NIST Interagency Report
NIST IR 8320E ipd

Hardware-Enabled Security

Confidential Computing of Data in Cloud Workloads

Initial Public Draft

Michael Bartock
Murugiah Souppaya
Timothy Knoll

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320E.ipd

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Interagency Report
NIST IR 8320E ipd

Hardware-Enabled Security

Confidential Computing of Data in Cloud Workloads

Initial Public Draft

Michael Bartock
Computer Security Division
Information Technology Laboratory

Murugiah Souppaya
**Former NIST employee; all work for this publication was done while at NIST.*

Timothy Knoll
Intel Corporation

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320E.ipd

May 2026

U.S. Department of Commerce
Howard Lutnick, Secretary

National Institute of Standards and Technology
Craig Burkhardt, Acting Under Secretary of Commerce for Standards and Technology and Acting NIST Director

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

NIST Technical Series Policies
Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on YYYY-MM-DD
Supersedes NIST Series XXX (Month Year) DOI

How to Cite this NIST Technical Series Publication:
Bartock M, Souppaya M, Knoll T (2026) Hardware-Enabled Security: Confidential Computing of Data in Cloud Workloads. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency Report (IR) NIST IR 8320E ipd. https://doi.org/10.6028/NIST.IR.8320E.ipd

NIST Author ORCID iDs
Michael Bartock: 0000-0003-0875-4555
Murugiah Souppaya: 0000-0002-8055-8527

Public Comment Period
May 29, 2026 – July 13, 2026

Submit Comments
hwsec@nist.gov
National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/pubs/ir/8320/e/ipd, including related content, potential updates, and document history.

All comments are subject to release under the Freedom of Information Act (FOIA).

Abstract

Confidential computing addresses data security and privacy concerns for organizations that move sensitive workloads to the cloud. It is a critical advancement that enables the encryption of data both while it is being processed in memory and in active use. As cloud adoption continues to grow, confidential computing will play a pivotal role in improving security and privacy in cloud environments. This report describes an effective approach to protecting data being acted upon by artificial intelligence workloads on cloud infrastructures so that the datasets are protected from malware, data theft, and other security-related vulnerabilities.

Keywords

Confidential computing; cryptographic key; hardware-enabled security; machine identity; machine identity management; trusted execution environment (TEE).

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Audience

The primary audiences for this report are security professionals, such as security engineers and architects; system administrators and other information technology (IT) professionals responsible for securing physical or virtual platforms; and hardware, firmware, and software developers who may be able to leverage hardware-enabled security techniques and technologies to improve machine identity management and protection.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or

   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: hwsec@nist.gov

Table of Contents

List of Figures

1. Introduction

1.1. Purpose and Scope

Confidential computing addresses one of the most pressing concerns for organizations that move sensitive workloads to the cloud: data privacy. In traditional cloud environments, data is typically encrypted while at rest and in transit but decrypted during processing, which exposes potential vulnerabilities. Confidential computing fills this gap by enabling the encryption of data even while it is being processed in memory, closing the gap that traditional at-rest and in-transit encryption leaves unaddressed.

This innovation is important for industries that handle highly sensitive information, such as finance, healthcare, and government. With confidential computing, organizations can confidently move their sensitive workloads to the cloud, where the data being processed can be made opaque to the cloud provider, thus reducing the risk of access by privileged infrastructure software. This not only enhances trust in cloud environments but also opens up new possibilities for collaboration and innovation, such as sharing data securely across different organizations with greater protection against unauthorized access. As cloud adoption continues to grow, confidential computing will play a pivotal role in improving security and privacy in cloud environments.

This report describes an effective approach to protecting data being acted upon by artificial intelligence (AI) workloads (i.e., data used for inference) on cloud infrastructures so that the datasets are protected from malware, data theft, and other security-related vulnerabilities. The reference implementations detailed in the appendices show how to perform inferences on sample health data with an encrypted artificial intelligence model.

The prototype implementation presented in this report is only one possible way to solve the security challenges. It is not intended to preclude the use of other products, services, or techniques that can also solve the problem adequately, nor is it intended to preclude the use of any cloud products or services that are not specifically mentioned in this report.

This report builds upon the terminology and concepts described in NIST Interagency Report (IR) 8320, *Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases* [1].

1.2. Terminology

For consistency with related NIST publications, this report uses the following definitions for trust-related terms:

- **Trust:** “The confidence one element has in another that the second element will behave as expected” [2]
- **Trusted:** An element that another element relies upon to fulfill critical requirements on its behalf

- **Trustworthy:** Worthy of being trusted to fulfill whatever critical requirements may be needed

1.3. Document Structure

This document is organized into the following sections and appendices:

- Section 2 defines the objective for the prototype implementation and the intermediate goals to be met in order to achieve the objective.
- Sections 3 through 6 describe the four stages of prototype implementation.
- The References section lists the references cited throughout this document.
- Appendix A provides an overview of the solution architecture.
- Appendix B describes how to enable a trusted execution environment (TEE).
- Appendix C describes how to set up key management and relying party services.
- Appendix D describes a reference AI model encryption implementation.
- Appendix E describes remote attestation of the TEE implementation.

2. Challenges with Protecting Data in Use

In traditional computing environments, data stored in memory would be unencrypted while applications performed actions on it through the system's processor. However, recent technological advances have allowed mechanisms to encrypt memory space and restrict which processes on a system can decrypt that memory while in use. The ultimate goal is to be able to use hardware-enabled mechanisms to protect data being processed by an AI model so that it is accessible only to authorized code within the trust domain. This involves encrypting the AI model's memory while in use and making sure that only compute resources that can attest to their hardware capabilities are given the decryption key to access the memory space.

- **Stage 0: Configure a TEE-capable virtual machine (VM)**

  Many cloud service providers offer virtual machines that have TEE capabilities enabled. Section 3 describes the benefits of utilizing a TEE and how it can be configured within a CSP.

