: `[09:41]` Which key is encrypt memory encrypted by?`(SGX.0418.md)`

Answer:

In Intel SGX, the memory of an enclave is encrypted using a key called the Memory Encryption Engine Key (MEE-K). This key is hardware-generated and unique to each processor. The MEE-K is not accessible to software, ensuring the confidentiality and integrity of the enclave's memory.

Question: `[09:43]` How does CPU run code in the encrypted memory in enclave? `(SGX.0418.md)`

Answer:

When the CPU executes code inside an enclave, it decrypts the memory contents on-the-fly using the Memory Encryption Engine (MEE) integrated into the CPU. This process is transparent to the software running inside the enclave.

When data is read from the enclave memory, the MEE decrypts it before passing it to the CPU for processing. When data is written back to the enclave memory, the MEE encrypts it before storing it. This encryption and decryption process is done in real-time and ensures that the code and data inside the enclave are protected from unauthorized access or tampering, even if an attacker has physical access to the system memory.

Question: `[17:16]` Can a program in an enclave see outside? `(SGX.0418.md)`

Answer:

A program running inside an Intel SGX enclave cannot directly access memory outside the enclave. The enclave's memory is isolated from the rest of the system, including other processes and even the operating system. This isolation helps protect the confidentiality and integrity of the data and code within the enclave.

However, an enclave program can communicate with the outside world through well-defined interfaces. The enclave can expose secure entry points, called e-calls, that allow the application outside the enclave to call functions within the enclave. Similarly, the enclave can make o-calls to interact with the outside world, for example, to request external services or perform I/O operations.

These interactions must be carefully designed to avoid leaking sensitive information or exposing vulnerabilities. Data passed between the enclave and the external environment should be appropriately encrypted and authenticated to protect it from unauthorized access or tampering. Additionally, the enclave's exposed interfaces should be minimal and thoroughly reviewed to ensure their security.

MRSIGNER `(230418.SGX.yaml)`

Question: `[09:52]` What is MRSIGNER? `(SGX.0418.md)`

Answer:

MRSIGNER is a term used in the context of Intel SGX (Software Guard Extensions) to refer to the measurement of the signer of an enclave. It is a 256-bit hash value that represents the public key of the signer who created and signed the enclave. This value is stored within the enclave's metadata and is part of the enclave's identity.

MRSIGNER is used during the attestation process to ensure that the enclave was created by a trusted party. During attestation, the MRSIGNER value is checked against a list of trusted signer public keys to determine whether the enclave's creator is trusted. If the MRSIGNER matches one of the trusted public keys, the attestation can proceed, indicating that the enclave is genuine and created by a trusted party.

Question: `[10:19]` What is enclave identity described in the paper? `[error]` `(Papers.SGX.0419.md)`

Answer:

In the paper, the term "enclave identity" refers to the unique identifier associated with an enclave. The enclave identity is a combination of the MRENCLAVE value, which represents the enclave's code and data, and the MRSIGNER value, which represents the enclave's signer or the entity that signed the enclave.

The enclave identity is used in various security policies, such as attestation and sealing. For example, when an enclave wants to attest its identity to a remote party, it provides a REPORT containing the MRENCLAVE and MRSIGNER values as part of the attestation process. Similarly, when sealing data, the sealing policy may be based on the enclave identity to ensure that the sealed data can only be accessed by specific enclaves with matching MRENCLAVE and/or MRSIGNER values.

In summary, the enclave identity in the paper is a combination of MRENCLAVE and MRSIGNER values that uniquely identify an enclave, and it is used in various security policies and operations related to attestation and sealing.

Question: `[10:34]` Who does generate MRSIGNER? `(SGX.0418.md)`

Answer:

MRSIGNER is generated by the enclave's developer or creator. The developer creates the enclave code and then signs it using their private key. The MRSIGNER value is a hash of the public key that corresponds to the private key used for signing the enclave.

During the build process, the enclave's code, data, and metadata are combined, and a measurement of the enclave's contents (MRENCLAVE) is calculated. The developer then signs the measurement and other metadata with their private key, creating a signature. The corresponding public key is hashed to generate the MRSIGNER value, which is stored as part of the enclave's metadata.

