ETSI GR SAI 009 V1.1.1 (2023-02)

Securing Artificial Intelligence (SAI); Artificial Intelligence Computing Platform Security Framework

Disclaimer

The present document has been produced and approved by the Securing Artificial Intelligence (SAI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/SAI-009

Keywords
artificial intelligence, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Securing Artificial Intelligence (SAI).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Artificial Intelligence is evolving very fast. It has been deployed everywhere trying to improve the quality of daily life. The compromise of AI systems will directly affect a vast number of people and can cause severe results. In an AI system, AI computing platform acts as the infrastructure for AI applications that provides resource and executing environments. Therefore, it is very important to study the security of AI computing platform. The reasons are two-folds:

- The security of AI computing platform is the baseline guarantee for AI system. If the platform is compromised, all of the applications running on it will be controlled by malicious attackers.
- Common security requirements from different AI applications are best resolved by AI computing platform that will apparently improve the protection levels for AI application and bring convenience for relevant stakeholders during development, use and maintenance of AI applications.

The present document first studies the role of AI computing platform in AI systems, its common structure and security requirements in AI systems. Secondly, the security components and interaction between these components that form the security framework of AI computing platform are studied. Last but not the least, the mechanisms which guarantee the security of the platform itself and provide services for relevant stakeholders in AI systems are studied in detail. The aim of the present document is to set out a reference security framework for AI computing platform developer and users to mitigate the security threats against AI systems in a cooperatively manner.

The study uses ETSI GR SAI 006 [i.3] as a starting point for hardware aspects and avoids overlap.

1 Scope

The present document describes a security framework of AI computing platform containing hardware and basic software to protect valuable assets like models and data deployed on AI computing platform when they are used in runtime or stored at rest. The security framework consists of security components in AI computing platform and security mechanisms executed by security components in the platform. By specifying the security framework, an AI computing platform can be consolidated against the relevant attack and can provide security capabilities to facilitate the stakeholders in AI systems to better protect the valuable assets (model/data) on an AI computing platform.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GR SAI 004: "Securing Artificial Intelligence (SAI); Problem Statement".

[i.2] ETSI GR SAI 005: "Securing Artificial Intelligence (SAI); Mitigation Strategy Report".

[i.3] ETSI GR SAI 006: "Securing Artificial Intelligence (SAI); The role of hardware in security of AI".

[i.4] Pingchuan Ma, Stavros Petridis, Maja Pantic: "Detecting adversarial attack on audiovisual speech recognition", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, Toronto, ON, Canada.

NOTE: Available at https://arxiv.org/pdf/1912.08639.pdf.

[i.5] Celia Cintas, Skyler Speakman, Victor Akinwande, William Ogallo, Komminist Weldemariam, Srihari Sridharan and Edward Mcfowland: "Detecting Adversarial Attacks via Subset Scanning of Autoencoder Activations and Reconstruction Error", Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence.

NOTE: Available at https://www.ijcai.org/proceedings/2020/0122.pdf.

[i.6] Mika Juuti, Sebastian Szyller, Samuel Marchal, N. Asokan: "PRADA: Protecting Against DNN Model Stealing Attacks", 2019 IEEE European Symposium on Security and Privacy (EuroS&P).

[i.7] Ren Pang, Xinyang Zhang, shouling Ji, Xiapu Luo, Ting Wang: "AdvMind: Inferring Adversary Intent of Black-Box Attacks", Proceedings of the 26th ACM SIGKDD, 2020.

NOTE: Available at https://arxiv.org/pdf/2006.09539.pdf.

[i.8] ETSI GS NFV-SEC 009: "Network Functions Virtualisation (NFV); NFV Security; Report on use cases and technical approaches for multi-layer host administration".

[i.9] ETSI GS NFV-SEC 012: "Network Functions Virtualisation (NFV) Release 3; Security; System architecture specification for execution of sensitive NFV components".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

AI asset: anything that has value to the organization, its business operation and its continuity relevant to AI

NOTE: Model, dataset and training script belongs to AI asset category.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AE	AutoEncoder
AI	Artificial Intelligence
API	Application Programming Interface
BIOS	Basic Input Output System
BMC	Baseboard management controller
CPU	Central Processing Unit
CVE	Common Vulnerabilities & Exposures
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
GPU	Graphics Processing Unit
HBRT	Hardware Based Root of Trust
HDD	Hard Disk Drive
HMEE	Hardware-Mediated Execution Enclave
HSM	Hardware Security Module
HTTPS	Hypertext Transfer Protocol Secure
HUK	Hardware Unique Key
IO	Input Output
IOT	Internet Of Things
JTAG	Joint Test Action Group
NIC	Network interface card
NPU	Neural network Processing Unit
OS	Operating System
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect express
REE	Rich Execution Environment
RoT	Root of Trust
RTS	Root of Trust for Storage
RTR	Root of Trust for Report
RPO	Recovery Point Objective
RTO	Recovery Time Objective
SDK	Software Development Kit
SE	Secure Element
SIM	Subscriber Identity Module
SMTP	Simple Mail Transfer Protocol
SoC	System on Chip
SSD	Solid State Disk
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPM	Trusted Platform Module

USB	Universal Serial Bus
VM	Virtual Machine

4 Convention Description

4.1 Notation

For the purpose of the present document, the following notations apply in clause 9:

- [Security component] stands for the "security components" described in clause 7 that are involved in the interactive procedures.
- <Service-based interface> stands for the "Service-based interface" described in clause 8.2 that is utilized to deliver relevant information or data between an AI computing platform and platform users.

5 Overview

5.1 AI computing platform description

An AI computing platform is defined as a fundamental platform to provide an execution environment and related resources in an AI system as shown in Figure 1. In particular, the platform provides required resource for AI model and AI application including computing resource, storage resource and networking resource; meanwhile it also provides an execution environment for AI models and AI applications to be able to operate properly and stably including a fundamental software framework, various kinds of libraries and different hardware drivers. On the other hand, for important assets like training dataset, testing dataset, inference data and training script, an AI computing platform also provides storage resource to guarantee the secure storage of these assets.

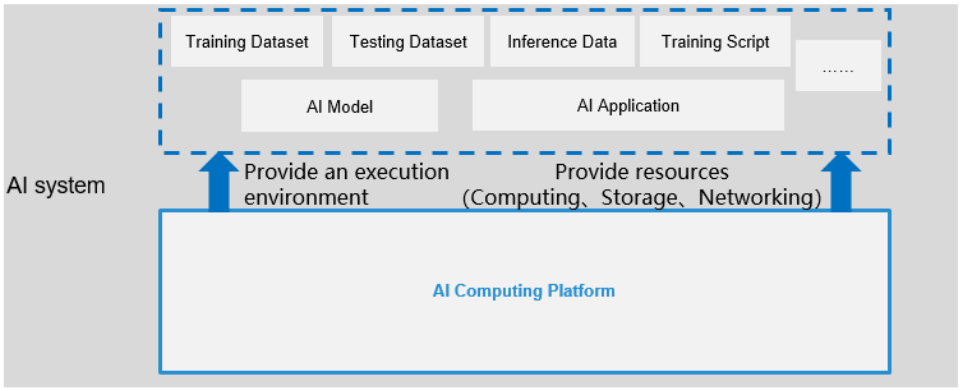

Figure 1: Role of AI computing platform in an AI system

The structure of an AI computing platform is constructed in three layers and illustrated in Figure 2. The hardware layer and basic software layer form the core/basic part of an AI computing platform while the application enabling layer is the extended/optional part as the components in the application enabling layer vary depending on different application scenarios and most components in this layer could be independent of underlying layers.

To be specific, the first layer contains different kinds of hardware to provide computing resource, storage resource and networking resource. Computing hardware includes general computing device (CPU) and AI-specific computing device like GPU and NPU. Storage hardware includes memory and non-volatile storage like hard disk drive and solid state disk. Networking hardware includes network interface card like smart IO card or network device connected by PCIe. The second layer is the basic software layer consisting of operating system, chip-level SDK, virtualization component for VM or containers, and some other software which directly manage the hardware devices and act as an interface to open up the hardware layer capabilities to AI application providers and users. The software in this layer are strongly related with hardware like OS, GPU/NPU accelerating libraries and GPU/NPU operating framework. The top layer in AI computing platform is the application enabling layer which contains different types of deep learning framework, application-level SDK, management module and so on. Herein, application-level SDK is a set of software components which has limited or no relation with hardware to enable the development, deployment and management of AI application; the management module provides management function for AI computing platform and deals with the cooperation in some application scenarios if needed like cluster computing, cloud computing, cloud-edge collaboration, etc.

EXAMPLE 1: Hypervisor and container engine belong to the category of virtualization software in basic software layer.

EXAMPLE 2: Tensorflow, Pytorch and Mindspore belong to the category of deep learning framework.

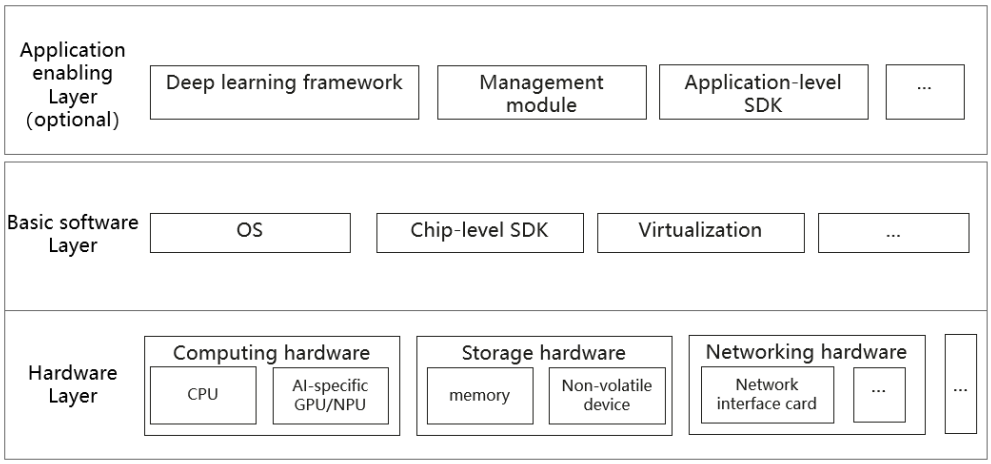

Figure 2: Structure of AI computing platform

A typical implementation of AI computing platform in common scenarios is an AI server equipped with GPU/NPU, server-grade OS and GPU/NPU related chip-level SDK.

EXAMPLE 3: GPU/NPU and the related chip-level SDKs are available on the market.

5.2 Implementation recommendations

The present document focuses on AI computing platforms that are deployed in data centres or edge computing environments. AI computing platforms in these environments have more resources (computing power and energy) for computing tasks, and for constructing security capabilities as well. In addition, these environments can involve more complex scenarios like ones supporting multi-tenant settings and remote access usage which introduce specific threats and may need corresponding mitigations. Other types of platforms like mobile phones or embedded devices capable of executing AI functionality may also refer to this security framework according to their specific conditions and security requirements.

5.3 Threats analysis of AI computing platform

As introduced in clause 5.1, an AI computing platform provides an execution environment and related resources to AI applications and AI models. Therefore, to comprehensively summarize the security requirements for an AI computing platform, a threats analysis is conducted in two aspects:

- The threats against the AI computing platform itself.
- The threats against valuable assets on the platform.

NOTE: It is very important to emphasize that the threats in Tables 1 and 2 are not exhaustive . New threats can emerge based on specific scenarios and technology development.

The first aspect of threats can be categorized and analysed as follows in Table 1. They are partitioned into several high level categories concentrated on the AI computing platform as the affected asset. Then, a detailed description is presented in each category. On the other hand, threats listed in Table 1 will directly or indirectly affect AI assets after the AI computing platform is being attacked. Direct affect can be caused by the threat directly affecting specific AI-related assets after the AI computing platform being attacked. Indirect affect is produced by the threats indirectly affecting specific AI-related assets after the AI computing platform being attacked. However, this kind of threats can create the available condition or execution environment for threat agent to implement further attacks directly on AI-related assets on the platform. The fourth column marks this information.

Table 1: Threats analysis for the AI computing platform itself

Category	No.	Threat description	Risk and negative consequence	Threat to AI assets
Physical destruction	1	The threat agent utilizes physical method to destroy the AI computing platform. Physical method can include fire, water, physical damage, etc.	AI computing platform can be destroyed and stop working. Valuable assets like business data/model/data set can be corrupted.	Direct on availability
Outsider manipulation	2	The threat agent utilizes a physical interface of the AI computing platform like USB and JTAG to implement unauthorized operation and access to the platform.	Configuration data and sensitive information can be leaked and the control permission of the platform can be gained by the threat agent to implement further attack.	Indirect
	3	The threat agent utilizes a remote interface of the AI computing platform like service interface and remote login to implement unauthorized operation and access to the platform.	Configuration data and sensitive information can be leaked and the control permission of the platform can be obtained by the threat agent to implement further attack.	Indirect
	4	The threat agent exploits known vulnerabilities of hardware and software components in the AI computing platform.	Sensitive information can be leaked. Or the control permission of the AI computing platform can be obtained by the threat agent for further attack.	Indirect
Insider manipulation	5	The threat agent utilizes a legitimate account or a legitimate asset like normal OS user account, virtual machine instance on AI platform to conduct privilege escalation to acquire more permissions than allowed.	Sensitive information can be leaked and the control permission of the platform can be obtained by the threat agent to implement further attack.	Indirect
	6	The threat agent utilizes legitimate resource/object like virtual machine, container and even PCI hardware device which it has operation permission to conduct lateral attack to other resource which belongs to other users.	Sensitive information and valuable assets of users who share resource of the AI computing platform can be leaked, tampered or stolen by malicious users on the same platform.	Indirect

Category	No.	Threat description	Risk and negative consequence	Threat to AI assets
Tampering	7	The threat agent maliciously tampers with the system clock which is maintained in the AI computing platform.	Some key subsystem running on the platform can be disrupted if the time clock is disordered. Log consequence and detail information can be disrupted so that the threat agent can utilize this disorder to avoid security audit or hide any malicious action on the platform.	Indirect
	8	The threat agent maliciously tampers with the components in the AI computing platform like firmware of devices and software in the platform.	The functionality of firmware can be modified so that the system fails. Or some malicious programs can be inserted into firmware, based on which threat agent can further control the system or steal the valuable information in the platform.	Indirect
	9	The threat agent deploys a malicious software onto the platform.	The performance of the platform can be affected like resource exhaustion.	Indirect
Data attack	10	The threat agent steals the storage device like HDD or SSD containing valuable assets if he has the permission to physically contact with AI computing platform.	Valuable assets like business data/model/data set or system configuration information can be leaked or stolen.	Direct on confidentiality
	11	The threat agent corrupts the storage of the platform by tampering or deleting the data on it.	Valuable assets like business data/model/data set or system configuration information can be corrupted or destroyed.	Direct on availability or integrity
Communication attack	12	The threat agent conducts DoS or DDoS attack to the exposed interface of the AI computing platform.	Communication availability and service accessibility of AI computing platform and the corresponding AI system it supports can be degraded or even destroyed.	Direct on availability
	13	The threat agent utilizes exposed interface to conduct an attack like injection attack and buffer overflow attack.	Sensitive information can be leaked and the control permission of the platform can be obtained by the threat agent to implement further attack.	Indirect
	14	The threat agent conducts an eavesdropping attack on the communication to and from the AI computing platform.	Sensitive information transferred to and from AI computing platform can be leaked.	Direct on confidentiality
	15	The threat agent conducts a spoofing attack in the procedure of communication to and from the AI computing platform.	Sensitive information transferred to and from the AI computing platform can be leaked and tampered by the threat agent.	Direct on confidentiality and integrity
Occupation	16	The threat agent gains complete control of the computing platform through specific physical or social engineering methods.	The platform can be fully controlled, all of the assets stored on the platform can be leaked and tampered. The input and output of the process running on the platform can be arbitrarily changed.	Direct on confidentiality, integrity and availability.