- **Stage 1: Remote attestation service configuration**

  Workloads that use confidential computing capabilities require a way to be trusted that they are implemented and enabled. Section 4 describes how remote attestation of workload achieves this goal.

- **Stage 2: Relying party configuration**

  Once a workload has attested that it has confidential computing capabilities enabled, policies can be applied for what kind of information it can be allowed to access. Section 5 describes how a relying party can make these types of determinations based on attestation reports.

- **Stage 3: Execute the key-release workflow**

  Section 6 brings the previous stages together to depict how a workload goes from instantiation to processing sensitive information by using protected keys in a confidential computing environment.

2.1. Objective

A reference implementation that protects AI workloads with confidential computing introduces a powerful layer of security by leveraging encryption keys for the AI model memory space for the data it processes. In the context of AI, both the model and the data it uses are valuable assets. The model — often developed through significant research and computational effort — holds intellectual property and represents a competitive advantage for organizations. Similarly, the data, which could contain sensitive or proprietary information, is crucial for accurate and effective AI processing. Confidential computing can safeguard both of these elements by isolating them in secure enclaves where they are encrypted, even when processed in the cloud or while in active memory [3].

Encrypting the AI model provides strong protection for both the model itself and the data it processes. By securing the model with an encryption key that is only accessible to the deploying entity, organizations can restrict model execution to a trusted, controlled environment. This significantly reduces the risk of unauthorized access or tampering, even from privileged infrastructure software, when the platform is patched and its configuration is verified through attestation. Because the model is encrypted, any data it processes is also protected from indirect exposure. Attackers face a substantially higher barrier to extracting insights, parameters, or hidden representations from the model to infer sensitive details about the input data. Instead, decryption and execution occur within a secure enclave, where both the model and the data can interact safely within hardware-enforced isolation.

This dual protection is especially important in regulated industries, where both intellectual property (i.e., the model) and sensitive information (i.e., the data) must be secured. Encrypting the model preserves proprietary algorithms and reinforces compliance with data protection regulations by providing hardware-enforced controls that are designed to prevent information handled by the model from being misused or exposed outside of authorized computations.

By combining model encryption with confidential execution, organizations can confidently deploy AI in the cloud while safeguarding both their innovations and the sensitive data that those models process.

2.2. Goals

The goal of this paper is to explore and detail the architecture that implements the protection of AI workloads using confidential computing, with a particular focus on the use of encryption keys for the AI model and implications for the data it processes. Examining this architecture will illustrate how confidential computing can be effectively applied to safeguard both the intellectual property embedded in AI models and the sensitive data they process, even when operating in potentially untrusted cloud environments.

This discussion will delve into the technical specifics of how the architecture isolates and encrypts AI models and data within secure enclaves designed so that only authorized computations can access and process them. Additionally, this paper will discuss the rationale behind using encryption keys and the benefits of this approach in terms of enhanced security and compliance with data protection regulations. Through this exploration, readers will gain a clear understanding of how this architecture is implemented and the practical implications for securing AI workloads in the cloud.

3. Stage 0: Configure a TEE-Capable VM

TEE attestation is crucial because it provides verifiable proof that an application or workload is running within a secure and isolated environment. In the context of cloud computing, where workloads often run on shared infrastructure, the ability to trust that data and computations are securely isolated is essential. TEE attestation allows cloud providers to demonstrate to users that their workloads are running in a TEE whose hardware and software configuration matches expected values. This assurance is key to building trust between cloud providers and users, especially when handling sensitive or critical data.

Furthermore, TEE attestation is important for maintaining the integrity and confidentiality of the data being processed. By using attestation, organizations can verify that the code running in the TEE matches expected measurements and has not been modified in unexpected ways. This is particularly important in scenarios where the integrity of the data and the computation is paramount, such as in financial services, healthcare, and government applications. TEE attestation provides a mechanism for verifying the trustworthiness of the execution environment to securely and reliably process sensitive workloads in potentially untrusted cloud environments.

Hardware-based confidential computing provides robust protection for AI data by leveraging secure enclaves within the processor to create isolated environments in which sensitive computations can occur without exposing data to the rest of the system under normal operation. These secure enclaves are designed to protect code and data from access by the operating system and hypervisor, with isolation enforced in hardware. When AI models and the data they process are loaded into these enclaves, they are encrypted and only decrypted within the enclave's boundaries, significantly raising the barrier to unauthorized access from external threats, including privileged software and other tenants.

This protection is particularly crucial for AI workloads, where both the data and the models are valuable assets. AI models are significant intellectual property that may use highly sensitive information, such as personal data or proprietary business insights. By isolating these components in a hardware-protected environment, confidential computing is designed to protect both the AI model and the data from unauthorized access or tampering during processing.

Additionally, hardware-based confidential computing enhances trust in AI systems, especially in cloud environments where resources are shared among multiple users. Ensuring that data and models remain encrypted and secure even when processed is particularly important in regulated industries like finance and healthcare, where compliance with data protection regulations is critical. Overall, hardware-based confidential computing provides a powerful solution for protecting AI data by enabling secure, efficient, and compliant AI processing in the cloud.

Many cloud service providers offer VMs that have TEE capabilities enabled. In order to use them, they must be launched through the corresponding CSP's portal/console, for which an

account and payment method is typically required. Once a TEE-capable VM has been instantiated, certain steps are required to connect a relying party and attestor:

- Dockerfile to build the Intel Key Broker Service (KBS) relying party container
- **KBS.env** environment file to configure the KBS, modified to suit a particular environment

The steps in a provisioning workflow are:

- Create an RSA key pair.
- Push the public key to the relying party.
- The replying party creates a wrapped key for encryption.
- Encrypt the ML model with the private key.