When the enclave is loaded and initialized, the MRSIGNER value is checked during the attestation process to verify the enclave's origin and ensure that it was created by a trusted party.

Question: `[10:38]` So the MRSIGNER value is generated in offline not at runtime. Right?`(SGX.0418.md)`

Answer:

Yes, that's correct. The MRSIGNER value is generated offline during the enclave build process, not at runtime. The enclave developer signs the enclave with their private key, and the corresponding public key is hashed to generate the MRSIGNER value. This value is then stored as part of the enclave's metadata.

During runtime, the attestation process uses the MRSIGNER value to verify the enclave's origin and ensure that it was created by a trusted party. The value is not generated or altered during runtime.

Question: `[10:39]` Where is the private key stored? `(SGX.0418.md)`

Answer:

The private key used to sign an enclave is typically stored securely by the enclave developer or their organization. It is crucial to keep the private key confidential to prevent unauthorized access or modification of the enclave software.

In practice, private keys may be stored in various secure storage solutions, such as hardware security modules (HSMs), encrypted storage, or secure key management systems. The choice of storage depends on the specific security requirements and policies of the organization responsible for the enclave.

SIGSTRUCT `(230418.SGX.yaml)`

Question: `[10:59]` Please explain the sentence in the paper: "The enclave builder presents the hardware with an RSA signed enclave certificate (SIGSTRUCT)" and list the steps. `(Papers.SGX.0419.md)`

Answer:

The sentence "The enclave builder presents the hardware with an RSA signed enclave certificate (SIGSTRUCT)" describes a step in the process of creating, loading, and initializing an enclave in the Intel SGX architecture. Let's break down the steps involved:

1. Enclave Creation: The enclave developer writes the enclave code and prepares it for execution in a secure environment provided by the Intel SGX.
2. Enclave Signing: The enclave developer or a trusted authority signs the enclave using a private key, generating a

SIGSTRUCT, which is an RSA-signed enclave certificate. The SIGSTRUCT contains important information about the enclave, such as the MRENCLAVE (measurement of the enclave code), MRSIGNER (measurement of the enclave signer), and various attributes.

3. Enclave Loading: The enclave builder prepares the enclave for loading into the SGX-enabled hardware. This includes setting up the memory layout, copying the enclave code into the secure enclave memory, and preparing any necessary data structures.
4. Presenting SIGSTRUCT to Hardware: In the step described by the sentence, the enclave builder presents the hardware with the SIGSTRUCT. This step is necessary for the hardware to verify the authenticity and integrity of the enclave.
5. Verification by Hardware: The Intel SGX hardware verifies the SIGSTRUCT by checking the RSA signature, the MRENCLAVE, the MRSIGNER, and other attributes. If the verification is successful, the hardware determines that the enclave is authentic and has not been tampered with.
6. Enclave Initialization: If the verification is successful, the hardware initializes the enclave, setting up the necessary runtime environment for the enclave code to execute securely.
7. Secure Execution: Once the enclave is initialized, it can begin secure execution within the protected environment provided by the Intel SGX architecture.

Keys `(230418.SGX.yaml)`

Question: `[11:19]` What is Sealing Key? `(Papers.SGX.0419.md)`

Answer:

A Sealing Key is a cryptographic key used in Intel SGX (Software Guard Extensions) to protect sensitive data in the context of an enclave. The purpose of the sealing key is to encrypt and decrypt (or “seal” and “unseal”) the data, ensuring that it remains confidential and can only be accessed by the enclave that generated the sealing key or other authorized enclaves.

Sealing keys are derived from a combination of the enclave’s identity (MRENCLAVE or MRSIGNER) and the platform-specific key material. This ensures that the sealing keys are unique to a particular enclave and the platform it is running on.

In the Intel SGX architecture, sealing keys are used for the following purposes:

1. Confidentiality: Ensuring that the sensitive data remains confidential and is not accessible to unauthorized entities, including other enclaves, applications, or the operating system.
2. Integrity: Protecting the sensitive data from unauthorized modifications and ensuring that it remains consistent and unaltered.
3. Binding: Tying the sensitive data to a specific enclave or group of enclaves, ensuring that only authorized enclaves can access the data.