On the other hand, an AI computing platform should provide sufficient functionality to protect the assets running or stored on the platform from known threats. Threat analysis for assets on the AI computing platform is important when summarizing the security requirements on an AI computing platform. The threats against valuable assets on an AI computing platform can be categorized and analysed as follows in Table 2.

Table 2: Threats analysis for valuable assets on AI computing platform

Asset	No.	Threat description	Risk and negative consequence	Security goal affected
Model	1	The threat agent accesses the storage where the model is stored and steals the key structure information and parameter value of model in the model file.	The intellectual property of the model provider is stolen which can be a huge loss as a well-performed model costs massive resources including computing resource, dataset resource, time resource and labour resource.	Confidentiality
Model	2	The threat agent accesses the memory where the model is loaded and steals the key structure information and parameter value of the model in running.	The intellectual property of model provider is stolen which can be a huge loss as a well-performed model costs massive resources including computing resource, dataset resource, time resource and labour resource.	Confidentiality
Model	3	The threat agent accesses the storage where the model is stored and unauthorizedly changes the content of the model file like structure information and parameter value.	The original model can be changed in the aspect of the performance and therefore the changes influence the outcomes of the AI application/system which utilizes this model. Worse results can be caused when such AI applications/systems are used in safety-critical scenarios like automobile and e-health.	Integrity
Model	4	The threat agent accesses the memory where the model is loaded and unauthorizedly changes the parameters of the model file like structure information and parameter value.	The original model can be changed in the aspect of the performance and therefore the changes influence the outcomes of the AI application/system which utilizes this model. Worse results can be caused when such AI applications/systems are used in safety-critical scenarios like automobile and e-health.	Integrity
Model	5	The model is not deleted/ erased from storage correctly and thoroughly. The threat agent recovers the model from storage.	The intellectual property of the model provider is stolen which can be a huge loss as a well-performed model costs massive resources including computing resource, dataset resource, time resource and labour resource.	Confidentiality
Model	6	The threat agent forges a well-designed fake model and inserts it into the AI application/system.	The outcome of the AI application/system can be misled as the threat agent wishes. Worse results can be caused when such AI applications/systems are used in safety-critical scenarios like automobile and e-health.	Integrity
Model	7	The threat agent accesses the storage where the model is stored and deletes the model file.	For model users, the business which relies on the model can be terminated. While for model providers, it can be seen as a huge business loss since resources for the training process of the model are wasted if the model is deleted.	Availability
Model	8	The threat agent destroys the model by physical methods to the storage media of the model file.	For model users, the business which relies on the model can be terminated. While for model providers, it can be seen as a huge business loss since resources for the training process of the model are wasted if the model is deleted or destroyed.	Availability

Asset	No.	Threat description	Risk and negative consequence	Security goal affected
Training data set	9	The threat agent accesses the storage where the training data set is stored and steals the data set.	The intellectual property of data providers is stolen.	Confidentiality
Training data set	10	The threat agent accesses the storage where the training data set is stored and tampers the data set.	The model which is being trained with the data set can be negatively affected. With the tampered data set, the performance of the model can be degraded. Furthermore, the tampered data set can be used to mount poisoning attacks and backdoor attacks.	Integrity
Training data set	11	The threat agent forges a well-designed fake training data set and inserts it into the training system.	The model which is being trained with the data set can be modified. With the fake data set, the outcome of the inference based on the model can be misled. Worse results can be caused when such AI applications/systems are used in safety-critical scenarios like automobile and e-health.	Integrity
Training data set	12	The training dataset is not deleted/ erased from storage correctly and thoroughly. The threat agent recovers the model from storage.	The intellectual property of data providers is stolen.	Confidentiality
Training data set	13	The threat agent accesses the storage where the training data set is stored and deletes the model file.	For model users, the training process which utilizes the data set can be terminated. While for data providers, it can be seen as a huge business loss.	Availability
Training data set	14	The threat agent destroys the training data set by physical methods to the storage media of the model file.	For model users, the training process which utilizes the data set can be terminated. While for data providers, it can be seen as a huge business loss.	Availability
Testing data set	15	The threat agent accesses the storage where the testing data set is stored and tampers the data set.	The model which is being tested with the data set can be affected negatively. Poor model performance or critical errors in the inference outcome may not be detected during the testing.	Integrity
Inference data	16	The threat agent eavesdrops the inference data.	The sensitive information included in inference data can be leaked. In some scenarios, privacy problems can be raised.	Confidentiality
Training script	17	The threat agent accesses the storage where the training script file is stored and steals the key content in the script like loss function and related algorithm design.	The intellectual property of data providers on training script is stolen. The training script is of great significance for model provider when training. A well designed training script can accelerate the training process, improve the performance of the trained data and save costs.	Confidentiality
Training script	18	The treat agent accesses the memory where the training script is stored and tampers the key content in the script like loss function and related algorithm designing.	Longer training period and more training resource can be consumed.	Integrity
Training script	19	The threat agent accesses the storage where the training script is stored and tampers the key content in the script like loss function and related algorithm design.	Longer training period and more training resource may be consumed than expected.	Integrity

Asset	No.	Threat description	Risk and negative consequence	Security goal affected
Training script	20	The threat agent accesses the storage where the training script is stored and deletes the model file.	For model users, the training process which utilizes the training script can be terminated. While for model providers, it can be seen as a huge business loss since the development of the script needs intellectual and resource investment.	Availability
Training script	21	The threat agent destroys the training script by physical methods to the storage media of the model file.	For model users, the training process which utilizes the training script can be terminated. While for model providers, it can be seen as a huge business loss since the development of the script needs intellectual and resource investment.	Availability

5.4 Security objectives and requirements for AI computing platform

In order to mitigate the threats in clause 5.3 and provide a consolidated infrastructure for AI system, several security objectives and requirements for AI computing platform should be satisfied as follows:

- The AI computing platform should be consolidated enough to mitigate the risks to itself in various aspects like system integrity, system access control and isolation, interface protection, communication conservation, etc.

NOTE 1: A secure foundation plays a key role in a secure system. The AI computing platform acts as a resource and environmental foundation in an AI system.

- The AI computing platform should have security functionalities to protect AI-specific assets running or stored on it to mitigate the risks to these assets like model, data sets and training script.

- The AI computing platform should provide security services or functionalities for the stakeholders in the AI system to support the mitigation, detection and remediation mechanisms implemented by them to protect their assets in the platform.

NOTE 2: Security services are different from security functionalities. A security service is a tool or in the form of an interface like an API provided by the AI computing platform to the stakeholders like the model provider or AI application provider. The stakeholders need to invoke the service in the process of programming or implementing mitigation mechanisms. A security functionality is a mechanism operating inside the AI computing platform to fulfill a security operation which processes automatically in the platform and requires no stakeholders to invoke.

EXAMPLE: A cryptographic tool for model is a security service. A model provider needs to invoke the tool to encrypt or decrypt the model file. Software integrity protection is a security function. It will be automatically executed when a software is to start in the platform without the need to be invoked by stakeholders.

6 AI Computing Platform Security Architecture

6.1 Baseline security capabilities of an AI computing platform

6.1.1 Overview

In clause 6.1, the detailed security requirements for an AI computing platform to mitigate the risks to itself are described and analysed. Upon satisfying these requirements, the platform can be consolidated and regarded as a secure foundation in an AI system.

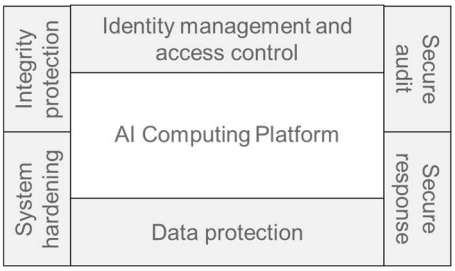

Figure 3: Baseline security capabilities of AI computing platform

As the present document focuses on security requirements and technology, the methods attempting to reduce the risks of the threat No.1 and 16 described in Table 1 are out of the scope of the present document.

6.1.2 Identity management and access control

An AI computing platform should implement identity management and access control methods as follows:

- The external physical interface of an AI computing platform like JTAG should be disabled after manufactured or access control should be implemented on the interface to prevent arbitrary access to the platform via the physical interface so that only authorized users or maintenance engineers have the access permission.
- The remote access of an account with root privilege should be forbidden in an AI computing platform.
- Fine grained role based access control should be implemented to the software running on an AI computing platform to facilitate the principle of least privilege.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of threats No. 2, 3 and 10 in Table 1.

6.1.3 Integrity protection

An AI computing platform should implement integrity protection methods as follows:

- An AI computing platform should implement secure boot to guarantee the integrity of related firmware, OS and software in the platform so that a chain of trust is established from a tamper-resistant initial code. Thus, it can be guaranteed that every component in the platform is not tampered and will operate as expected.
- An AI computing platform should support the trusted computing technology and have the capability to cooperate with hardware security chips like TPM to facilitate the mechanisms like proactive integrity measurement and remote attestation.

NOTE: The detailed procedure of some security capabilities mentioned here like secure boot and proactive integrity measurement are illustrated in clause 8.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of threats No. 7 and 8 in Table 1.

6.1.4 System hardening

An AI computing platform should implement system hardening methods as follows:

- An AI computing platform should disable unnecessary physical and logical interfaces to reduce the possible attack entrance.
- Components in an AI computing platform should cut down or unload unnecessary modules to reduce attack surface of the components.

EXAMPLE: OS can unload modules related to e-mail services like modules of SMTP server and DNS server.

- An AI computing platform should fix all the known high-risk vulnerabilities which have been exposed and listed by vulnerability databases like CVE.
- An AI computing platform should isolate the network of management plane from service data plane to ensure security problems of one plane cannot influence on the other.
- An AI computing platform should set the minimum permission for each process running on the platform. Processes should not obtain the root permission.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of threats No. 4, 5 , 6 and 13 in Table 1.

6.1.5 Data protection

An AI computing platform should implement data protection methods as follows:

- An AI computing platform should utilize a secure communication protocol to protect transmitted data to and from the platform with the help of cryptographic technology and identity authentication.

NOTE: HTTPS can be used as a secure communication protocol between two entities.

- An AI computing platform should have the capability to back up important system information like configuration parameters and valuable data, and have the capability to recover the information as needed.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of threats No. 11, 14 and 15 in Table 1.

6.1.6 Secure audit

An AI computing platform should implement secure audit methods as follows:

- An AI computing platform should have the capability to collect logs related to security and important operations in the platform like configuration changes, logging in/out, interface/resource access and utilization, etc. Thus, logs of security event and security status can be obtained in a sequential and timely manner to support the security analysis and response.
- An AI computing platform should provide event information about AI specific assets in the logs.

NOTE: This information could be very useful for the relevant stakeholders like model providers and model users in compliance with the requirements in AI ACT.

EXAMPLE: The changing of model parameters, the selection of training data sets and deletion of model related files are typical AI specific events, which could help detecting and responding to possible threats on AI assets.

- An AI computing platform should implement protection methods for log information so that that it cannot be modified, and it can only be accessed by the authorized users like the security administrator.
- An AI computing platform should provide interface/mechanism for secure transfer of logs to a third platform or entity for analysis, backup and better protection.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of threats No. 2, 3, 5 to 8, 10, 12 and 13 in Table 1.

6.1.7 Secure response

An AI computing platform should implement secure response methods as follows:

- An AI computing platform should implement a network traffic detection mechanism on network interfaces and further implement a control mechanism to regulate the traffic and manage abnormal traffic.
- An AI computing platform should implement an intrusion detection mechanism to timely identify network attacks at the initial stage and support secure responses against these attacks.

Upon the implementation of these methods, an AI computing platform can significantly reduce the risk of No. 3, 7 to 9, 12 and 13 in Table 1.

6.1.8 Resilience

An AI computing platform should have resilience mechanisms as follows:

- An AI computing platform should implement adjustment or protection mechanism for AI-specific accelerators. The mechanism should monitor real-time parameters of GPU or NPU like temperature, power, etc. to estimate the health status of these accelerators. If abnormal status found, an AI computing platform should automatically change certain configurations like degrading the performance or reallocating workflows on these accelerators to avoid further system faults.
- An AI computing platform should have a safe mode which executes a set of minimally required tasks when it has been attacked or disrupted. The required tasks should include at least platform recovery procedure which brings the platform back to a predefined normal state.

Upon the implementation of these mechanisms, an AI computing platform can withstand attacks or accidents and recover to an expected state. In this way, the consistency and stability of AI applications running on it can be maintained.

6.2 Security functions/services of an AI computing platform

6.2.1 Overview

In clause 6.2, security functions/services provided by an AI computing platform are described. Upon utilizing these functions/services via predefined interfaces, platform users like model providers and AI application providers can effectively reduce the risks on their AI assets like models and datasets. The content listed in clause 6.2 only introduces the effect of each security function/service while the detailed mechanisms and interfaces of these functions/services are described in clause 8.

6.2.2 AI assets confidentiality protection

6.2.2.1 AI assets encryption/decryption

An AI computing platform should provide encryption/decryption service for stakeholders to protect their important AI assets like models and datasets. Stakeholders can utilize the service to protect their AI assets from stealing and eavesdropping at storage and transfer phase. The service should satisfy the following requirements in order to fulfil the protection task:

- The service should support multi-tenant scenarios where different users utilize different keys to implement encryption and decryption. Different users may only manage their own keys and should not access and obtain each other's keys.

NOTE 1: This is particularly important in a data centre environment where the AI computing platform is shared.

- The service should support authorization management for users to permit a third party to decrypt their AI assets for use. Decryption privilege may be granted based on various conditions like time duration and identity of decryption operator.

NOTE 2: This is particularly important in some scenarios where different stakeholders need to exchange information exclusively or a model provider grants the usage permission of a well-trained model to the users.

- The service may support hardware-bound decryption so that only the platform with designated AI-specific accelerator has the permission to decrypt the AI assets.

NOTE 3: It is very useful in edge computing scenarios where model providers only allow their well-trained models to be decrypted and deployed on designated edge devices for inference so that intellectual property of models can be protected.

6.2.2.2 AI-specific computing resource isolation

When deployed in a data centre environment where computing resource can be shared by different tenants, an AI computing platform should provide an AI-specific computing resource isolation function so that AI assets from different tenants can be isolated from each other. Utilizing this function, an AI-specific computing device like GPU and NPU can simultaneously allocate several isolated computing and storage resource for different tenants sharing the device. The function should satisfy the following requirements in order to fulfil the protection task:

- The resource allocated to one tenant should only be used by this tenant. Other tenants should not have access to or use the resource. In this way, AI assets of different tenants can securely run or be stored on the same AI-specific computing device for training and inference without breaching confidentiality.

- The resource allocated to one tenant should not exceed its pre-configured quota so that security risks can be quarantined within the certain scope and the blast radius can be minimized.