4. Stage 1: Remote Attestation Service Configuration

Remote attestation is a critical component of ensuring trust in confidential VMs in cloud computing environments. It allows a remote party (e.g., user, trusted workload orchestrator) to verify that a VM is running in a secure and expected environment and in a hardware and software configuration that matches expected values. This is especially important in multi-tenant cloud scenarios where sensitive workloads may be deployed on shared infrastructure. Remote attestation provides cryptographic evidence that a VM is running on hardware equipped with TEE capabilities and that the VM's memory and state are isolated and protected from both other tenants and the hypervisor on a patched and attested platform.

The process of remote attestation typically begins with the generation of a hardware-backed attestation report by the TEE. This report includes measurements of the VM's initial state (e.g., firmware, bootloader, operating system, application code) and a unique identifier for the platform. These measurements are securely signed by the TEE's attestation key, which is rooted in hardware and vouched for by the processor manufacturer. The VM or a trusted software component inside it transmits this report to a remote verifier, which may be a tenant or an external trust service.

The remote verifier then checks the attestation report's authenticity and integrity by validating the cryptographic signature and comparing the measurements against known good or expected values. These measurements can be forwarded to a relying party to evaluate whether the requesting VM or service should be given access to secrets or keys.

5. Stage 2: Relying Party Service Configuration

Relying parties play a critical role in enabling TEEs for confidential computing in cloud environments. TEEs are isolated regions of memory within a CPU that provide a secure enclave for code and data to protect them from access by privileged system software, including the hypervisor and OS. Relying parties (e.g., application owners, external verifiers) depend on these assurances and interact with TEEs to establish trust before processing sensitive data. As organizations increasingly migrate sensitive workloads to the cloud, concerns about data confidentiality and integrity — especially from potentially untrusted infrastructure providers — have driven the adoption of TEEs.

Relying parties often enable the use of TEEs through remote attestation, which is a cryptographic protocol that allows the TEE to prove to a relying party that the code running inside the secure enclave matches expected measurements. The relying party verifies this attestation (e.g., using endorsements from hardware vendors) to ensure that the execution environment meets predefined security policies. Only after successful attestation does the relying party provision sensitive data or secrets into the enclave, so that the data is not exposed outside of the secure boundary under normal operation.

Relying parties also influence the life cycle of confidential workloads by defining policies for enclave code versions, revocation statuses, and the integrity of data pipelines. For instance, they may require specific measurements (i.e., cryptographic hashes) of the application code before trusting the enclave. This level of control enables fine-grained governance and accountability, particularly in regulated industries like healthcare and finance. Moreover, by integrating with cloud-native tools and confidential computing platforms, relying parties can automate these trust evaluations and maintain confidentiality without significantly impacting performance or scalability.

Relying parties are foundational to the trust model of confidential computing in the cloud. Their role in verifying and enforcing the security properties of TEEs provides meaningful assurance that data confidentiality can be maintained during processing, even in untrusted or multi-tenant cloud environments. As TEEs and attestation frameworks mature, the collaboration between cloud providers and relying parties will become even more critical to securing the next generation of cloud-native applications.

6. Stage 3: Execute the Key-Release Workflow

In a confidential computing model, the key-release workflow is central to enabling secure data inference, particularly in scenarios where sensitive data must be processed within a TEE. This workflow (see Fig. 1) is designed to release encryption keys that are used to decrypt model inputs, parameters, or outputs only to verified and trusted execution environments, supporting confidentiality and integrity.