The sealing and unsealing processes are critical to the overall security of Intel SGX, as they ensure that sensitive data remains protected even when it is stored outside the secure enclave memory or transmitted between enclaves.

Sealing Identity `(230418.SGX.yaml)`

Question: `[11:30]` How does “Sealing to the Sealing Identity” work? Please list steps. `(Papers.SGX.0419.md)`

Answer:

“Sealing to the Sealing Identity” refers to the process of encrypting data with a sealing key that is derived from the enclave’s identity. This ensures that the sealed data can only be accessed by the same enclave or other authorized enclaves that share the same identity. Here are the steps involved in this process:

1. Determine the Sealing Identity: The sealing identity is typically a combination of the MRENCLAVE (measurement of the enclave code and data) or MRSIGNER (measurement of the enclave signer) values. It is used as the basis for generating the sealing key.

2. Generate the Sealing Key: The enclave requests a sealing key from the Intel SGX hardware using the EGETKEY instruction. The key is derived from the sealing identity, the platform-specific key material, and the enclave's attributes. This ensures that the sealing key is unique to the enclave and the platform it runs on.
3. Encrypt the Data: The enclave uses the sealing key to encrypt the sensitive data. This process is known as "sealing." It ensures that the data remains confidential and can only be accessed by the enclave that generated the sealing key or other authorized enclaves.
4. Store the Sealed Data: The sealed data can be stored outside the enclave, either in the application's untrusted memory or in a persistent storage location such as a disk file. Since the data is encrypted, it remains protected even when stored outside the secure enclave memory.
5. Unseal the Data: When the enclave or another authorized enclave with the same identity needs to access the sealed data, it follows a similar process to generate the sealing key using the EGETKEY instruction. With the sealing key, the enclave can decrypt and access the sealed data.
6. Verify Integrity: Optionally, the sealing process can also include a message authentication code (MAC) or digital signature to ensure the integrity of the sealed data. During the unsealing process, the enclave verifies the MAC or digital signature to ensure that the data has not been tampered with.

By following these steps, "Sealing to the Sealing Identity" ensures that sensitive data remains confidential and can only be accessed by authorized enclaves that share the same identity. This provides a secure way to store and share data between enclaves and protect it from unauthorized access or tampering.

Question: `[11:32]` Please compare "Sealing to the Enclave Identity" and "Sealing to the Sealing Identity" in a table. `(Papers.SGX.0419.md)`

Answer:

Here is a comparison table of "Sealing to the Enclave Identity" and "Sealing to the Sealing Identity":

Aspect	Sealing to the Enclave Identity	Sealing to the Sealing Identity
Identity Basis	MRENCLAVE	MRENCLAVE or MRSIGNER
Enclave Measurement	Unique to each enclave build	Shared by same signer or enclave
Key Derivation	Based on MRENCLAVE	Based on MRENCLAVE or MRSIGNER
Use Case	Specific enclave instance	Enclaves with same identity
Data Accessibility	Limited to the specific enclave	Accessible by authorized enclaves
Flexibility	Lower, tied to a single enclave	Higher, shared among same identity
Software Update Considerations	Need to migrate data to new MRENCLAVE	Easier with same MRSIGNER

In summary, "Sealing to the Enclave Identity" provides more restrictive access to sealed data, ensuring that it can only be accessed by a specific enclave instance. This method is suitable for scenarios where data must be strictly protected and accessible only to that specific enclave.

On the other hand, "Sealing to the Sealing Identity" allows for more flexibility in data access among enclaves with the same identity, which can be based on MRENCLAVE or MRSIGNER. This method is suitable for scenarios where multiple enclaves need to access shared data or when enclaves have a common signer, such as when dealing with software updates or versions.

EPID `(230418.SGX.yaml)`

Question: `[14:20]` Does Intel EPID require TPM? `(SGX.0418.md)`

Answer:

Intel EPID (Enhanced Privacy ID) does not require a Trusted Platform Module (TPM) to function. EPID is a separate privacy-preserving technology designed for remote attestation and device authentication in a scalable and anonymous manner.