6.2.3 AI related log protection

An AI computing platform should provide a protection service for stakeholders to guarantee that AI related logs cannot be tampered or deleted, or to provide a mechanism to verify whether logs stored in platform have not been changed in content or sequence. These AI related logs can be used to support liability analysis and to improve model explicability and transparency. Usually, these logs are generated from operations of AI applications or from the procedures of data acquisition, data curation, model training, etc. The service should satisfy the following requirements in order to fulfil the protection task:

- The service should provide interfaces for stakeholders to send AI related logs to the platform for log storage and protection. In this way, AI computing platform can collect or receive logs from different components in an AI system.

NOTE 1: Stakeholders include model providers, AI application providers and AI application users.

NOTE 2: One form of interfaces is an API through which stakeholders can proactively or periodically send logs to the platform.

- The service should utilize a cryptography mechanism to verify the integrity of the logs stored in the platform. When integrity is breached, the platform should send alerts. Other mechanisms may also be used to handle the breach.

NOTE 3: The Merkle tree structure with a hash algorithm is a suitable cryptography mechanism for integrity protection especially when the amount of logs is very large.

- An AI computing platform should utilize an access control method to restrict the access and operation permission of different stakeholders, including administrators, in AI system to protect the integrity of AI related logs collected from the AI system.

6.2.4 Training procedure recovery

An AI computing platform should support a recovery function of model training procedure for stakeholders who execute model training tasks. This function may be activated when an AI computing platform meets some system error accidently or suffers from some cyberattack by malicious adversaries. These cyberattacks against AI computing platform or training procedure include:

- Denial of service against AI computing platform resources like CPU, memory, GPU/NPU and NIC to starve the training procedure and consequently unexpectedly cease the training.
- Fault injection attack against computing resources like GPU/NPU to disrupt the training procedure.

This function will recover the training procedure, the corresponding intermediate data, training results, system parameters and etc. to the time point of system abnormality. It effectively mitigates the risks of loss of important data in the training procedure when AI computing platform turns into malfunction. This function could be useful in the scenarios where the training procedure takes longer time.

EXAMPLE: A big model is being trained where a huge amount of parameters are adjusted. Without this function, the interrupted training procedure will be restarted again from the original point, which wastes a lot of computing resources and delays the delivery time of a well-trained model.

The function should satisfy the following requirements to fulfil the recovery task:

- The function should be able to detect the system abnormality via different aspects of the AI computing platform including the chip level, the server node level and the network level. In this way, an AI computing platform can timely react to the system error or cyberattack and shorten the RPO.
- The function should be able to resume the training procedure as soon as the AI computing platform has been recovered from the abnormal state. RTO is the main index to reflect the speed of this function to fulfil the recovery.

NOTE 1: Recovery Point Objective (RPO) is the maximum acceptable time duration from the last recovery point and the maximum targeted period during which transactional data is lost due to a major crash.

NOTE 2: Recovery Time Objective (RTO) is the targeted duration of time within which a procedure is to be restored after a disruption in order to avoid unacceptable consequences associated with a break in business continuity.

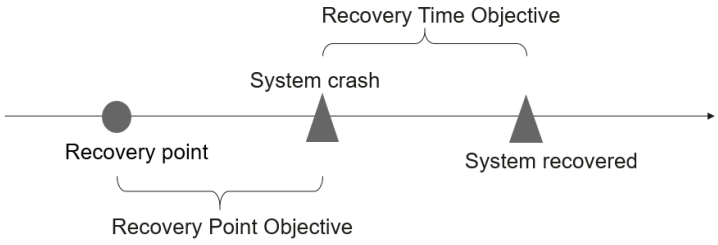

Figure 4: Example diagram of RPO and RTO

6.2.5 Inference attack detection function

As described in ETSI GR SAI 004 [i.1] and ETSI GR SAI 005 [i.2], a model can suffer from adversarial example attacks and model stealing/extraction attacks during the deployment stage. Upon providing resources and an execution environment, an AI computing platform could provide an inference attack detection function for models deployed on it. It should be noted that model-agnostic methods for attack detection are preferred for AI computing platform because the platform needs to support various kinds of models and it may not be the responsibility of the platform to access the model internal to build some detection capabilities.

One possible way of implementing this function is to deploy another model on the platform aside simultaneously as inference attack detection model to recognize suspicious inference queries, i.e. utilizing AI to protect AI. In specific, an inference query sent to the target inference model will also be sent to the detection model, and another inference will be executed based on the detection model. Different from the former inference, the latter inference aims to detect whether sample in this inference query are benign or adversarial, or determine whether the query is normal or malware. As different models provide different detection capabilities due to their different structure and parameters, inference attack detection function in AI computing platform could deploy and utilize different detection models in different scenarios according to different threats and impact on the target model.

EXAMPLE 1: The deployed detection model for adversarial example attack detection is trained with the dataset of various adversarial examples. [i.4] introduced a model to detect adversarial attack on audiovisual speech recognition. The detection model described in [i.5] utilized AutoEncoder (AE) networks to detect adversarial inputs from normal ones.

EXAMPLE 2: The deployed detection model for stealing/extracting attack detection is trained with the dataset of different patterns of query samples to steal/extract information from the target model. [i.6] introduced novel approaches for generating synthetic queries which can be used in model extraction detection training and also proposed a detection method against model extraction attack. [i.7] proposed a new class of estimation model that infers the extraction intent of adversaries.

6.3 Reference architecture of an AI computing platform

In this clause, the reference architecture of an AI computing platform in terms of security components is illustrated in Figure 5. Security components are divided into three layers which are consistent with the structure of an AI computing platform presented in Figure 2. The security components shown in this reference architecture are utilized to fulfil one or more security objectives described in clause 5.4. The detailed description of each security component is presented in clause 7.

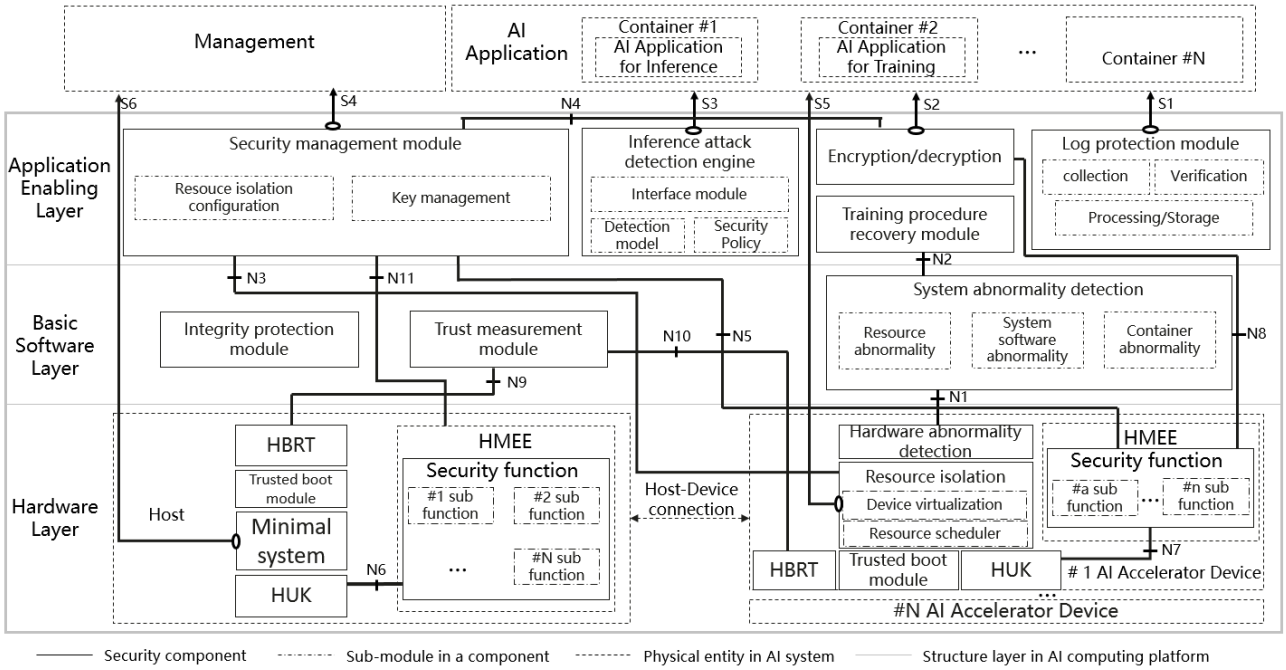

Figure 5: Reference architecture of security components in AI computing platform

It should be noted that one security component may be utilized to fulfil more than one security objectives and a combination of two or more security components may be utilized to fulfil just one security objective which is further described in clause 8.

NOTE: Security components illustrated in Figure 5 are logical components in an AI computing platform. A security component categorized in a certain layer is regarded to be physically implemented by one component or a combination of two or more components in this layer.

EXAMPLE: Integrity protection module could be implemented in OS, chip-level SDK or by an exclusive software module in basic software layer. However, the specific implementation is not limited.

7 Security Components

7.1 Overview

In clause 7, each security component in each layer is described in terms of functionality and interaction with other security components in an AI computing platform.

7.2 Security components in hardware layer

7.2.1 Security related hardware element description

7.2.1.1 Introduction

Security related hardware elements refer to some components or techniques where all functions, and data, are trusted to operate and be operated on and protected by hardware so as to minimize any leakage of security knowledge by observation, either direct or indirect. These components or techniques includes SE, HSM, TPM, TEE, etc. In an AI computing platform, these security related hardware elements can be used to form security components in hardware layer described in the following clauses.

7.2.1.2 TEE

A Trusted Execution Environment (TEE) is a proprietary manifestation of HMEE [i.8] and usually acts as a protected area of process space and memory within a system which delivers confidentiality and integrity of instructions and data associated with that space. It is protected from eavesdropping, replay and alteration attacks as the programs/data within the space are executed [i.3].

7.2.1.3 TPM

A Trusted Platform Module (TPM) is a hardware cryptographic module that can securely store sensitive data and perform various cryptographic operations. Authentication (a process to prove the identity attribute of an entity, i.e. the TPM acting as the integrity reporting entity) and attestation (a process that enables the software integrity state to be reported and verified in order to determine its trustworthiness) are necessary steps to ensure trusted computing. A TPM can authenticate itself using the credentials stored in the shielded memory and provide integrity measurements reports to prove platform software is trustworthy [i.8].

7.2.1.4 SE

A Secure Element (SE) is a hardware tamper-resistant platform (typically a one chip secure microcontroller) capable of securely hosting applications keeping the data integrity and their confidential and cryptographic data (for example cryptographic keys) in accordance with the rules and security requirements set by well identified trusted authorities. There are different form factors of SE: smartcards, embedded and integrated SEs, SIM/UICC, smart microSD, etc. SEs exist in different form factors to address the requirements of different business implementations and market needs.

7.2.1.5 HSM

A Hardware Security Module (HSM) is a dedicated system that physically and logically secure cryptographic keys and cryptographic processing and are available in two forms:

- Standalone network-attached appliances.
- Hardware cards that plug into existing network-attached systems.

HSMs are fully contained solutions for cryptographic processing, key generation, and key storage. As purpose-built appliances, they automatically include the hardware and firmware (i.e. software) necessary for these functions in an integrated package. Physical and logical protection of the appliance is supported by a tamper resistant/evident shell; and protection from logical threats, depending on the vendor's products, is supported by integrated firewall and intrusion prevention defences.

Functions supported by HSMs include [i.8]:

- Life-cycle management of cryptographic keys used to lock and unlock access to digitized information: generation, distribution, rotation, storage, termination, and archival.
- Cryptographic processing which produces the dual benefits of isolating and offloading cryptographic processing from application servers.

7.2.2 Host HMEE security function module

In AI context, Host HMEE security function module refers to a security component that needs to be deployed and operate in host side HMEE like host TEE, SE or HSM. Host TEE refers to the trusted execution environment implemented in the host side of an AI computing platform rather than TEE on AI-specific accelerator. The functionality and the cooperation with other security components are described as follows:

- Functionality: host HMEE security function module provides a subset of security crucial and sensitive functions from AI application or system management. It contains some parts of application codes from AI application and system software to execute some secure and sensitive operations to support more secure operation. The module needs to be deployed in and protected by host HMEE. Only authorized components within an AI system are allowed to communicate to the module. Furthermore, communications between authorized components and this module are usually restricted, i.e. output data from HMEE to REE is strictly controlled.
- Coordination: None.

7.2.3 AI accelerator HMEE security function module

In contrast to host HMEE security function module, an AI accelerator HMEE security function module refers to a security component that is deployed and operate in the HMEE of the AI accelerator side, including AI accelerator TEE, SE or HSM. In specific, an AI accelerator acts as a hardware-based protected execution environment where AI accelerator resources and AI accelerator memory space are isolated from outside. The functionality and the cooperation with other security components are described as follows:

- Functionality: Working in the same way with host HMEE security function, an AI accelerator HMEE security function module provides a subset of security crucial and sensitive functions from AI application or system management. Different from host HMEE, GPU/NPU-based unique information could be protected and securely processed in HMEE of AI accelerator side. As a result, the function provided by this security module can support some security services/functions in AI computing platform to bind designated device to fulfil some security objective. Only the encryption/decryption module from the same host can communicate with this module. For remote security management module on other hosts, this module should also communicate with them utilizing secure communication protocol like TLS.
- Coordination: an AI accelerator HMEE could communicate with security management module through N5, with AI accelerator HUK through N7 and with encryption/decryption module through N8, to support the fulfilment of security functionalities of these security components and security services of AI computing platform, especially for the security services that needs to bind designated device.

7.2.4 Hardware abnormality detection module

An hardware abnormality detection module monitors on hardware states and operation condition. The functionality and the cooperation with other security components are described as follows:

- Functionality: an hardware abnormality detection module acts as a monitor to estimate whether underlying hardware like CPU, memory space, GPU/NPU and NIC works as expected. When malfunction or errors from hardware happens, this module needs to timely detect these abnormalities and alert relevant security components via a certain interface for quick response. It also needs to provide interfaces for upper layer OS and other applications to retrieve the real-time state and operation statistics.

- Coordination: an hardware abnormality detection module could communicate with system abnormality detection module through N1.

7.2.5 AI accelerator resource isolation module

An AI accelerator resource isolation module provides isolation of resource of an AI accelerator like computing unit and memory space for different users. The functionality and the cooperation with other security components are described as follows:

- Functionality: an AI accelerator resource isolation module acts as a manager to isolate resource of an AI accelerator according to configuration. Different users can only utilize their authorized computing unit and memory space so that if an error happens in one isolated space or a malicious user, it will not affect other users' resource. Also, this component needs to manage that isolated resource utilized by each user will not exceed the pre-configured quota. In addition, an interface needs to be provided by this component for administrators to configure relevant parameters like resource quota for a certain user.

- Coordination: an AI accelerator resource isolation module could communicate with security management module through N3.

7.2.6 Host trusted boot module

Trusted boot module is regarded as inherently trusted and secure by design [i.8]. Trusted boot module could act as a tamper-resistant trust anchor in the host side of an AI computing platform. The functionality and the cooperation with other security components are described as follows:

- Functionality: a host trusted boot module usually contains initial boot code for AI computing platform upon which other components in the platform will be successively verified and started. Host trusted boot module needs to guarantee that initial boot code will not be tampered. It is usually implemented in CPU SoC.