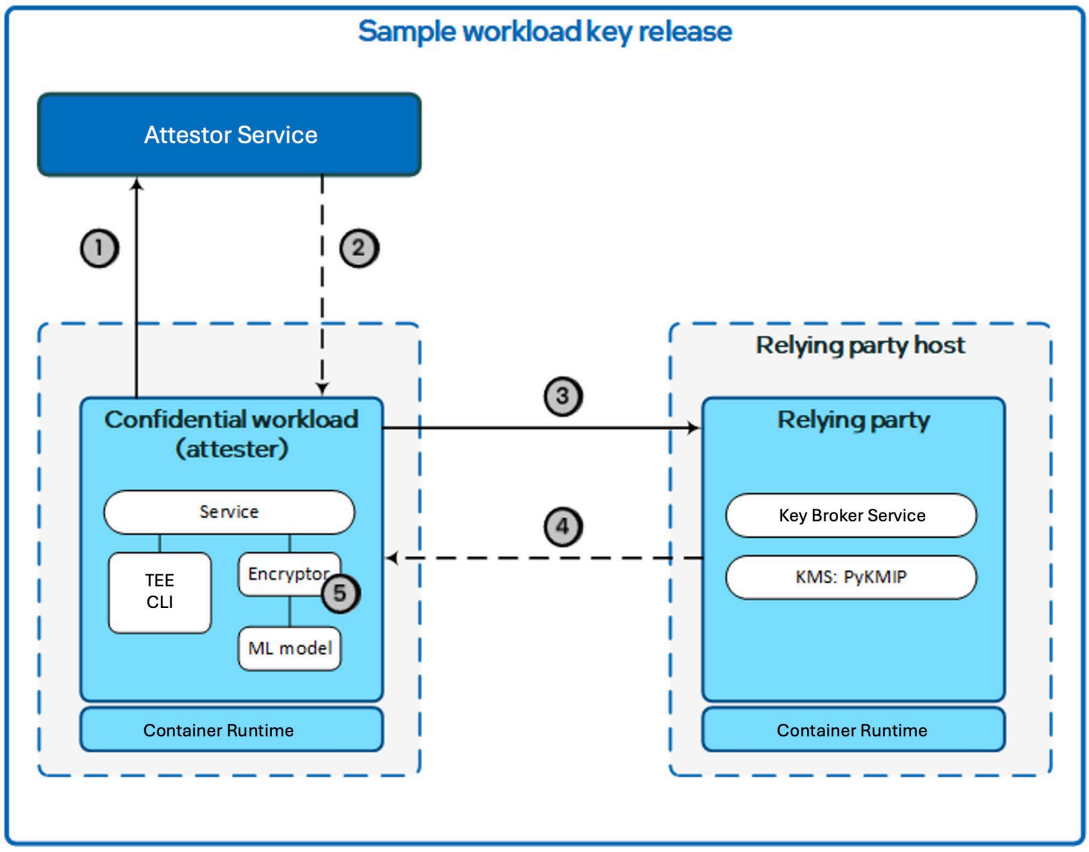

Fig. 1. Sample workload key release

The process typically begins with remote attestation, where the TEE presents cryptographic evidence (e.g., attestation report, quote) that describes its identity, the code it is running (e.g., a hash, measurement), and its security state. This evidence is sent to a key management or attestation service, which compares the attestation data against predefined trust policies set by the data owner or a relying party. These policies may require the enclave to run a specific, approved version of the machine learning model; use up-to-date TEE firmware; and be deployed on verified hardware.

If the attestation is successful and complies with policy, the key management service proceeds with key release. This usually involves encrypting the decryption key in a way that only the TEE can access, often leveraging hardware-protected key-wrapping or sealing techniques. The key is then transmitted to the enclave, where it can be safely unwrapped and used to decrypt the encrypted model or data inputs. The key is not exposed outside of the TEE in plaintext form

under normal operation, which protects it from potential compromise in multi-tenant cloud environments when deployed on a patched, attested platform. Once the key is released and the data is decrypted inside the enclave, the model can securely perform inference.

The output may also be encrypted before leaving the TEE, depending on the desired level of protection. This end-to-end trust chain — from attestation through key release to data processing — allows sensitive workloads (e.g., healthcare predictions, financial risk analysis, personal data classification) to be run in the cloud while limiting the exposure of data to the platform provider or other tenants. The key-release workflow is therefore a cornerstone of secure, scalable confidential computing for AI inference.

References

[1] Bartock M, Souppaya M, Savino R, Knoll T, Shetty U, Cherfaoui M, Yeluri R, Malhotra A, Banks D, Jordan M, Pendarakis D, Rao JR, Romness P, Scarfone KA (2022) Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) NIST IR 8320. https://doi.org/10.6028/NIST.IR.8320

[2] Polydys ML, Wisseman S (2009) Software Assurance in Acquisition: Mitigating Risks to the Enterprise. Available at https://apps.dtic.mil/sti/tr/pdf/ADA495389.pdf

[3] Confidential Computing Consortium (2022) A Technical Analysis of Confidential Computing. Available at https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