Although both Intel EPID and TPM provide security features, they serve different purposes and can be used independently or in conjunction with each other, depending on the specific use case and security requirements of a system.

Question: `[14:26]` How is EPID keys enrolled into a device? `(SGX.0418.md)`

Answer:

The EPID keys are enrolled into a device during its manufacturing process. Here’s an outline of the typical EPID key enrollment process:

1. Device manufacturer generates an EPID private key and public key pair for each device during the manufacturing process.
2. The device manufacturer contacts the EPID Issuer (which is responsible for managing EPID keys). The manufacturer provides the device’s public key and a unique device identifier.
3. The EPID Issuer validates the authenticity of the device manufacturer and the device public key. After validation, the EPID Issuer issues a certificate (also known as an EPID signature) that binds the device’s public key to its unique identifier. The certificate serves as proof that the device’s public key is part of the EPID group.
4. The device manufacturer then securely stores the EPID private key and the corresponding certificate within the device, typically in a protected storage area.

The enrollment process ensures that each device has a unique EPID key pair and a valid certificate. The private key is securely stored within the device and is used for signing attestation requests, while the public key can be used by remote parties to verify the authenticity of the device’s attestation signatures.

Slideshow - Intro to Intel SGX `(230420.Slide.SGX.yaml)`

Question: `[14:36]` Below slideshow is “Intro to Intel SGX” `(Slides.SGX.0420.md)`

Answer: … …

My Questions `(230420.Slide.SGX.yaml)`

Question: `[14:39]` Why is Intel SGX not suited for password managers? `(Slides.SGX.0420.md)`

Answer:

Intel SGX is not well-suited for password managers primarily because of its limitations in handling input/output (I/O) operations. Password managers usually require user interaction for entering master passwords, copying generated passwords, and auto-filling login forms. These actions involve I/O operations that are outside the scope of the enclave’s protected environment.

In addition, Intel SGX is vulnerable to certain side-channel attacks, which can potentially leak sensitive information, such as passwords or encryption keys, by analyzing the access patterns of the CPU cache or other hardware resources. While Intel SGX provides strong isolation and protection against software-based attacks, it cannot completely secure applications that require sensitive I/O operations, such as password managers or banking apps.

However, this does not mean that Intel SGX is entirely useless for such applications. It can still be used to enhance specific aspects of these applications, like securing the storage and processing of sensitive data within the enclave. But, due to the I/O limitations and side-channel vulnerabilities, it is not a comprehensive solution for securing password managers or similar applications.

Question: `[15:01]` What does “size of TCB” mean? `(Slides.SGX.0420.md)`

Answer:

“Size of TCB” refers to the size of the Trusted Computing Base. The Trusted Computing Base is a term used in computer security to describe the set of all hardware, firmware, and software components that are critical to a system’s security. The TCB is responsible for enforcing the security policies of a system, and its integrity is crucial to maintain the overall security of the system.

The size of the TCB is an important consideration because it generally correlates with the potential attack surface and the complexity of the system. A smaller TCB typically indicates a more focused and manageable set of components, which can be more easily analyzed, verified, and secured. On the other hand, a larger TCB can introduce more potential vulnerabilities and make it more difficult to ensure the security of the system.

In the context of hardware security mechanisms, like Intel SGX, a smaller TCB may provide stronger protection against attacks, as there are fewer components that could be exploited. Reducing the size of the TCB is one of the primary goals in designing secure systems, as it helps to minimize the attack surface and the potential for security vulnerabilities.

Question: `[15:06]` What is “accountable decryption”? `(Slides.SGX.0420.md)`

Answer:

Accountable decryption is a concept in cryptography that aims to ensure that the decryption of encrypted data can be traced back to a specific party or entity. This concept is often employed in systems where multiple parties have access to the decryption keys, and it’s important to know who decrypted the data for various reasons, such as security, auditing, or legal compliance.