- Coordination: None.

7.2.7 AI accelerator trusted boot module

An AI accelerator trusted boot module could also act as a tamper-resistant trust anchor in each AI accelerator of an AI computing platform. The functionality and the cooperation with other security components are described as follows:

- Functionality: an AI accelerator trusted boot module usually contains initial boot code for the starting of the accelerator so that each GPU/NPU could be started as expected with its own trust anchor. Similarly to the host trusted boot module, this component needs to guarantee that the initial boot code will not be tampered. It is usually implemented in GPU/NPU SoC.

- Coordination: None

7.2.8 Host HBRT

HBRT acts as initial root of trust defined in ETSI GS NFV-SEC 012 [i.9]. It is used as a specific security hardware component to protect the system against unauthorized changes and attacks such as malware and root kits [i.8]. HBRT could be based on hardware-based TPM or equivalent hardware root of trust such as HSM or SE (SIM, embedded SE, integrated SE).

EXAMPLE 1: When used in IOT device, embedded SEs are providing secure services such as RoT for measurement, attestation, encryption, signature, integrity and confidentiality services used for the device security.

With all these possible forms, an host HBRT acts as RTS and RTR for integrity measurement on the host side. The functionality and the cooperation with other security components are described as follows:

- Functionality: a host HBRT could record the integrity measurement from the host side of an AI computing platform into security storage and report these measurement values through interfaces to upper layer components as needed. These values in security storage need to be protected from deletion and tampering. It is usually implemented on the motherboard of an AI computing platform.

EXAMPLE 2: The integrity measurement of BIOS, shim of host OS and kernel are recorded and protected in a host HBRT.

- Coordination: a host HBRT could communicate with a trust measurement module through N9.

7.2.9 AI accelerator HBRT

Like a host HBRT, an AI accelerator HBRT acts as RTS and RTR for integrity measurement exclusively for an AI accelerator. The functionality and the cooperation with other security components are described as follows:

- Functionality: an AI accelerator HBRT could record the integrity measurement from the AI accelerator in an AI computing platform and report these measurement values through interfaces to upper layer components as needed. These values in security storage need to be protected from deletion and tampering. It is usually embedded on AI accelerator SoC.

EXAMPLE: The integrity measurement of an AI accelerator is recorded and protected in an AI accelerator HBRT.

- Coordination: an AI accelerator HBRT could communicate with a trust measurement module through N10.

7.2.10 Minimal system

A minimal system usually acts as a safe box for an AI computing platform to execute a set of minimally required tasks when it has been attacked or disrupted. The functionality and the cooperation with other security components are described as follows:

- Functionality: a minimal system is usually responsible to implement a safe mode for an AI computing platform when the platform has been attacked or it out of work. It contains the minimally required codes for system recovery in any conditions except for physical destruction. Upon this security component, an AI computing platform can be recovered to an expected state.

- Coordination: None.

7.2.11 Host HUK

A host HUK could act as the unique identity credential of an AI computing platform. Normally, each platform has its own proprietary credential. The functionality and the cooperation with other security components are described as follows:

- Functionality: a HUK could act as the unique credential for an AI computing platform. It usually contains a random value which is used to generate some other keys or a predefined key that is utilized to protect other keys. HUK should not be manipulated after first setting at factory and it can only be accessed by the authorized components like security functions deployed in HMEE of host.

- Coordination: a HUK module could communicate with host HMEE security function module through N6.

7.2.12 AI accelerator HUK

Similarly to a host HUK, an AI accelerator HUK could act as the unique identity credential of a certain AI accelerator. Normally, each accelerator device has its own proprietary credential. The functionality and the cooperation with other security components are described as follows:

- Functionality: an AI accelerator HUK is the unique credential for an AI accelerator device. Like a host HUK, it also usually contains a random value which is used to generate some other keys or a predefined key that is utilized to protect other keys. The key should not be manipulated after first setting at factory and it can only be accessed by the authorized components like security functions deployed in the HMEE of the device.

- Coordination: an AI accelerator HUK module could communicate with AI accelerator HMEE security function module through N7.

7.3 Security components in basic software layer

7.3.1 System abnormality detection module

A system abnormality detection module is usually deployed upon the host OS of an AI computing platform and monitors the system state and operation condition of an AI computing platform. The functionality and the cooperation with other security components are described as follows:

- Functionality: a system abnormality detection module could act as a monitor to estimate whether an AI computing platform works as expected and to identify whether an AI computing platform is being or has been attacked. On one hand, this module collects hardware monitoring information from hardware abnormality detection module; on the other hand, this module needs to monitor OS and other software in basic software layer, and provides abnormality detection function to identify whether the system has been or is being attacked. When malfunction or errors in the AI computing platform happens, this module could timely detect these abnormalities and alert relevant security components via certain interfaces for a quick response. To cooperate with other system components, it often provides interfaces for AI applications or relevant security components to retrieve the real-time state and operation statistics of an AI computing platform.

EXAMPLE: Network interface and traffic, OS operation and container operation are in the scope of system abnormality detection.

- Coordination: a system abnormality detection module could communicate with a hardware abnormality detection module through N1 and a training procedure recovery module through N2.

7.3.2 Trust measurement module

A trust measurement module usually acts as a software component to enable the capabilities of a host HBRT and an AI accelerator HBRT. The functionality and the cooperation with other security components are described as follows:

- Functionality: The main task of a trust measurement module is to utilize the security capability of an underlying HBRT. In specific, this module usually provides an interface for upper applications and OS in an AI computing platform to send measurement values to a HBRT for security storage and fulfils remote attestation procedure with a HBRT.

- Coordination: a trust measurement module could communicate with a host HBRT through N9 and an AI accelerator HBRT through N10.

7.3.3 Integrity protection module

An integrity protection module could utilize cryptographic methods and underlying hardware protection to implement integrity protection for other components and important AI assets. The functionality and the cooperation with other security components are described as follows:

- Functionality: an integrity protection module could provide integrity protection for AI applications and other security components in basic software layer and upper AI application enabling layer. In more details, this module could utilize cryptographic algorithms like hash and digital signature to guarantee the integrity and authority of program codes and AI assets. It can also work in coordination with security components in hardware layer as needed to utilize hardware-based integrity protection.
- Coordination: None.

7.4 Security components in application enabling layer

7.4.1 Security management module

A security management module provides management and configuration of related security functions and services in an AI computing platform. It usually provides interfaces for administrators to manage security functions and services in terms of security parameter configuration and sensitive information management like cryptographic keys. The functionality and the cooperation with other security components are described as follows:

- Functionality: a security management module could act as an interface for platform users to manage the security functions and services of an AI computing platform. In specific, this module usually provides interfaces to create and delete the isolation instance of an AI accelerator, and to configure the parameters of the isolation instance. On the other hand, this module could provide key management for the encryption/decryption service. It usually provides interfaces for different tenants to manage their own keys like creation, deletion or authorization to other users. The keys might be used in cryptographic procedure in multi-tenant environment. It can communicate with security functions in an HMEE for more secure key protection and utilization if needed. This module could also communicate with the encryption/decryption module and an AI accelerator HMEE security function module on a remote AI computing platform utilizing secure communication protocol like TLS.
- Coordination: a security management module could communicate with a hardware isolation module through N3, a encryption/decryption module through N4, an AI accelerator HMEE security function through N5.

7.4.2 Inference attack detection engine

An inference attack detection engine can provide a model-agnostic detection service for AI application. An AI application could invoke this security component to protect models deployed on AI computing platform in inference stage. The functionality and the cooperation with other security components are described as follows:

- Functionality: an inference attack detection engine analyses inference queries and related sample sent to inference models and judges whether queries are malicious or samples are adversarial. In detail, it usually provides an interface for AI applications to send inference queries to the engine and receive analysis results upon the queries. This engine includes a detection model for queries analysis and inference sample analysis. The model deployed in this engine can be changed for different detection tasks according to different threats in application scenarios. Also, an AI application could utilize this module either in a serial fashion or parallel fashion as needed. The reference implementation examples for these two fashions are presented in clause 9.6.5.
- Coordination: None.

7.4.3 Encryption/decryption module

An encryption/decryption module can implement a cryptographic process for AI assets. The confidentiality of AI assets could be protected upon utilizing this module. The functionality and the cooperation with other security components are described as follows:

- Functionality: an encryption/decryption module usually utilizes predefined keys and certain cryptographic algorithms to encrypt AI assets so that unauthorized visitors cannot obtain the plain content with value data and information at transmission or storage phase. On the contrary, this module decrypts AI assets with the appropriate key when authorized users need to use them. In an AI computing platform, this module communicates with a security management module to acquire available cryptographic keys for encryption and decryption. For platform users, this module usually provides interfaces for users to send AI assets into this module for encryption and retrieve AI assets from the module. It can be designed to automatically implement cryptographic procedures upon the cooperation with a security management module according to the user identity. In this scenario, the platform users need not to consider the relevant keys during the cryptographic computation and only need to store assets and load assets as common practice, which facilitates the integration of a encryption/decryption mechanism into an existing AI application logic.

- Coordination: an encryption/decryption module could communicate with a security management module through N4.

7.4.4 Training procedure recovery module

A training procedure recovery module is responsible for quick recovery of model training procedure. The aim of this module is to resume the training at the very time point when the system is attacked or goes wrong accidently. The functionality and the cooperation with other security components are described as follows:

- Functionality: a training procedure recovery module usually identifies a training procedure abnormality, and timely backs up execution context and important training data. The module identifies an abnormality through two ways. First, it can monitor the training process and identify whether the training software functions as expected. On the other hand, this module could receive the abnormality alerts from underlying abnormality detection modules via interfaces. Last but not least, this module could restart the training procedure with backup data automatically if the abnormality resolved or sends alerts to platform users for human intervention.

NOTE: The automatic recovery can be implemented in different ways. Redundant GPU/NPU device scheduling and reallocation can be a practical countermeasure.

- Coordination: a trust measurement module could communicate with a system abnormality detection module through N2.

7.4.5 Log protection module

A log protection module could provide a log protection service for AI applications running on an AI computing platform. It usually collects AI-relevant logs from AI applications and protects the integrity, authority and original sequence of these logs. The functionality and the cooperation with other security components are described as follows:

- Functionality: a log protection module can utilize cryptographic methods and underlying hardware to protect collected logs generated from AI applications and AI computing platform users. It usually provides interfaces for AI applications and relevant platform users to send logs into this module. Then this module could utilize cryptographic methods like hash algorithm and Merkle tree structure to record the evidence of correct logs. This evidence can be further protected by underlying hardware security components. Also, this module usually provides another interface for the verification of logs in terms of correct integrity, authority and sequence. When a bunch of logs needs to be verified in scenarios like audit or forensic, this module could verify the correctness of these logs and provide the conclusion to users via an interface.

- Coordination: None.

8 Reference points and service-based interface

8.1 Reference point between security components

N1: The N1 reference point between [Hardware abnormality detection module] and [System abnormality detection module] could be used to transmit hardware real-time monitoring data and abnormality alerts.

N2: The N2 reference point between [System abnormality detection module] and [Training procedure recovery module] could be used to transmit system real-time monitoring data of an AI computing platform and abnormality alerts both from the system software and hardware.

N3: The N3 reference point between [Security management module] and [Resource isolation module] could be used to transmit AI-specific resource isolation management and configuration data of a certain AI-specific accelerator.

N4: The N4 reference point between [Security management module] and [Encryption/decryption module] could be used to transmit cryptographic keys used in the model and data encryption and decryption.

N5: The N5 reference point between [Security management module] and [AI accelerator security function] could be used in key management scenarios to transmit cryptographic keys used in the model and data encryption and decryption related to device binding application scenarios.

N6: The N6 reference point between [Host HUK] and [Host HMEE security function] could be used to transmit a host HUK to a security function module in a host HMEE.

N7: The N7 reference point between [AI accelerator HUK] and [AI accelerator HMEE security function] could be used to transmit an AI accelerator HUK to a security function module in a host HMEE.

N8: The N8 reference point between [Encryption/decryption module] and [AI accelerator HMEE security function] could be used in AI assets encryption/decryption scenarios to transmit cryptographic keys used in the model and data encryption and decryption related to device binding application scenarios.

N9: The N9 reference point between [Host HBRT] and [Trust measurement module] could be used to transmit integrity related data of firmware and software on the host side of as AI computing platform in integrity verification scenarios.

N10: The N10 reference point between [AI accelerator HBRT] and [Trust measurement module] could be used to transmit integrity related data of firmware and software on the AI accelerator side of as AI computing platform in integrity verification scenarios.

8.2 Service-based interface for platform users

S1: S1 provides two interfaces for platform users to invoke the AI log protection service. The reference interface description is listed below:

- Log protection interface:
- Input: logs.
  - Output: not defined, reception confirmation or none reply is possible implementation.
- Log verification interface:
- Input: verification request, including information of logs to be verified.
  - Output: log files and the verification result indicating whether being verified logs is correct in content and sequence.

S2: S2 provides an interface for platform users to invoke the encryption/decryption function. The reference interface description is listed below:

- Encryption interface:
- Input: AI assets to be encrypted.
  - Output: None.
- Decryption interface:
- Input: AI asset load query. The request contains AI asset identity information and user identity of query. Example of identity information of AI assets are AI asset file name, unique number or storage path in filesystem.
  - Output: AI asset with plaintext.

S3: S3 provides an interface for platform users to invoke the inference attack detection function. The reference interface description is listed below:

- Query detection request interface:
- Input: inference query that includes inference sample, user identity of query, query related information like time, network address, etc.
  - Output: detection results that clarify whether this query is malicious or benign. For adversarial attack, the detection results may also infer the location of adversarial parts and degree of confidence for these parts.

S4: S4 provides an interface for platform users to implement security management for the relevant security function and components. The reference interface description is listed below:

- AI resource isolation management interface:
- Input: operations for AI resource including creating an isolation instance with defined resource quote, revising an existing isolation to a targeted resource quote and deleting an existing isolation.
  - Output: Not mandatory.
- Encryption/decryption key management interface:
- Input: operations for key management including creating a key bound to a specific user identity, deleting a certain key, allocating a certain key to a specific device, etc.
  - Output: Not mandatory.

S5: S5 provides an interface for platform users for requesting their isolated AI-specific computing resource to execute AI model training or inference tasks. The reference interface description is listed below:

- AI resource request interface.
- Input: request for isolated resource utilization including user related identification information and resource requirement, and the data to be processed on AI-specific accelerator.
  - Output: computation results.

S6: S6 provides an interface for platform users to recover the platform to a predefined operation state when the platform gets attacked or malfunctions. The reference interface description is listed below:

- Recovery interface.
- Input: request for system recovery.
  - Output: Not mandatory.

9 Mechanism of Security Functions/Services

9.1 Overview

In clause 9, the detailed description of each security function or security service is analysed and illustrated in the form of a reference example. The example includes the description of involved security components and corresponding coordination, interaction information, sequence diagrams, interactive interface between users and AI computing platform, etc.

9.2 AI assets encryption/decryption

9.2.1 Description of a reference example for the mechanism

As the reference example to implement the mechanism, an AI assets encryption/decryption mechanism could be executed by the cooperation of [AI accelerator HMEE security function module], [Encryption/decryption module] and [Security management module]. Upon this mechanism, platform users could utilize the AI assets encryption/decryption service described in clause 6.2.2.1. This service provides confidentiality protection for AI assets from stealing and eavesdropping at storage and transfer phase, especially important in multi-tenant scenarios. The mechanism could provide interfaces for AI application developers to utilize this mechanism for key assets protection. As a reference example, <S2> could be provided by [Security management module] to invoke encryption/decryption function in their application logic. <S4> could be provided by [Security management module] to manage the cryptographic keys related to encryption and decryption.