Appendix A. Solution Architecture

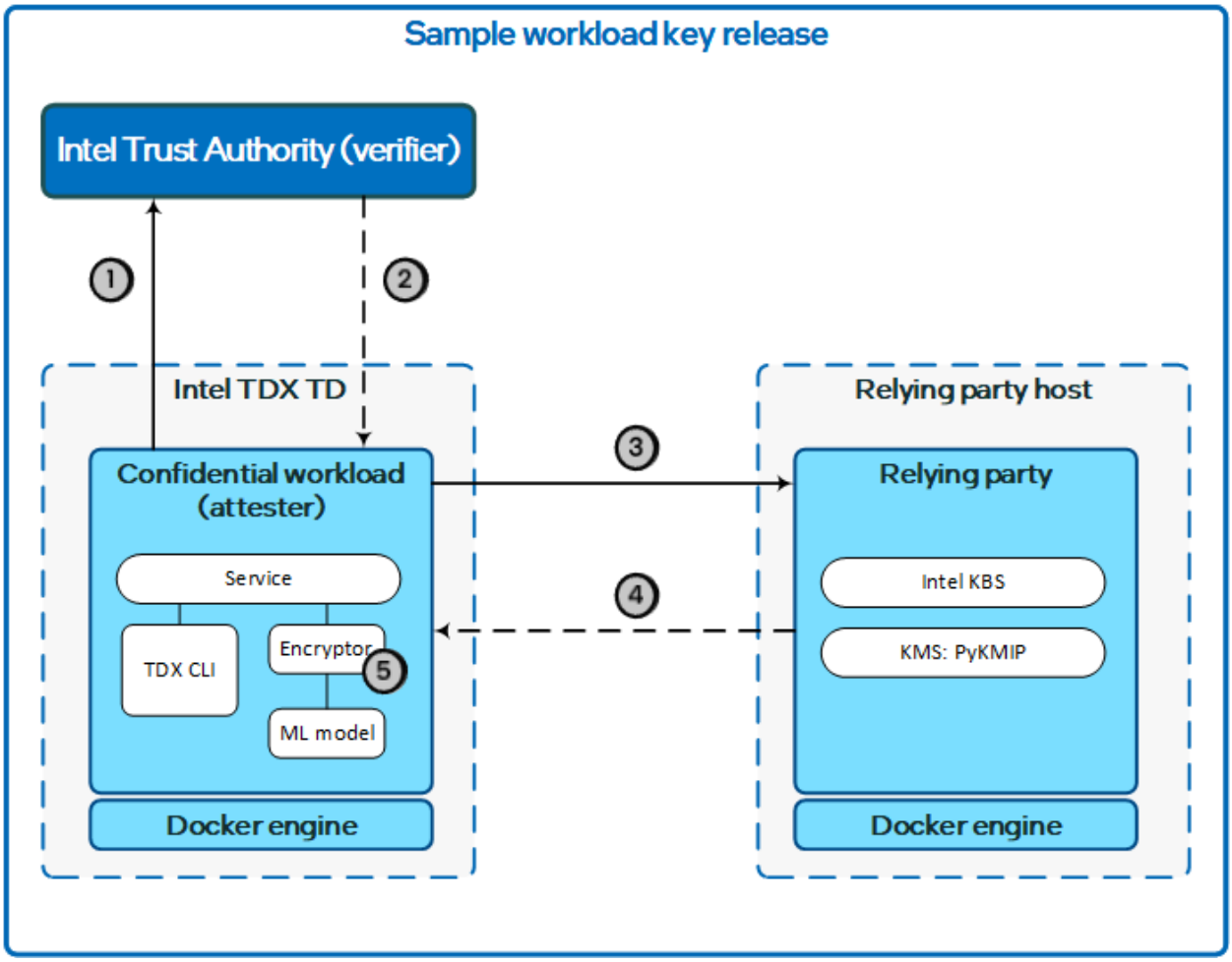

Fig. 2. Cloud workload key release

1. The attesting workload obtains a quote from the Intel TDX TD by using the Intel Trust Authority CLI for Intel TDX ("TDX CLI"), which is also used to request an attestation token from the Intel Trust Authority.
    a. The workload service creates an RSA key pair. The private key is retained by the service, and the public key is base64-encoded and then copied to the user-data parameter in the token command. The user data is output in the **attester_held_data** claim in the attestation token. The public key is used by the KBS to encrypt (i.e., encapsulate) the key request response body.
2. The Intel Trust Authority evaluates the quote and returns an attestation token.
3. The workload forwards the attestation token to Intel KBS in a Representational State Transfer (REST) API request for the decryption key needed to decrypt the ML model.
4. The KBS evaluates the request and applies its key-release policy. An Intel KBS key-release policy compares values in the attestation token to reference values stored in the key-release policy. If the key-release policy evaluation is successful (i.e., all compared

values match), the KBS contacts the key management system (KMS) to obtain the decryption key. The KMS will also apply its own key-release policy, which is independent of the KBS policy. If the KMS policy allows, the KMS will return the requested key or secret to the KBS. The KBS uses the public key passed in the attestation token to wrap the KMS-supplied decryption key before sending it back to the workload.

   a. Intel KBS creates a symmetric AES key to encrypt the KMS-supplied key in the response body of a key request. This key is called the symmetric wrapping key (SWK). For enhanced security, Intel KBS can also use asymmetric key encapsulation with attestation tokens provided by Intel Trust Authority. Asymmetric key encapsulation uses a public key provided by the attester to encrypt the response body, including the SWK.

5. The workload service decrypts the Intel KBS response by using the RSA private key from step 1.1. The service extracts the SWK from the response and uses it to decrypt the KMS-supplied key, which can then be used to decrypt the ML model. The model is now ready for use.

Appendix B. Trusted Execution Environment Implementation

B.1. Creating a VM With Intel TDX on Microsoft Azure

To create an Azure confidential VM with Intel TDX, create a VM with the following attributes:

- Security type: Trusted launch virtual machine
- Image: Ubuntu Server 22.04 LTS
- Size: DC2esv5

The following steps create an Azure VM with these attributes:

1. Sign in to Azure.
2. Select “Create a resource.”
3. Select “virtual machine.”

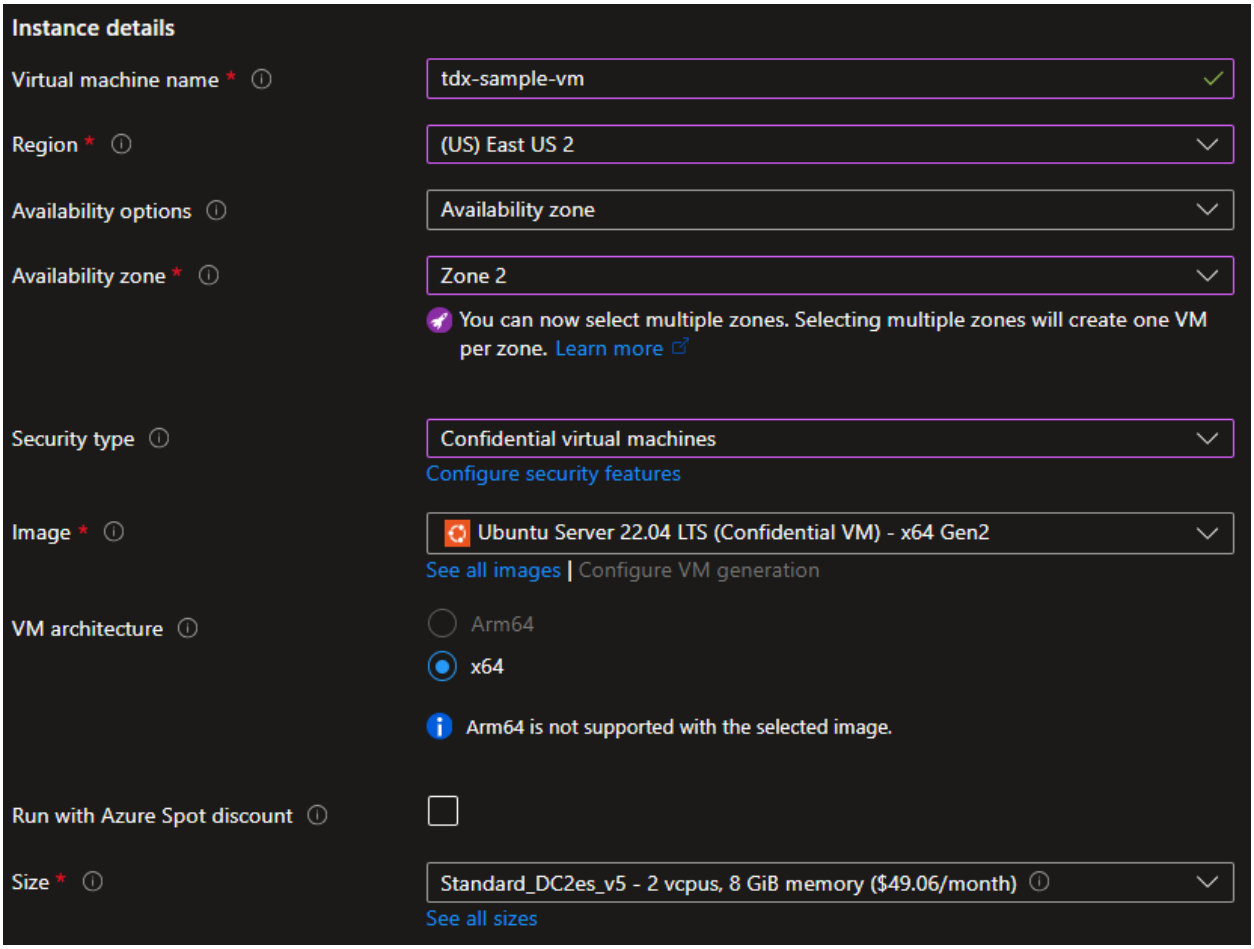

Fig. 3. Creating a cloud VM

4. Complete the following fields:
   - Virtual machine name: Give your virtual machine a name.

- Region: Select **(US) West US 2**.
- Availability options: Select **Availability zone**.
- Availability zone: Select **zone 2**.[1]
- Security type: Select **Trusted launch virtual machine**.
- Image: Select **Ubuntu Server 22.04 LTS - x64 Gen2**.
- Size: Select **DC2esv5**.[2]
- Public inbound: **None**.

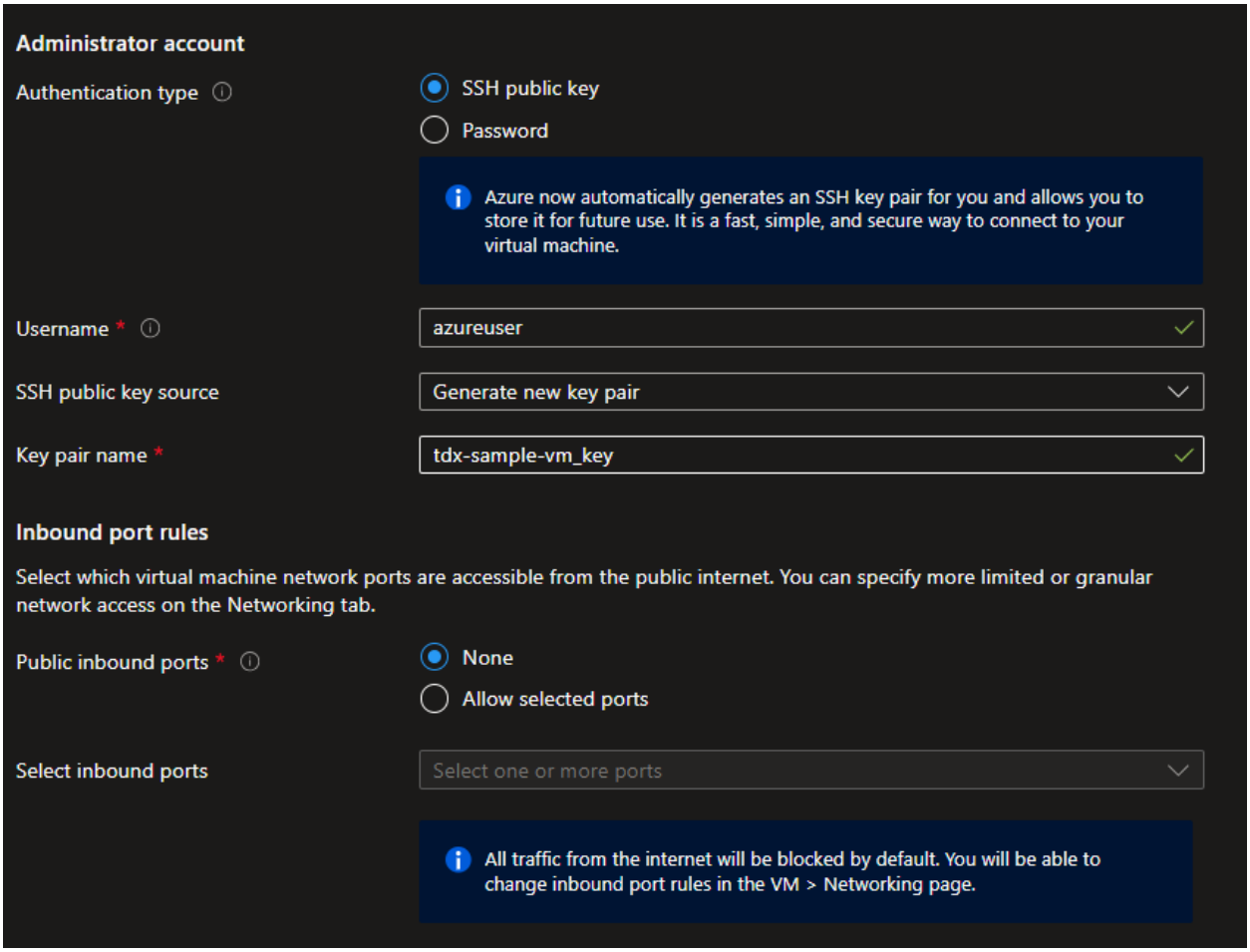

Fig. 4. Creating a cloud VM (continued)

- On the Networking page: For Public IP, select **None**.

5. Review the options and then create the VM. Deployment typically takes a few minutes.

[1] The availability of specific confidential virtual machine images and sizes in specific regions and availability zones is dynamic and may change. Check the Azure Products by Region page to find the regions and availability zones with available Confidential VM support.
[2] If not immediately listed, you may need to select "**see all sizes**" to choose **DC1s_v3** from the full list.