In an accountable decryption scheme, the decryption process generates a unique proof or decryption tag that can be used to identify the party responsible for the decryption. This proof or tag is often linked to the decryption key or the party’s identity in a way that cannot be forged or tampered with. This allows for the attribution of decryption actions to specific parties, enabling better control and monitoring of access to sensitive information.

Accountable decryption can be implemented through various cryptographic techniques, including threshold cryptography, digital signatures, and zero-knowledge proofs. The choice of technique depends on the specific requirements and the overall system design.

Question: `[16:03]` What is endorsement key? `(Slides.SGX.0420.md)`

Answer:

An endorsement key (EK) is a unique, private cryptographic key that is embedded within a security device or module, such as a Trusted Platform Module (TPM) or a hardware security module (HSM). This key is generated during the manufacturing process and is paired with a corresponding public key. The private portion of the endorsement key is never exposed outside of the secure environment and is used to prove the authenticity and integrity of the device.

The public portion of the endorsement key is typically used along with a certificate issued by the manufacturer, which serves as a proof of the device’s authenticity. This certificate is called the endorsement certificate and is used to establish trust in the device’s identity during secure communications or attestation processes.

In summary, the endorsement key is a fundamental component of hardware-based security solutions, as it helps to establish trust and verify the authenticity of a device or module in secure communications and attestation scenarios.

Launch Enclave `(230420.Slide.SGX.yaml)`

Question: `[15:10]` What is “Launch Enclave”? `(Slides.SGX.0420.md)`

Answer:

A Launch Enclave is a secure environment in the context of Intel Software Guard Extensions (SGX), a set of security-related instruction codes that are built into some Intel processors. The Launch Enclave is responsible for verifying and controlling the launch of other secure enclaves within the SGX environment.

In Intel SGX, enclaves are isolated, protected memory regions that run sensitive code and store sensitive data. They are designed to protect the confidentiality and integrity of this information from external threats, even in the presence of compromised operating systems or hardware.

The Launch Enclave is a special enclave provided by Intel that plays a crucial role in the SGX ecosystem. When an application wants to create and run a new enclave, it must first request the Launch Enclave to create a launch token. The Launch Enclave verifies the new enclave's identity and attributes, and if everything is in order, it creates the launch token. This token is then used by the application to launch the new enclave.

By controlling the launch of other enclaves, the Launch Enclave helps maintain the security and integrity of the SGX environment. It ensures that only trusted and authorized enclaves can be created and executed, thus preventing rogue enclaves from compromising the security of the system.

Question: `[15:11]` Is Launch Enclave a software? `(Slides.SGX.0420.md)`

Answer:

Yes, the Launch Enclave is a software component provided by Intel as part of the Intel Software Guard Extensions (SGX) framework. It is designed to run within the secure environment of an Intel SGX-enabled processor.

The Launch Enclave is a specific enclave with predefined functionality that manages the creation and launch of other secure enclaves within the SGX environment. By acting as a gatekeeper, the Launch Enclave helps ensure that only authorized and trusted enclaves are executed, maintaining the overall security of the system.

Question: `[15:12]` Does Launch Enclave run in OS? `(Slides.SGX.0420.md)`

Answer:

The Launch Enclave runs within the secure environment provided by Intel SGX, which is a set of security-related instruction codes built into Intel processors. While it is not part of the operating system (OS) itself, it does interact with the OS through a set of APIs and drivers.

The OS plays a role in managing the resources and interactions between the untrusted, non-secure part of the system and the secure enclave environment. In this sense, the Launch Enclave does run on a system with an OS, but it operates within the protected enclave environment provided by the Intel SGX, isolated from the untrusted portion of the OS.

Question: `[15:13]` Does the binary of Launch Enclave reside in BIOS or ME? `[title]` `(Slides.SGX.0420.md)`

Answer:

The binary of the Launch Enclave does not reside in the BIOS or the Intel Management Engine (ME). The Launch Enclave is a part of the Intel SGX software stack, which is installed on the operating system. The Intel SGX software stack includes the Intel SGX SDK and Platform Software (PSW).

The Launch Enclave is loaded as part of the Intel SGX runtime system when an application wants to create a new enclave. The Launch Enclave binary is signed by Intel, and its primary purpose is to verify and manage the launch of other enclaves, ensuring that they have been signed with an authorized launch key.