9.2.2 Involved security components

This reference example mechanism involves:

- [AI accelerator HMEE security function module] described in clause 7.2.3.
- [AI accelerator HUK] described in clause 7.2.12.
- [Security management module] described in clause 7.4.1.
- [Encryption/decryption module] described in clause 7.4.3.

9.2.3 Reference point and service-based interface

This reference example mechanism involves:

- Reference point: <N4>
- Reference point: <N5>
- Reference point: <N7>
- Reference point: <N8>
- Service-based interface: <S2>
- Service-based interface: <S4>

9.2.4 Mechanism procedure

In this reference implementation example, there are three related procedures included in AI assets encryption/decryption, i.e. key management, AI assets encryption and decryption.

For key management, the procedure can be further divided into two categories, i.e. common key management procedure and device bound key management procedure. Device bound key management is only available when device bound encryption/decryption is utilized where AI assets can only be decrypted on an allocated device (NPU/GPU) as described in clause 6.2.2.1. The detailed procedure is shown in Figure 6 as an example to illustrate the interaction between different security components:

- Step 1: The AI application or user management software sends a key management request to [Security management module] via <S4>. The request contains the key management operation information, tenant information and device information if needed.
- Step 2: [Security management module] implements key management operation as indicated in the request received in step 1. The operation includes creating a new key for a designated tenant, revising the existing key information, deleting a key and so on.
- Step 3: If key management is related to device-bound encryption/decryption key, [Security management module] sends management requests via <N5> to [AI accelerator HMEE security function] on allocated device indicated in the request received in step 1. If the management request is to create a device-bound key or revise an existing key, new key will be transmitted within the request. The communication channel between [Security management module] and [AI accelerator HMEE security function] should utilize secure communication protocol as described in clauses 7.2.2 and 7.4.1.
- Step 4: Upon receiving the management request, [AI accelerator HMEE security function] requests the device HUK stored in [AI accelerator HUK] on the same allocated device via <N7>.
- Step 5: [AI accelerator HUK] sends the device HUK to [AI accelerator HMEE security function] via <N7>.
- Step 6: Utilizing HUK, [AI accelerator HMEE security function] implements the key management operation. For more detail, in the operation of creating a new key or revising an existing key, [AI accelerator HMEE security function] utilizes HUK to encrypt the relevant key for storage protection.

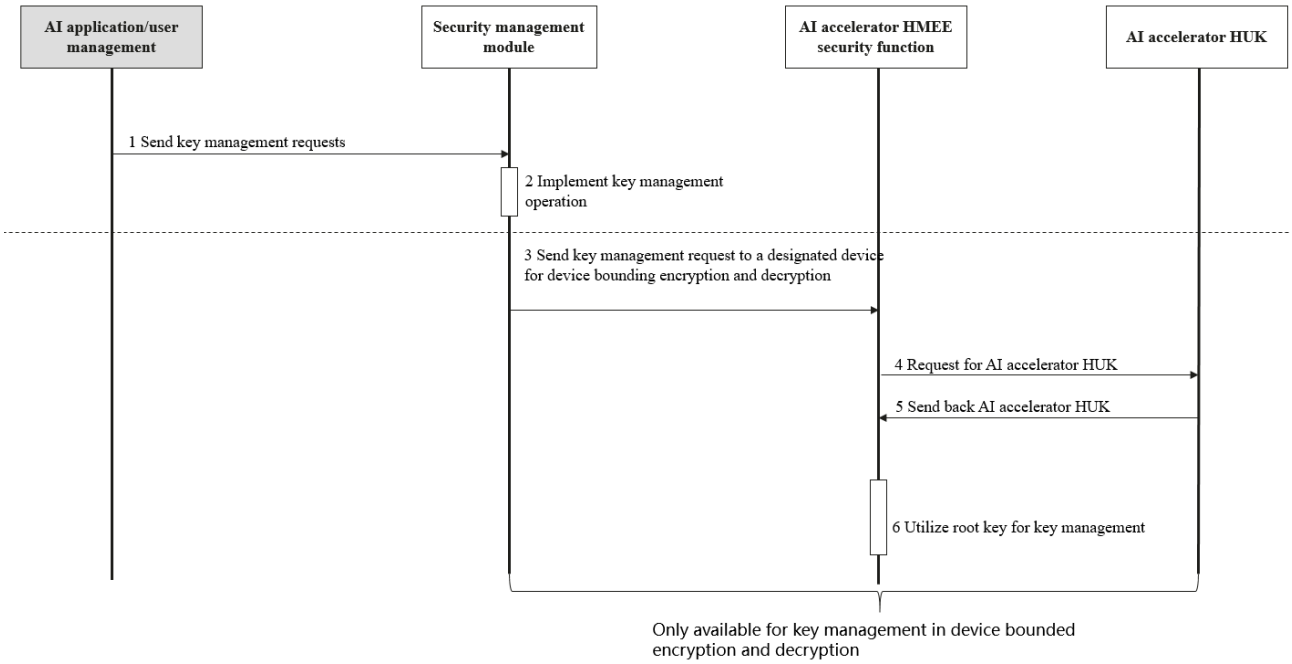

Figure 6: Mechanism sequence diagram for key management

For encryption, the detailed procedure is shown in Figure 7.

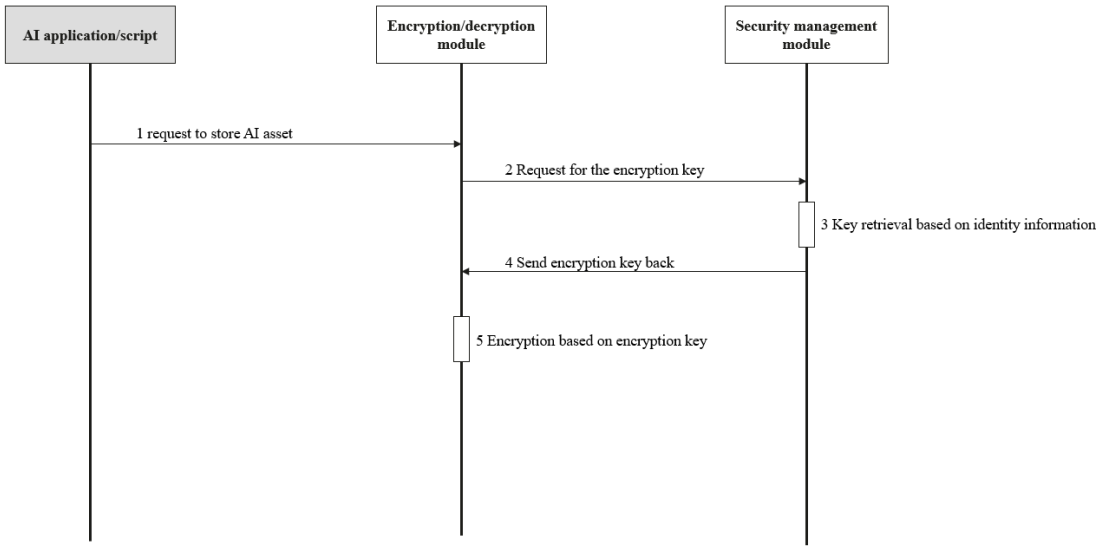

Figure 7: Mechanism sequence diagram for AI asset encryption

- Step 1: The AI application or script sends a storage request to [Encryption/decryption module] for an AI asset encryption via <S2>. The request contains the AI assets to be encrypted and stored, and the tenant identity information who runs the AI application.
- Step 2: Upon receiving the request, [Encryption/decryption module] sends a request to get the encryption key to [Security management module] via <N4>. The request contains the tenant identity information received in step 1.
- Step 3: [Security management module] searches for the required encryption key according to the tenant identity information.
- Step 4: [Security management module] sends the required encryption key to [Encryption/decryption module] via <N4>.
- Step 5: [Encryption/decryption module] implement encryption with the encryption key obtained in step 4.

For decryption, the procedure can also be further divided into two categories, i.e. common decryption procedure and device bound decryption procedure. Device bound decryption is only available when it has been configured by platform users for the sake of strict protection of AI assets. The detailed example interaction between security components is shown in Figure 8.

For common decryption procedure:

- Step 1: The AI application or script sends an AI asset storage request to [Encryption/decryption module] for AI asset decryption via <S2>]. The request contains the identity information of AI assets and of the tenant who runs the AI application.
- Step 2: Upon receiving the request, [Encryption/decryption module] sends a request to get the decryption key to [Security management module] via <N4>. The request contains the tenant identity information received in step 1.
- Step 3: [Security management module] searches for the required decryption key according to the tenant identity information.
- Step 4: [Security management module] sends the required decryption key to [Encryption/decryption module] via <N4>.
- Step 5: [Encryption/decryption module] implements decryption with the decryption key obtained in step 4.

- Step 6: [Encryption/decryption module] sends the AI asset which has been decrypted back to the AI application or script <S2>.

For device bound decryption scenarios, as the decryption key has been transferred to the allocated AI accelerator, the detailed example procedure is different from common scenarios as shown in the lower part of Figure 8.

- Step 1: The AI application or script sends an AI asset storage request to [Encryption/decryption module] for AI asset decryption via <S2>]. The request contains the identity information of AI assets and of the tenant who runs the AI application.
- Step 2: Upon receiving the request, [Encryption/decryption module] sends a request to get the decryption key to [AI accelerator HMEE function module] via <N8>. The request contains the tenant identity information received in step 1.
- Step 3: [AI accelerator HMEE function module] searches for the required decryption key according to the tenant identity information.
- Step 4: Upon finding the relevant key in the ciphertext protected by the device HUK, [AI accelerator HMEE security function] requests the device HUK stored in [AI accelerator HUK] on the same allocated device via <N7>.
- Step 5: [AI accelerator HUK] sends the device HUK to [AI accelerator HMEE security function] via <N7>.
- Step 6: [AI accelerator HMEE security function] implements decryption with HUK obtained in step 5 and obtains the decryption key for the AI asset in plaintext.
- Step 7: [AI accelerator HMEE security function] sends decryption back to [Encryption/decryption module] via <N8>.
- Step 8: [Encryption/decryption module] implements decryption with the decryption key obtained in step 7.
- Step 9: [Encryption/decryption module] sends the AI asset which has been decrypted back to the AI application or script <S2>.

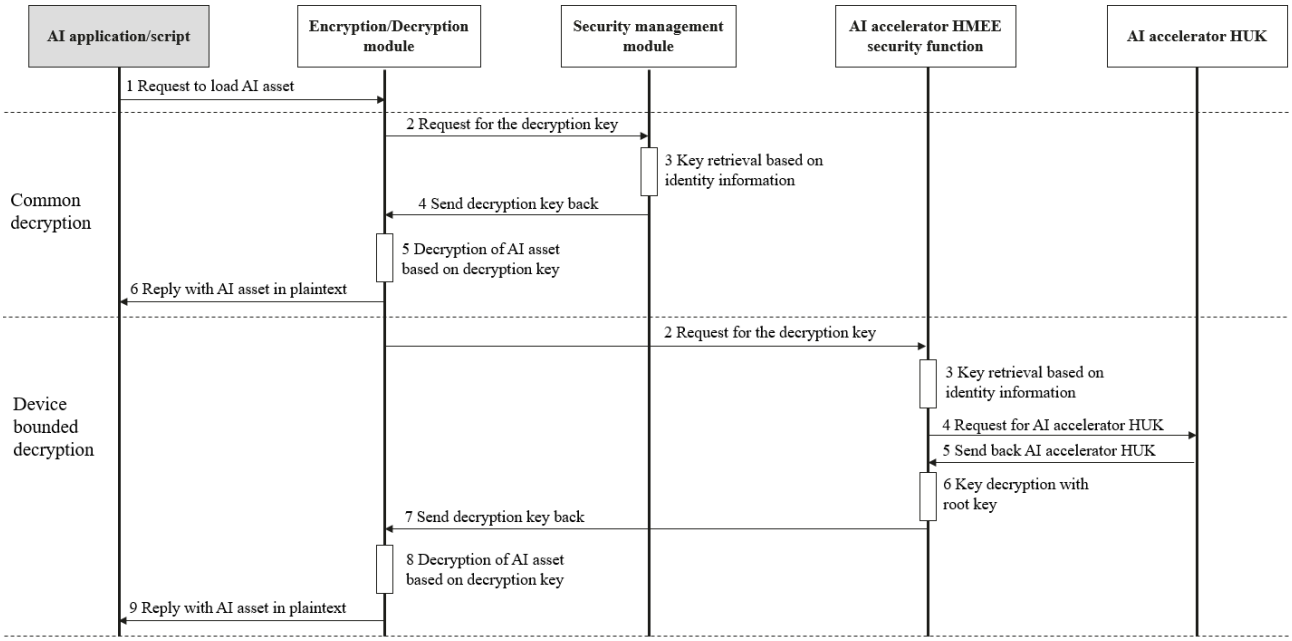

Figure 8: Mechanism sequence diagram for AI asset decryption

NOTE: Device bound decryption is only available after the key has been bound and transferred to the designated device through the procedure of device bound key management.

9.2.5 Reference deployment

The reference deployment of the AI assets encryption/decryption service is illustrated in Figures 9 and 10 as an implementation example to better understand the mechanism and its utilization. Figure 9 focuses on datacenter scenarios while Figure 10 mainly considers edge computing scenarios.

For datacenter scenarios, [Security management module] is usually deployed on a management node. By invoking <S4>, users can integrate the key management function in their own management software or portal. In the management node, [Security management module] can further cooperate with some other hardware security components for more secure protection on the keys it manages. In the work node, for automatic and non-invasive encryption/decryption manner for AI application, [Encryption/decryption module] can be integrated into AI containers that run AI application. When an AI application loads the AI model for inference from the filesystem or stores a well-trained model to filesystem after training via <S2>, [Encryption/decryption module] can automatically implement encryption or decryption based on the cooperation with [security management module]. The communication between the two modules can be protected by utilizing TLS and other communication security technology.

As seen in Figure 9, different tenants will utilize different keys in cryptographic procedures for their own AI assets protection. For security purpose, tenant bound keys stays and are protected in a security management module, while working keys for certain AI assets encryption and decryption are allowed to be transmitted between [Security management module] and [Encryption/decryption module]. Working keys are protected by tenant bound keys while tenant bound keys are protected by other hardware security components on the AI computing platform like [Host HMEE security function module] and [Host HUK]. The detailed key utilization and protection structure is described below in details.