B.2. Deploying Bastion

1. Select the VM resource.
2. Select **Connect** and then select **Bastion**.
3. Select **Deploy Bastion**. Bastion deployment takes a few minutes.
4. Let Azure create a new SSH key pair. Download the private key to connect to the VM.
5. Connect using “SSH private Key from Local File.”
6. Enter the username.[3] ().
7. Set the “local file” to the downloaded key <vm_name_key.pem>.

B.3. Configuring TDX Prerequisites

Verify that the /dev/tpmrm0 device exists.

```
ll /dev/tpmrm0
crw-rw---- 1 tss tss 253, 65536 Jan  2 03:11 /dev/tpmrm0
```

This is the device used by Microsoft Azure to expose Intel TDX functionality. Other providers and bare-metal hardware may use a different device name, usually /dev/tdx_guest.

Verify that the Intel TDX feature is active.

```
sudo dmesg | grep -i tdx
[    0.853076] Memory Encryption Features active: Intel TDX
```

In this example, a TEE is achieved using Intel TDX, as provided by Microsoft Azure’s Confidential VMs.

[3] The default username is “azureuser.”

Appendix C. Key Management and Relying Party Implementation

C.1. Deploy the Key Broker Service

- Prerequisites
- Makeself
- Go v1.21 or newer
- Create an attestation API key via the Intel Trust Authority web portal (requires subscription).

Download the kbs.env file, and update the configuration.

```
wget https://raw.githubusercontent.com/intel/trustauthority-samples/main/deployment/kbs/kbs.env
```

When the relying party container is run, a startup script configures Intel KBS to use the ADMIN_USERNAME and ADMIN_PASSWORD provided in the kbs.env file. An attestation API key will also need to be provided for TRUSTAUTHORITY_API_KEY, which can be obtained from the Intel Trust Authority portal. An HTTPS proxy may also need to be configured, depending on the network configuration. In the following sample kbs.env file, replace the parameters marked with angle brackets (e.g., <admin-username>).

```
LOG_LEVEL=INFO
KEY_MANAGER=kmip
ADMIN_USERNAME=<admin-username>
ADMIN_PASSWORD=<admin-password>
KMIP_VERSION=2.0
KMIP_SERVER_IP=127.0.0.1
KMIP_SERVER_PORT=5696
KMIP_CLIENT_KEY_PATH=/etc/pykmip/client_key.pem
KMIP_CLIENT_CERT_PATH=/etc/pykmip/client_certificate.pem
KMIP_ROOT_CERT_PATH=/etc/pykmip/ca_certificate.pem
PYKMIP_LOG_LEVEL=info
HTTPS_PROXY=
TRUSTAUTHORITY_BASE_URL=https://portal.trustauthority.intel.com
TRUSTAUTHORITY_API_URL=https://api.trustauthority.intel.com
TRUSTAUTHORITY_API_KEY=<API Key>
```

The kbs.env file contains secrets (e.g., Intel KBS administrator password, KMS password or access token) and an Intel Trust Authority attestation API key. For a secure production system, delete the kbs.env file after initial configuration to avoid compromising the configuration secrets.

Run the following command to start the Intel KBS as a Docker container:

```
sudo docker run --name relying-party -d --env-file kbs.env --
restart=always -p 9443:9443 relying-party:latest
```

Appendix D. AI Model Encryption Implementation

D.1. Prepare the Sample AI Workload Image

This section describes how to build and deploy the demonstration workload, which is an application that runs a machine learning (ML) model in an Intel TDX trust domain. Since the ML model is assumed to contain confidential data and parameters, it is encrypted when first deployed. The demo workload (i.e., attester) must obtain an Intel Trust Authority attestation token and include the token with a request to Intel KBS to retrieve the key needed to decrypt the ML model. This workflow implements a secure key-release use case in passport attestation mode.

Download the Dockerfile for building the Intel TDX demo workload.

```
wget https://raw.githubusercontent.com/intel/trustauthority-samples/main/deployment/sample-workload/Dockerfile
```

Run the following command to build the Docker image for the TDX demo workload:

```
sudo docker build --no-cache -t trustauthority-demo .
```

D.2. Provision the Relying Party and Encrypt the AI Model

Download the sample workload environment file.

```
wget https://raw.githubusercontent.com/intel/trustauthority-samples/main/deployment/sample-workload/workload.env
```

The workload.env file contains the settings needed for the sample workload to communicate with Intel KBS and Intel Trust Authority. This file contains secrets. Embedding secrets in an unencrypted file is not a recommended best practice for secure systems. It is only done here to make the demo easy to configure.

The KBS_ADMIN and KBS_PASSWORD must match the ADMIN_USERNAME and ADMIN_PASSWORD used in KBS.env.

The KBS_URL is of the form https://<IP_address>:9443/kbs/v1, where IP_address is the IP address of the host machine on which the relying party (KBS) container is running. If the KBS is running on the same TDVM as the workload, then provide 172.17.0.1 for the IP_address. Port 9443 is the default listening port for Intel KBS. If the port number for the Intel KBS container is changed, a corresponding change will need to be made to KBS_URL.

Setting SKIP_TLS_VERIFICATION to true will skip the TLS server certificate verification. For this demo, set it to true since a self-signed TLS certificates is being used for the KBS.