While the Launch Enclave is not directly part of the BIOS or ME, both the BIOS and ME play a role in enabling Intel SGX on the system. The BIOS must support and properly configure the hardware for Intel SGX, and the Intel ME is responsible for managing security features and providing cryptographic services to the system, including those related to Intel SGX.

Question: `[15:16]` Is Launch Enclave provided by another vendor (No Intel)? `(Slides.SGX.0420.md)`

Answer:

No, the Launch Enclave is provided by Intel as part of the Intel SGX software stack. Intel is responsible for signing the Launch Enclave binary and distributing it as a part of the Intel SGX SDK and Platform Software (PSW). This is an important aspect of the SGX security model, as it allows Intel to maintain control over the trustworthiness of the Launch Enclave and ensure that only authorized enclaves can be launched on SGX-enabled systems.

Privacy CA `(230420.Slide.SGX.yaml)`

Question: `[16:14]` How does the Solution 1 “Privacy CA for provisioning ak” work? Please list the steps. `(Slides.SGX.0420.md)`

Answer:

“Privacy CA for provisioning ak” refers to using a Privacy Certification Authority (Privacy CA) to provision an Attestation Key (ak) for a TPM (Trusted Platform Module) while maintaining the privacy of the device. The process allows a TPM to interact with third parties without disclosing its unique identity. Here are the steps involved in this solution:

1. TPM generates an Attestation Key (ak) pair: The TPM generates a public-private key pair intended for use in attestation. The private part of the key remains within the TPM, while the public part is shared externally.

2. TPM creates an Attestation Key Certificate Request: The TPM generates a certificate signing request (CSR) for the public portion of the Attestation Key. This request typically includes information about the TPM, such as its public Endorsement Key (EK) and the desired ak.

3. Privacy CA verifies TPM identity: The Privacy CA receives the CSR and verifies the authenticity of the TPM using the provided public EK and the corresponding endorsement certificate issued by the TPM manufacturer. The Privacy CA may also check other information, such as the platform’s conformance to certain security standards.

4. Privacy CA issues an Attestation Key Certificate: Once the Privacy CA is satisfied with the TPM’s authenticity, it signs the CSR and generates an Attestation Key Certificate. This certificate is then sent back to the TPM.

5. TPM receives and stores the Attestation Key Certificate: The TPM securely stores the Attestation Key Certificate, linking the ak to a trusted Privacy CA. This certificate can be used in future attestation processes to establish trust with third parties.

In summary, using a Privacy CA for provisioning an Attestation Key (ak) involves generating an ak pair within the TPM, creating a CSR for the public ak, verifying the TPM’s identity by the Privacy CA, issuing an Attestation Key Certificate, and securely storing this certificate in the TPM. This process allows the TPM to prove its authenticity and integrity while preserving its privacy.

Question: `[16:18]` What is “Privacy CA”? `(Slides.SGX.0420.md)`

Answer:

A Privacy Certification Authority (Privacy CA) is a trusted third-party entity that issues digital certificates in a privacy-preserving manner. Privacy CAs play a crucial role in remote attestation processes for secure hardware platforms, such as Intel SGX (Software Guard Extensions) and Trusted Platform Modules (TPMs).

The main purpose of a Privacy CA is to verify the authenticity and integrity of a secure hardware platform or an enclave (in the case of Intel SGX) without revealing sensitive information about the platform or enclave to other parties. The Privacy CA is responsible for validating the identity and measurements of the secure platform, and then issuing a digital certificate as proof of the platform’s trustworthiness.

In this process, the Privacy CA acts as an intermediary between the secure platform and the external party seeking to establish a trusted communication channel. The Privacy CA helps to maintain the confidentiality of the platform’s sensitive information while still providing the necessary level of assurance to the external party.

For example, in the context of Intel SGX, the Privacy CA is the Intel Attestation Service (IAS). The IAS issues digital certificates for Intel SGX enclaves after verifying their authenticity and integrity based on measurements provided by the enclave during the remote attestation process.