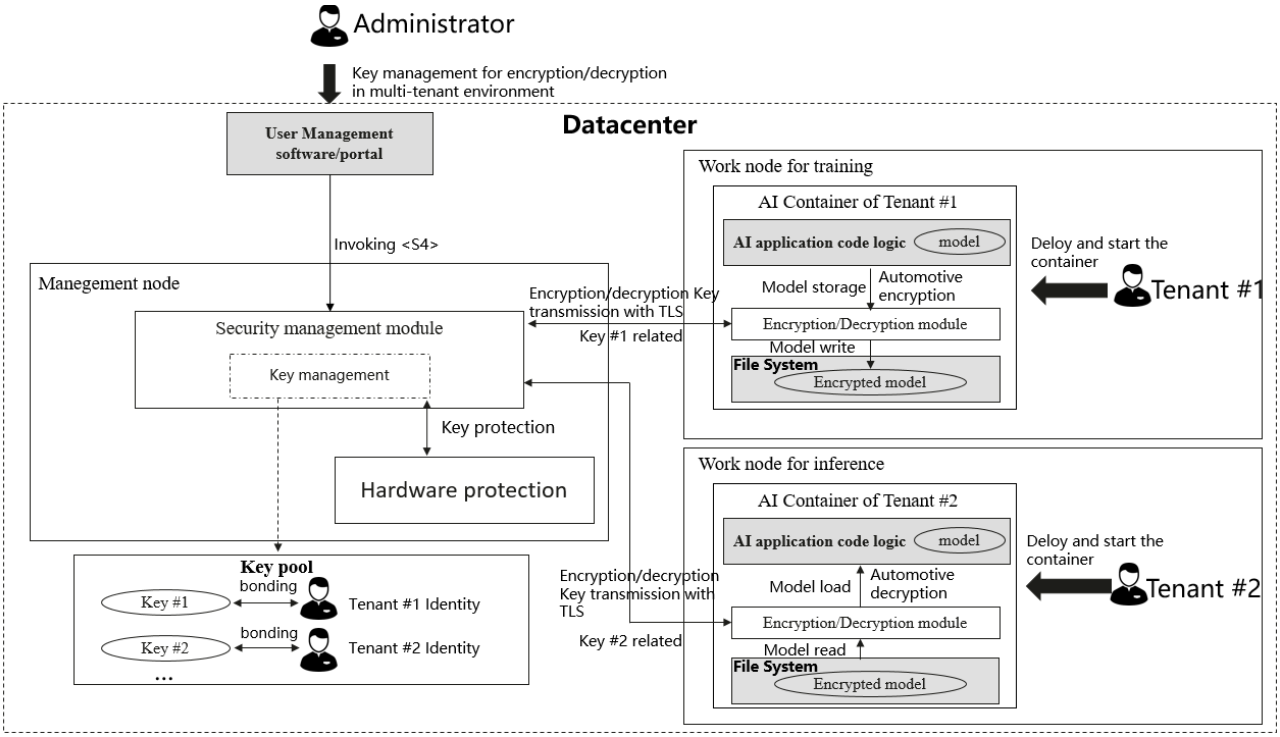

Figure 9: Diagram of reference deployment model for encryption/decryption service in datacenter scenarios

For edge computing scenarios, AI models are usually trained in datacenter and deployed on edge for inference application. In these conditions, device bound decryption is a very useful method to protect the well-trained model deployed on an edge node. As shown in Figure 10, [Security management module] is still deployed on a management node. An administrator can invoke <S4> to manage the keys used for device bound decryption. [Encryption/decryption module] is integrated in an AI inference container and deployed on the edge computing platform. In order to fulfil the device bound decryption, [AI accelerator HMEE security function module] and [AI accelerator HUK] on the bound AI accelerator (#1 accelerator in figure as an example) are utilized to store, protect and provide the decryption key transmitted from [security management module] on the management node. With the help of the [AI accelerator HMEE security function module], the model can only be decrypted on #1 accelerators as the relevant keys are only stored and protected on this device.

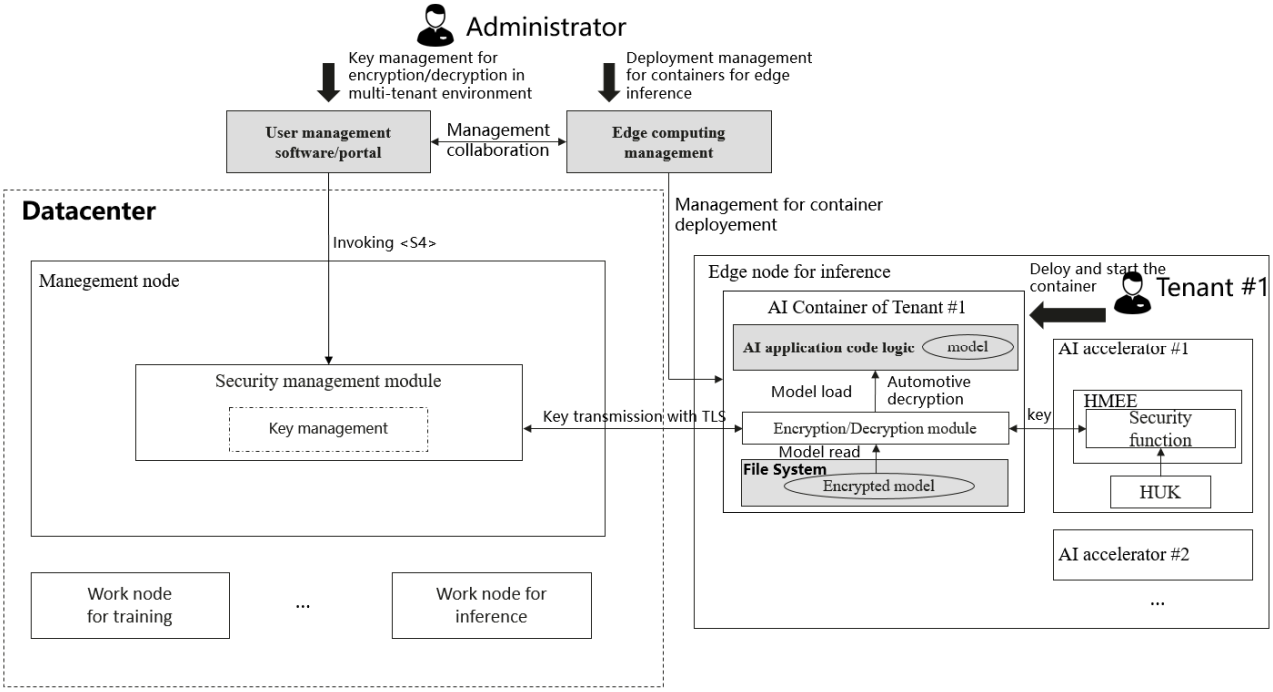

Figure 10: Diagram of reference deployment model for encryption/decryption service in edge scenarios

An example procedure from model training to model inference with the protection of encryption/decryption service on an AI computing platform is illustrated in Figure 11. This example, a user trains the model in a datacenter and deploys the model for inference on an edge computing platform to reduce the delay of the AI service. First, as shown with remark "1" in the figure, the model is trained on a work node within the datacenter and the model is encrypted utilizing the encryption/decryption service provided by the platform; then, the image containing the encrypted model is uploaded to the container registry. Secondly, with remark "2" in the figure, the user binds the relevant key to the AI accelerator #1 on the edge node. Finally, the platform user deploys the inference image onto the edge node and pulls up the container; when an AI application attempts to load a model in the filesystem, encryption/decryption service automatically decrypts the model and delivers to the AI application for inference. It can be found with remark "3" in the figure.

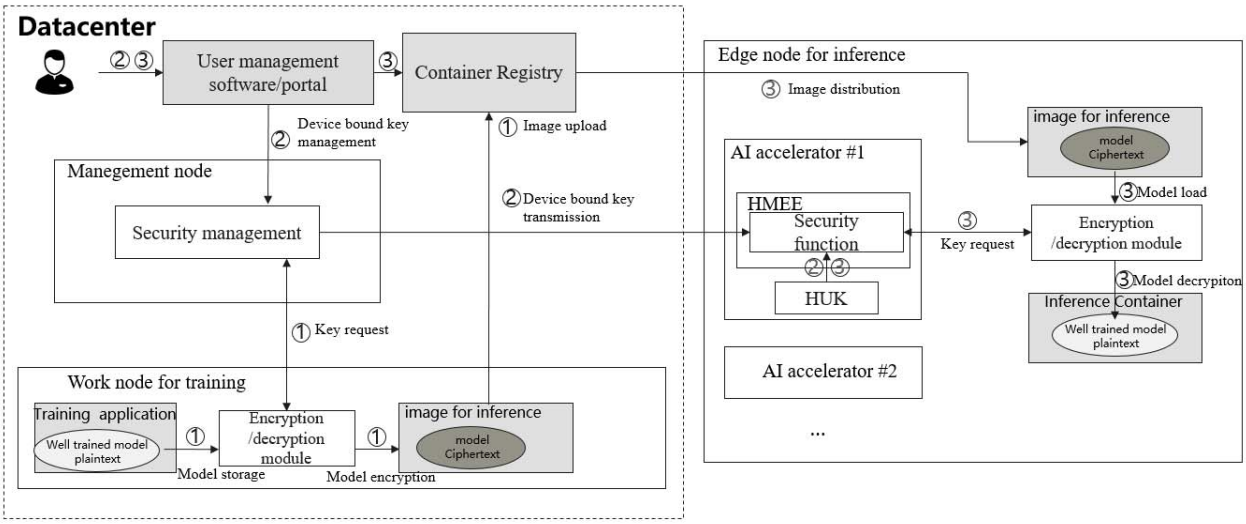

Figure 11: Example diagram of encryption/decryption service utilization

For key utilization and protection, a three-layer key structure is recommended in encryption/decryption scenarios. As shown in Figure 12, layer 1 key is platform-bound HUK which is utilizes to protect layer 2 keys. It should be guaranteed that each platform has only one HUK. In an AI computing platform, the host HUK described in clause 7.2.11 can be utilized as the HUK. Layer 2 key is a tenant-bound key which is utilized to protect layer 3 keys.

Each key in this layer should be bound to a specific tenant and only be authorized to this tenant. A layer 3 key is an AI assets-bound key which is utilized to protect the bound to AI assets. Utilizing a three-layer key structure, each tenant can only encrypt and decrypt their own AI assets as each AI assets is protected by a layer 3 key and a layer 3 key can only be decrypted with the right layer 2 key authorized only to the bound tenant. The layer 1 HUK acts as a trust anchor for the key structure, as HUK is always protected on a hardware-level. In addition, encryption and decryption procedures for layer 2 keys and layer 3 keys can be further implemented in hardware security module like the host HMEE security function module described in clause 7.2.2.

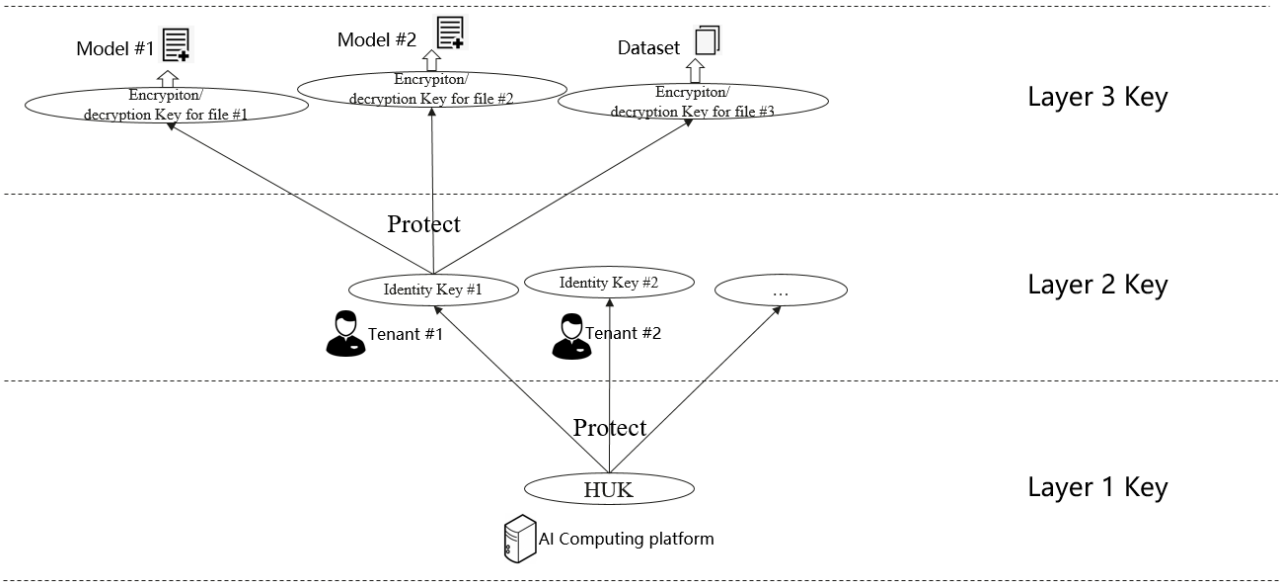

Figure 12: Three tier key structure

9.3 AI-specific computing resource isolation

9.3.1 Mechanism description

As the reference example to implement the mechanism, AI-specific computing resource isolation is executed by [AI accelerator resource isolation module] and managed by [Security management module]. Upon this mechanism, platform users could utilize AI-specific computing resource isolation function described in clause 6.2.2.2. This service could provide exclusive AI computing resource for different workloads and fault isolation between these workloads so that multi-tenants can securely share a single AI accelerator device. As a reference example, <S4> could be provided for administrators to manage the isolation configuration according to different application scenarios and requirements from multi-tenants. <S5> could be provided for AI applications to utilize their designated accelerator resource for training and inference. The recommendation example for deployment is described in clause 9.3.5.

9.3.2 Involved security components

This reference example mechanism involves:

- [AI accelerator resource isolation module] described in clause 7.2.5.
- [Security management module] described in clause 7.4.1.

9.3.3 Reference point and service-based interface

This reference example mechanism involves：

- Reference point：<N3>
- Service-based interface：<S4>

- Service-based interface：<S5>

9.3.4 Mechanism procedure

In this reference implementation example, there are two related procedures for AI-specific computing resource isolation, i.e. resource isolation management and resource isolation implementation.

For resource isolation management, the detailed procedure is shown in Figure 13.

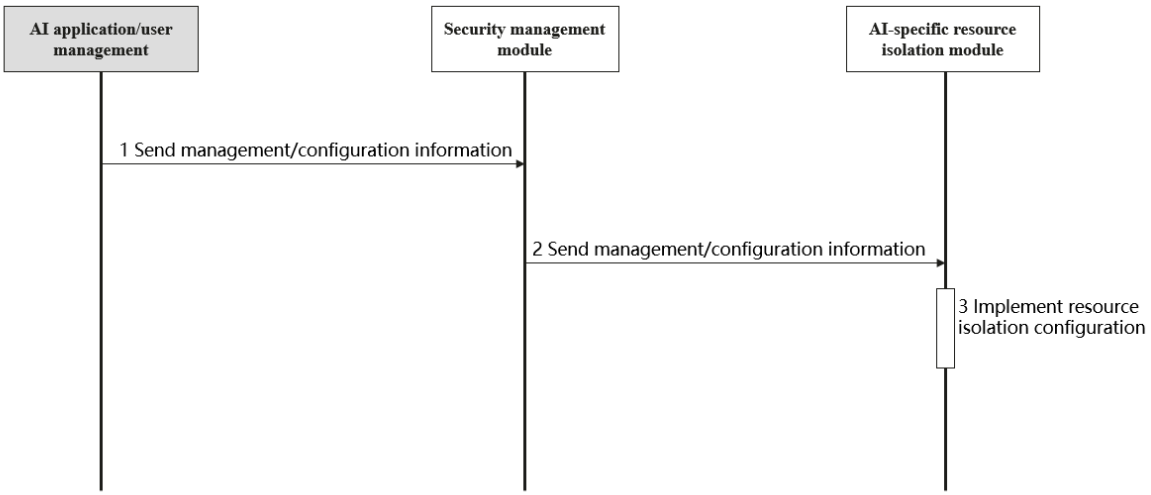

Figure 13: Mechanism sequence diagram for resource isolation management

- Step 1: the AI application or user management software sends management/configuration information to [Security management module] via <S4>.
- Step 2: [Security management module] receives the relevant management information and sends it to [AI-specific resource isolation module] via <N3>.
- Step 3: upon receiving management information, [AI-specific resource isolation module] implements resource isolation management operation according to the configurations sent by users.