TRUSTAUTHORITY_API_KEY is the same as used in kbs.env.

```
KBS_ADMIN=<KBS-admin-username>
KBS_PASSWORD=<KBS-admin-password>
```

```
KBS_URL= https://172.17.0.1:9443/kbs/v1
SKIP_TLS_VERIFICATION=true
PLATFORM=azure
# HTTPS_PROXY=<uncomment and enter proxy information if necessary>
TRUSTAUTHORITY_API_URL=https://api.trustauthority.intel.com
TRUSTAUTHORITY_API_KEY=<API Key>
```

Run the following command to start the demo workload as a Docker container:

```
sudo docker run --name ita-demo -d --restart=always --env-file workload.env --device=/dev/tpmrm0 -v /tmp:/tmp -p 12780:12780 --group-add $(getent group tss | cut -d: -f3) trustauthority-demo:latest
```

The device option is used to set the Intel TDX device that the workload will use. The example provided uses /dev/tpmrm0, which is the device used by Microsoft Azure Confidential VMs. If a different Cloud provider or other hardware is used, set the device accordingly. Most non-Azure providers use /dev/tdx_guest.

```
sudo docker run --name ita-demo -d --env-file workload.env --device=/dev/tpmrm0 -p 12780:12780 --group-add $(getent group <user-group> | cut -d: -f3) trustauthority-demo:latest
```

Where <user-group> is the group owning the tdx_guest or tpmrm0 device:

```
crw-rw---- 1 <user> <user-group> 10, 123 /dev/tpmrm0
```

On a successful run, the container will execute the encrypt-model.sh script, which performs a number of provisioning steps to finish configuring the demo environment:

- Authenticate to the KBS.
- Create a new Key Transfer Policy, which dictates when the KBS will release the decryption key. In this demo, the policy requires Intel TDX and an updated attester (i.e., the demo workload running on the Azure Confidential VM) Trusted Compute base (TCB). Because this is a “background check” model attestation flow, the relying party (i.e., the KBS) is the enforcer of policy rather than the Intel Trust Authority (i.e., the verifier).
- Register the decryption key with the KBS and associate the wrapped key with the Key Transfer Policy.
- Encrypt the diabetes-linreg.model file.

The key transfer policy is:

```
{
"attestation_type": "TDX",
"tdx": {
```

```
"attributes": {
"enforce_tcb_upto_date": false
}
}
}
```

These steps prepare the workload and KBS for the attestation demo. The AI model is now encrypted, and the decryption key is held in wrapped form by the KBS. The KBS has a policy that restricts the release of the wrapped decryption key to requests that contain an attestation that Intel TDX is in use by the attester and that the attester's TCB is current.

The container then generates an answer file at /tmp/execute_workload_flow.env. This answer file contains the URLs needed to execute the demo flow as well as the KBS key ID generated during provisioning.

```
WORKLOAD_URL=https://127.0.0.1:12780
KBS_URL=https://172.17.0.1:9443/kbs/v1
KBS_KEY_ID=<key ID>
```

In a real-world production environment, a systems administrator would perform these provisioning steps before uploading the workload with the encrypted AI model (e.g., to their image registry, Cloud provider). For the purposes of this demo, the steps are scripted for simplicity.

Appendix E. Remote Attestation Implementation

E.1. Execute the Key-Release Workflow

Download the execute_workload_flow.sh script to execute the secure key-release workflow:

```
wget https://raw.githubusercontent.com/intel/trustauthority-samples/main/deployment/sample-workload/execute_workload_flow.sh
```

Run the following command to execute the workflow script:

```
bash execute_workload_flow.sh
```

The script will:

- Request an attestation token for the workload from the Intel Trust Authority using evidence collected from Intel TDX
- Request the model decryption key from the KBS using the attestation token
- Use the decryption key to decrypt the AI model
- Execute the AI model against sample patient data and output the results
- Clear the decrypted AI model from memory

```
✓ curl is installed
✓ jq is installed
============================================================
==========Using /tmp/execute_workload_flow.env environment file
Using Workload Host url : https://127.0.0.1:12780
Using KBS host url      : https://172.17.0.1:9443/kbs/v1
Using KBS Key id        : 2fa1511e-6606-42d9-adfd-854bb6b37f5a
============================================================
==========Get Attestation token from Intel Trust Authority:
Attesting TDX workload with Intel Trust Authority by submitting TDquote.
Url | https://127.0.0.1:12780/taa/v1/token
Method | GET
Expected Code | 200
Received Code | 200
   Response | {"attestation_token":"<Base64-encoded attestation token>"}
```

Decoded token

{<Decoded attestation token>}

Get decryption key from KBS: Requesting decryption key from KBS by sending attestation token obtained in previous step.

Url | https://127.0.0.1:12780/taa/v1/key

Method | POST

Expected Code | 200

Request | {

"attestation_token": "<Base64-encoded attestation token>",

"key_transfer_url": "https://172.17.0.1:9443/kbs/v1/keys/2fa1511e-6606-42d9-adfd-854bb6b37f5a/transfer"

}

Received Code | 200

Response | {"wrapped_key":"<wrapped decryption key>"}

Decrypt model: Decrypting Model using decryption key received from KBS.

Url | https://127.0.0.1:12780/taa/v1/decrypt

Method | POST

Expected Code | 204

Request | {"wrapped_key":"<wrapped key>","wrapped_swk":"<key encryption key>"}

Received Code | 204

Execute model: Executing model with sample test data.

Url | https://127.0.0.1:12780/taa/v1/execute

Method | POST

Expected Code | 200

```
    Request | {
  "pregnancies": "1",
  "blood-glucose": "128",
  "blood-pressure": "88",
  "skin-thickness": "39",
  "insulin": "110",
  "bmi": "36.5",
  "dbf": "1.057",
  "age": "37"
}
Received Code | 200
   Response | {"high-risk":1}
-------------------------------------------------------------------------------
Reset Model: Clearing decrypted model from memory.

       Url | https://127.0.0.1:12780/taa/v1/reset
    Method | POST
Expected Code | 204
Received Code | 204
-------------------------------------------------
```

Appendix F. List of Symbols, Abbreviations, and Acronyms

API
Application Programming Interface

CPU
Central Processing Unit

CSP
Cloud Service Provider

ID
Identifier

KBS
Key Broker Service

KMS
Key Management System

REST
Representational State Transfer

SSH
Secure Shell

SWK
Symmetric Wrapping Key

TEE
Trusted Execution Environment

TLS
Transport Layer Security

TPM
Trusted Platform Module

VM
Virtual Machine

VPN
Virtual Private Network