For resource isolation implementation, the detailed procedure is shown in Figure 14.

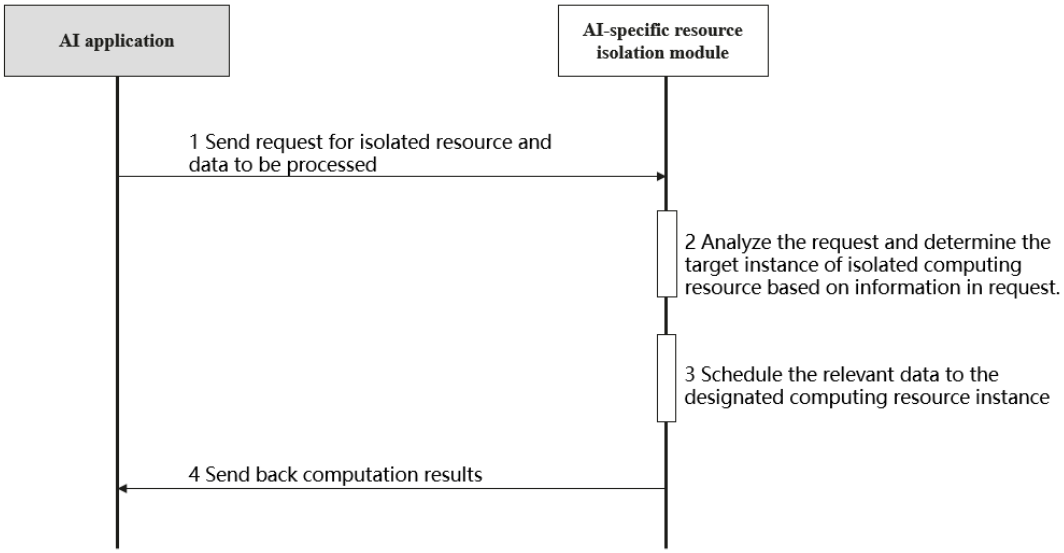

Figure 14: Mechanism sequence diagram for resource isolation implementation

- Step 1: the AI application sends a resource utilization request and data to be processed to [AI-specific resource isolation module] via <S5> aiming to execute inference or model training with the sent data on requested resource.
- Step 2: [AI-specific resource isolation module] determines the target instance of the isolated computing resource based on the user identification information and the requested amount of computing resources.
- Step 3: [AI-specific resource isolation module] schedules the data to the designated instance according to the results from Step 2.
- Step 4: After computation with the sent data on scheduled resource instance, [AI-specific resource isolation module] sends back the computation results to the AI application.

9.3.5 Reference deployment

The reference deployment of an AI-specific computing resource isolation function is illustrated in Figure 15 as an implementation example to better understand the mechanism and its utilization. In an AI system, users can integrate AI-specific computing resource isolation management in their own management software or portal by invoking <S4> when developing code logic of their management software. On the other hand, <S5> can be invoked by virtualization software like hypervisor and container engine in an AI computing platform to implement resource isolation and utilization. The system administrators need to configure the isolation resource instance first and then containers and VMs owned by different users can utilize these instances for computation without interfering with each other.

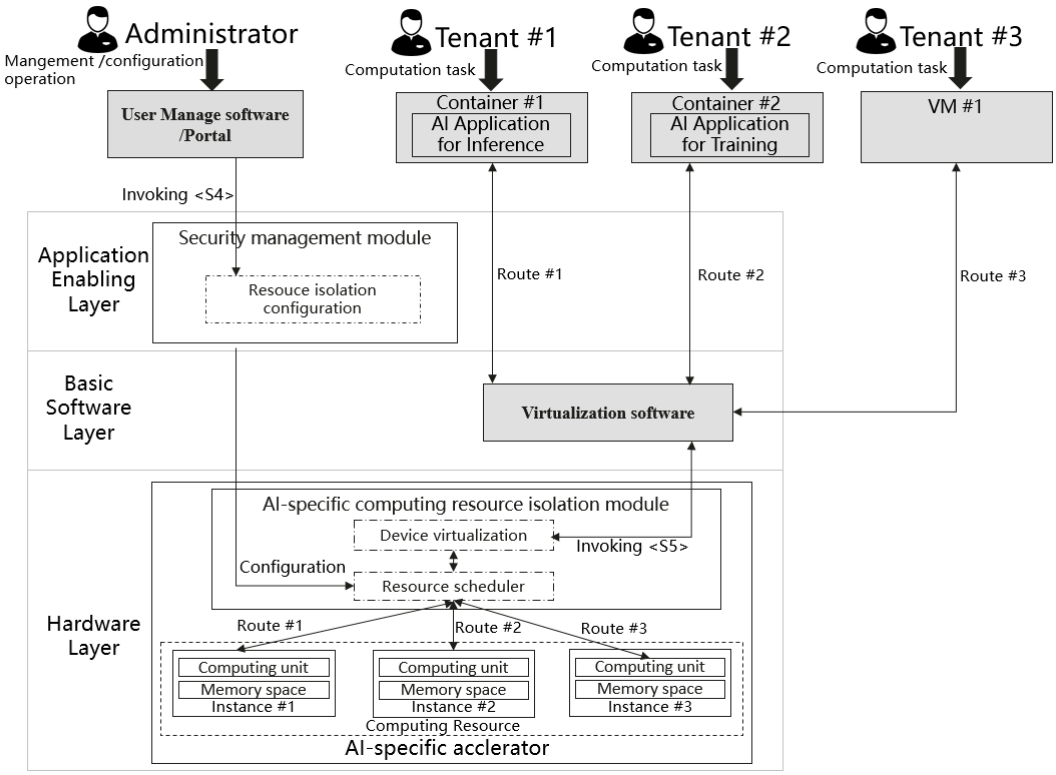

Figure 15: Diagram of reference deployment model for AI-specific computing resource isolation function

9.4 AI related log protection

9.4.1 Description of reference example for the mechanism

As the reference example to implement the mechanism, an AI related log protection mechanism is executed by [log protection module] and an AI application running on the platform. Upon this mechanism, platform users could utilize the AI related log protection service as described in clause 6.2.3. This service provides log protection capability for AI applications and other relevant components in an AI system via an interface <S1> through which logs are sent to the AI computing platform for content and sequence integrity protection. <S1> also provides an interface for log readers to verify whether partial or overall logs have not been tempered in content or in order. The recommendation example for deployment is described in clause 9.4.5.

9.4.2 Involved security components

This reference example mechanism involves:

- [Log protection module] described in clause 7.4.5.

9.4.3 Reference point and service-based interface

This reference example mechanism involves：

- Service-based interface：<S1>

9.4.4 Mechanism procedure

In this reference implementation example, the detailed procedure is illustrated in Figure 16.

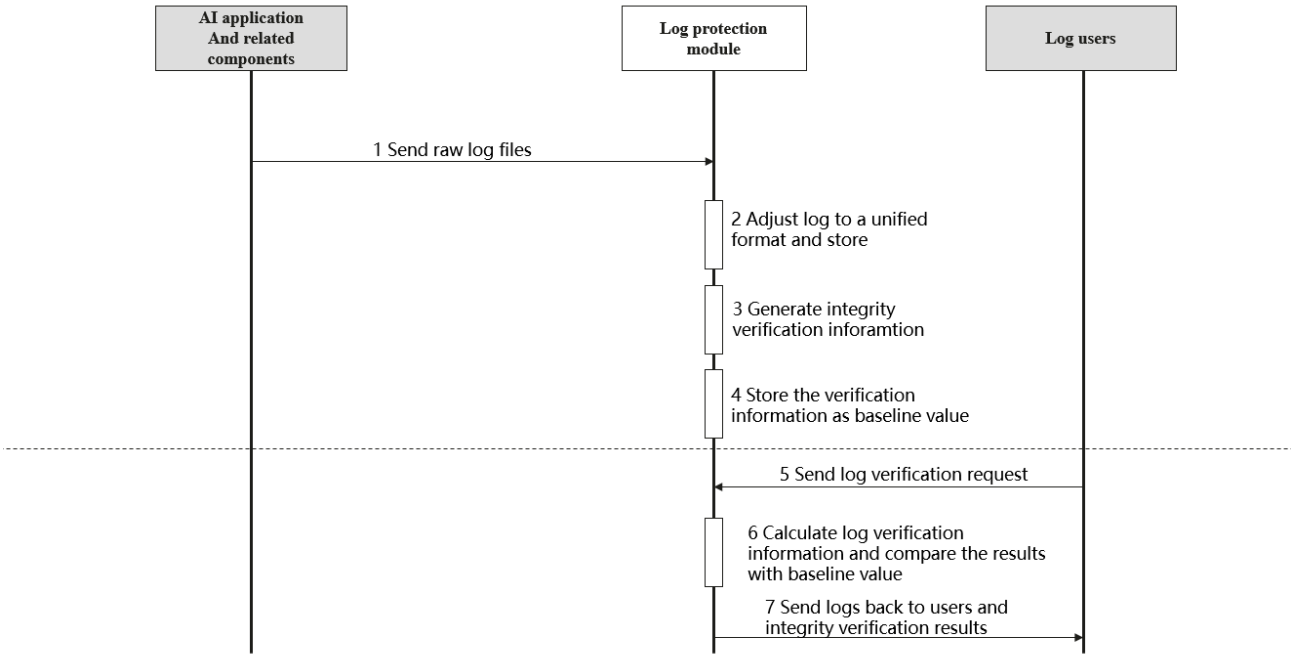

Figure 16: Mechanism sequence diagram for log protection mechanism

For secure storage of AI related logs:

- Step 1: an AI application and related components send raw log files to [Log protection module] via <S1>.
- Step 2: [Log protection module] receives the raw logs, convert these log files to a unified format and store these logs.

- Step 3: [Log protection module] calculates the integrity verification value for these logs utilizing a certain cryptographic method.
- Step 4: [Log protection module] stores the calculated verification value as a baseline value for following verification procedure. The storing of the baseline value can utilize other security techniques.

For log verification:

- Step 5: Log users send a log verification request to [log protection module] via <S1>.
- Step 6: [Log protection module] receives the requests and executes the log verification procedure according to the log information delivered in the requests. [Log protection module] calculates the integrity verification value of the requested logs again and makes a comparison with the baseline value to determine whether the logs requested are correct in content and sequence.
- Step 7: [Log protection module] send back requested log files and verification results via [S1] if verification succeeds.

9.4.5 Reference deployment

The reference deployment of an AI related log protection service is illustrated in Figure 17 as an implementation example to better understand the mechanism and its utilization. In an AI system, different components can send their generated logs into an AI computing platform via <S1> provided by [Log protection module] and different log users like AI system administrator, technique maintainers, and regulator supervisors can request for logs files via <S1> for system oversight, maintenance, tracking and accounting, forensics, etc.

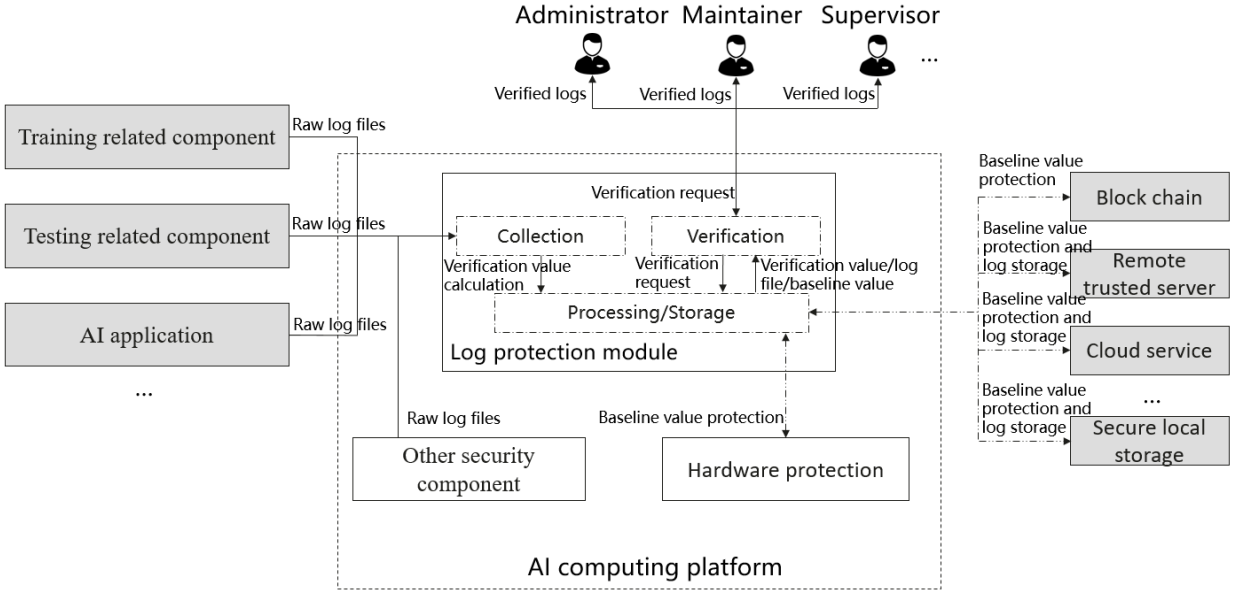

Figure 17: Diagram of reference deployment model for log protection mechanism

In this reference deployment model, different components in an AI system at different model lifecycle stages proactively send log files into the AI computing platform via <S1> provided by [log protection module] for secure storage and protection, or AI computing platform periodically collects log files from these relevant components. On the other hand, log readers like AI system administrator, AI system maintainer and supervisors from government retrieve history AI system logs via <S1> for management, maintenance or forensics objective.

From inside perspective of [log protection module], it executes log files format shaping, cryptographic calculation for integrity verification and storage. One available way for integrity verification is to utilize Merkle tree structure with hash algorithm to calculate the integrity evidence of a sequence of logs. A reference diagram is shown in Figure 18. In Merkle tree, if the root hash is correct comparing to the baseline value, the entire log sets are correct in content and sequence. If the root hash is not correct, the manipulated log files can be located after comparing the calculated hash from high level node to leaf node.

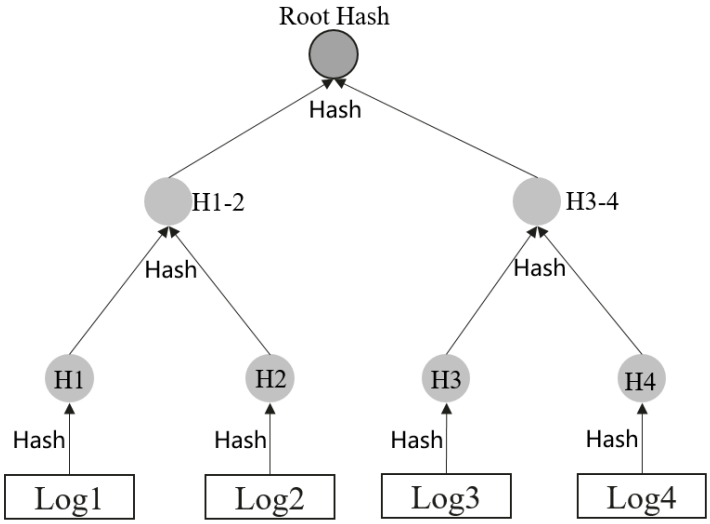

Figure 18: Diagram of Merkle tree for a log set with 4 files

To securely store verification values and log files, a log protection module may utilize some other hardware security components like a TPM and a HMEE security function in an AI computing platform for verification baseline value and log files protection. In addition, the module can also utilize block chain, remote trusted server, cloud service and secure local storage like encrypted SSD to better protect the baseline value and log files. However, these cooperations are implementation-specific and out of the scope of the present document.

9.5 Training procedure recovery

9.5.1 Description of reference example for the mechanism

As the reference example to implement the mechanism, the training procedure recovery mechanism could be executed by a cooperation between a subset of security components. Upon this mechanism, the platform can fulfil the training procedure recovery function as described in clause 6.2.4. This function will recover the training procedure timely when an AI computing platform goes wrong.

9.5.2 Involved security components

This reference example mechanism involves:

- [Hardware abnormality detection module] described in clause 7.2.4
- [System abnormality detection module] described in clause 7.3.1
- [Training procedure recovery module] described in clause 7.4.4

9.5.3 Reference point and service-based interface

This reference example mechanism involves:

- Reference point：<N1>
- Reference point: <N2>

9.5.4 Mechanism procedure

In this reference implementation example, the detailed procedure for the training procedure recovery mechanism is illustrated in Figure 19.

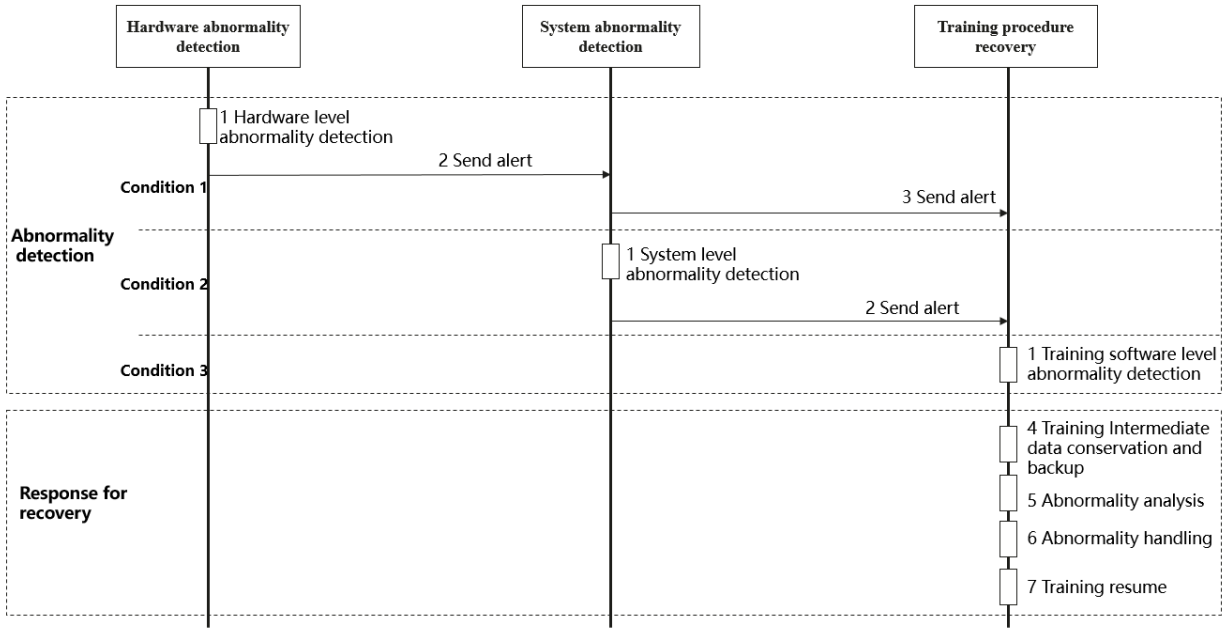

Figure 19: Mechanism sequence diagram for training procedure recovery

The mechanism procedure could be divided into two phases, i.e. abnormality detection and response for recovery. In abnormality phase, there can be three different conditions for detection procedure:

Condition 1: hardware abnormality occurs.

- Step 1: [Hardware abnormality detection module] oversights the AI-specific accelerator's operation state in real time and analyses whether abnormality on accelerator occurs.
- Step 2: When abnormality occurred and is detected, [Hardware abnormality detection module] sends a hardware abnormality alert to [System abnormality detection module] via <N1>.
- Step 3: Upon receiving a hardware abnormality alert, [System abnormality detection module] sends a hardware abnormality alert to [Training procedure recovery module] via <N2>. The procedure enters next phase.

Condition 2: the system abnormality of AI computing platform occurs.

- Step 1: [System abnormality detection module] oversights the system state of the AI computing platform in real time and analyses whether abnormality on system occurs.
- Step 2: When abnormality occurred and is detected, [System abnormality detection module] sends a system abnormality alert to [Training procedure recovery module] via <N2>. The procedure enters next phase.

Condition 3: procedure abnormality of training software occurs.

- Step 1: [Training procedure recovery module] oversights the function operation state of training software in real time and analyses whether abnormality on the training procedure occurs. If abnormality is detected, the procedure enters next phase.

NOTE: The three levels of abnormality detection execute simultaneously to fully cover different levels of system malfunction.

In response for recovery phase, the following steps are executed:

- Step 4: [Training procedure recovery module] conserves and backs up the intermediate data in training at the time point when abnormality occurs in order to support the recovery procedure to the very point when the abnormality happens.
- Step 5: [Training procedure recovery module] analyses the abnormality according to the alerts.
- Step 6: [Training procedure recovery module] executes the relevant abnormality response according to different abnormality causes as determined in the analysis results generated in step 5.

EXAMPLE: Abnormality response includes available resource (includes accelerator level and server node level resource) re-allocation, software/system reboot or recovery.

- Step 7: After abnormality has been addressed, [Training procedure recovery module] executes the training procedure resumption based on the conserved data in step 4.

9.6 Inference attack detection

9.6.1 Description of reference example for the mechanism

As the reference example to implement the mechanism, inference attack detection mechanism could be executed by [inference attack detection module] and an AI application running on the platform. Upon this mechanism, platform users could utilize the inference attack detection function as described in clause 6.2.5. This function could provide inference detection capability for an AI application via an interface <S3> through which an AI application sends a query with inference sample to the detection module and gets back a detection result for further processing. The recommendation example for deployment is described in clause 9.6.5.

9.6.2 Involved security components

This reference example mechanism involves:

- [Inference attack detection module] described in clause 7.4.2.

9.6.3 Reference point and service-based interface

This reference example mechanism involves:

- Service-based interface：<S3>

9.6.4 Mechanism procedure

In this reference implementation example, the detailed procedure for inference attack detection mechanism is illustrated in Figure 20.

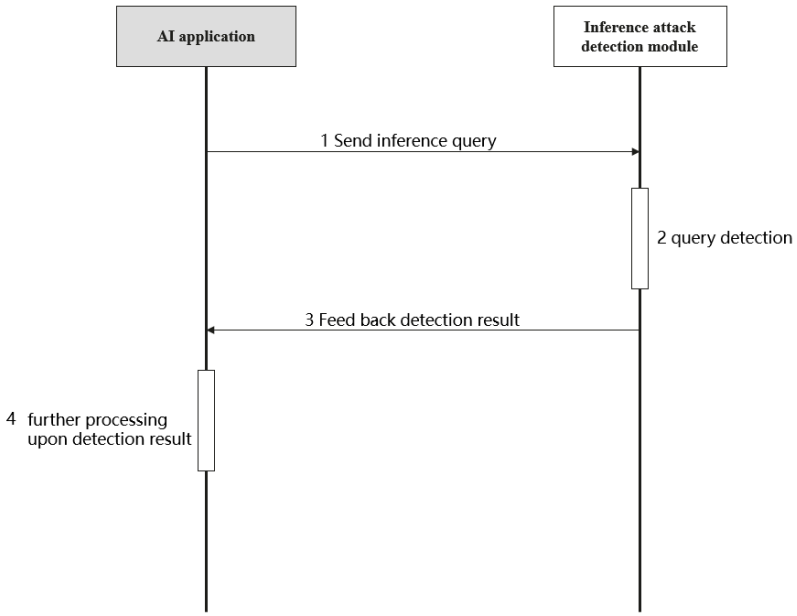

Figure 20: Mechanism sequence diagram for inference attack detection

- Step 1: An AI application sends an inference query to [Inference attack detection module] via <S3>.
- Step 2: [Inference attack detection module] receives the query and executes the attack detection on the query.
- Step 3: [Inference attack detection module] feeds back the detection result to the AI application.
- Step 4: The AI application executes further processing upon detection result.

9.6.5 Reference deployment

The reference deployment of inference attack detection function is illustrated in Figure 21 as implementation example to better understand the mechanism and its utilization. In AI inference scenarios, an AI application usually includes an API gateway which acts as a proxy to receive inference queries from AI users and send back inference results, and an inference engine model which loads well-trained inference model and executes inference computation. In [Inference attack detection module], the interface module acts as a proxy to communicate with an AI application via an API. The detection model is a well-trained model to execute inference on the received query sample to detect the possibility of attack. The security policy sub-module acts as an decision maker to produce a final detection results for an AI application based on a predefined security policy and detection results generated from the inference of the detection model.

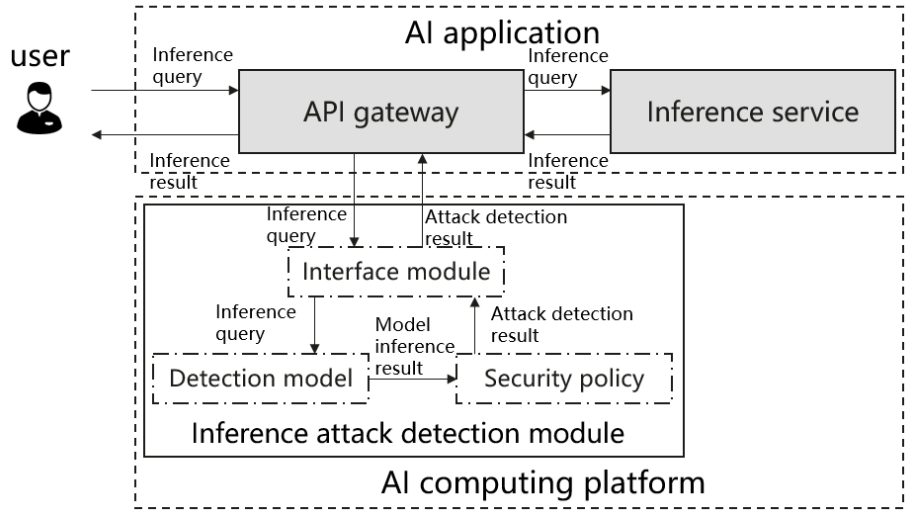

Figure 21: Reference deployment model of inference attack detection function

There are two reference models to utilize this security function:

- The first one refers to the idea of direct deployment model where attack detection results are directly utilized to determine whether this being-detected query can be sent to an inference service for inference computing. In this reference model, the API gateway will not forward inference query to the inferring model unless it receives an attack detection results that shows a benign result from the inference attack detection module. In this reference model scenario, the detection results of the inference attack detection module will directly affect the inference procedure of the AI application and exclude more malicious queries for an AI application, but more delay will be brought in as one more inference computation is incurred in one inference procedure. It should be noted that the inference attack detection module is only responsible for the detection of inference query and sample in it. The logic of forwarding a query to the inference service is included in the API gateway module, which is out of the scope of an AI computing platform. The diagram is illustrated in Figure 22 (a).

- The second one refers to the idea of a bypass deployment model where attack detection results will not affect the determination whether this being-detected query can be sent to the inference service for inference computing, but to be a reference proof or a system alert to the AI application for further response to the following query. Compared to the first model, bypass deployment has less processing delay, but can exclude less malicious queries as the API gateway does not depend on the detection result for forwarding decision like blocking the next query from this source, sending an alert to the application user or sanitizing the sample. In this model, the follow-up response logic is implemented in the API gateway module which is out of the scope of the AI computing platform, while the inference attack detection module only provides the detection results to the API gateway. The diagram is illustrated in Figure 22 (b).

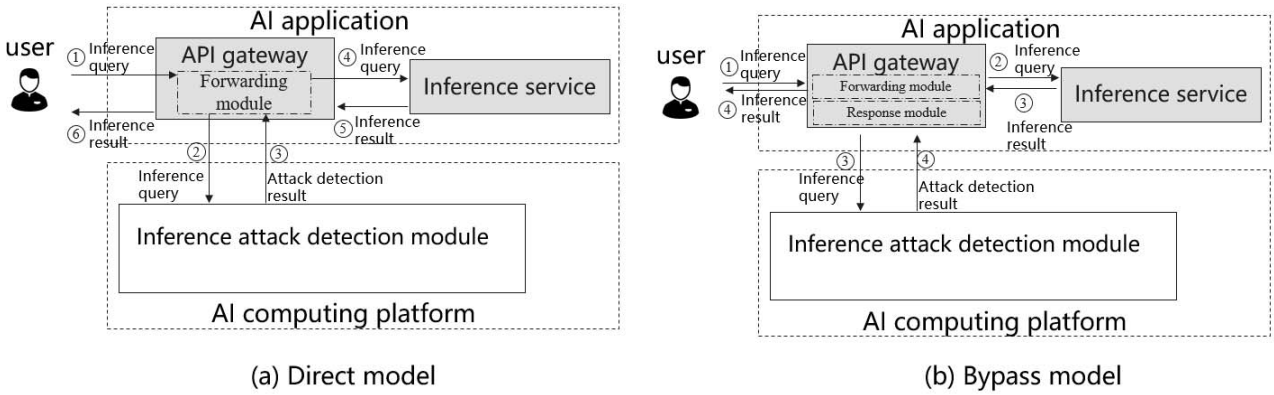

Figure 22: Diagram of two reference deployment model

9.7 Measured boot

In an AI computing platform, the host and AI accelerators need to implement measured boot procedure independently upon each own [Host trusted boot module] and [AI accelerator trusted boot module] as the initialization code, respectively. The measure results of host components during the boot should be recorded into [Host HBRT]. Meanwhile, the measure results of components on the AI accelerator should be recorded into [AI accelerator HBRT].

9.8 Recovery from minimal system

As a reference example for system recovery, platform users can recover an AI computing platform from an unexpected state to a predefined initial state utilizing the function provided by <S6>. In this mechanism, <S6> could be provided by [minimal system] and can be invoked by user management software so that system recovery function can be integrated into the management portal. A recommended model for the deployment of recovery is to utilize BMC chip as the [minimal system] and integrate BMC interface into device management portal for recovery operation.

Annex A: Bibliography

ETSI TS 102 165-1: "CYBER; Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA)".

History

Document history		
V1.1.1	February 2023	Publication