The Second Generation of IBM Hyper Protect Platform

A technical overview

Table of Contents

1. Motivation

Data is surrounding us in all aspects of our lives. We are collecting data to derive information in the pursuit of knowledge. Data can identify us and convey insights abouts us. Possession of data can be a differentiator for organisations, governments, and businesses. The value and risk that lies within the possession, access and use of data is illustrated by the rich set of classifications, protection, and regulations that exist. The construction of IT systems, that are storing and processing this valuable data are in many cases a reflection of the perspective towards data of the owners and operators of these systems. A constantly evolving and growing collection of data protection rules, policies, and regulations represent the common understanding and approach towards data.

As organizations and enterprises continue their journey towards digital transformation, the need for flexible IT investments that optimize their spending and are aligned to their strategies continues to become increasingly important. Organisations are reshaping IT operation strategies towards technology externalisation to service providers, who are providing as a service product covering the breadth of the hybrid cloud ecosystem from on-premise to Hyperscaler clouds. Externalizing parts of IT operation provides clear benefits in providing business enabling IT operations, innovation, and optimized technology delivery.

While there are clear benefits to this approach, it also poses a threat to data privacy and data protection for organisations. Balancing business benefit of externalised operations with the thread to data protection and data privacy can be a delicate act, that often not only impacts business owned intellectual capital, but also the data and privacy of people, clients, customers and/or consumers of services provided by these organisations. Data leakage and data privacy violations can be consequences resulting from these decisions. In this paper we are going to explain how technology can help to address this issue, entering a new era of data protection and data privacy.

The application of Confidential Computing technology is evolving to address the needs of this transforming IT operation landscape. Since its introduction, Confidential Computing technologies enable strong separation of process operation and data access. Trusted execution environments [1] protect data and the workload interacting with this data (while in use while the process is running within an operated system). The adoption of these technologies will become common as they address the development, and business value at a whole and as these technologies are integrated into established guidelines like Zero Trust or mandated regulations for a given organisation.

2. Introduction

This paper describes the approach taken by the Hyper Protect Platform in its second Generation to provide the bases for end-to-end secure environment for data at rest, data in motion and data in use. This platform offers privacy of code and data that spans the software and data lifecycles. While protecting data and code, the Hyper Protect Platform also supports a consistent developer experience that does not require special coding efforts in order to establish this level of protection.

The paper starts by outlining the *underlaying technology* provided by IBM Z® and how the requirements around Confidential Computing are considered when creating protected kernel virtual machines (KVM) in Linux® for the s390x architecture. KVM is an industry and community driven hypervisor and well-established technology on IBM zSystems and IBM® LinuxONE. The KVM enables scalability on a common virtualization and middleware stack. With Secure Execution for Linux on IBM Z, the protection of a workload against the underlaying hypervisor is introduced. Unique features like leveraging encrypted guest images and 3rd party attestation of the boot will be outlined.

Based on this technology this paper will introduce the *Hyper Protect Layer* which provides a hardened operating system with specific services, to provide data protection and technical assurance. Technical assurance is achieved when a chain of trust and set of encryption keys protect against the system or platform administrator and/or service provider. This form of assurance stands in contrast with operational assurance, a type of insurance offered by most service providers. Operational assurance ensures service providers *will not* access client workloads, whereas technical assurance ensures that service providers *cannot* access client workloads.

The importance of trustworthy platform software is considered in the context of which persona are involved, how the trusted code base is established and maintained throughout the software lifecycle.

This paper provides insight in how to *leverage the secure platform* of Hyper Protect. In addition example use cases are outlined: The advantages when working with Digital Assets and the ability leverage access to a Hardware Security Module (HSM) from the protected workload. Another premier use case is around Multiparty Computation and the ability to participate for a joint cause, while being ensured that no sensitive data is exposed. The generic use case of enterprise data protection as well as adoption towards Kubernetes based environments concludes the use cases outlined to provide end to end protection of keys, data, workloads, and authenticity.

3. Underlaying Technology – Secure Execution for Linux

The IBM Hyper Protect Platform in its first generation leverages the Secure Service Container technology [2]. This technology already maintains the confidentiality and integrity of hosted client data. It provides the support for enterprise DevSecOps solutions like the IBM Hyper Protect Secure Build or the simplified approach to industry and regulatory challenges, which technical assurance provides.
In its second generation, the IBM Hyper Protect Platform leverages IBM® Secure Execution for Linux (see [3] and [4] for details). This is a hardware-based security technology, which was introduced with the IBM z15® and IBM® LinuxONE III generation systems for Kernel Virtual Machines. It is designed to provide scalable isolation for individual workloads to help protect them from not only external attacks, but also insider threats. Secure Execution can help protect and isolate workloads on-premises, or on IBM Z and IBM® LinuxONE hybrid cloud environments.
To achieve this, the zSystem firmware contains a so called Ultravisor, a trusted firmware component, which enforces memory protection and offers the owner of a given KVM guest to securely pass secret information to the Ultravisor by using the public host key.

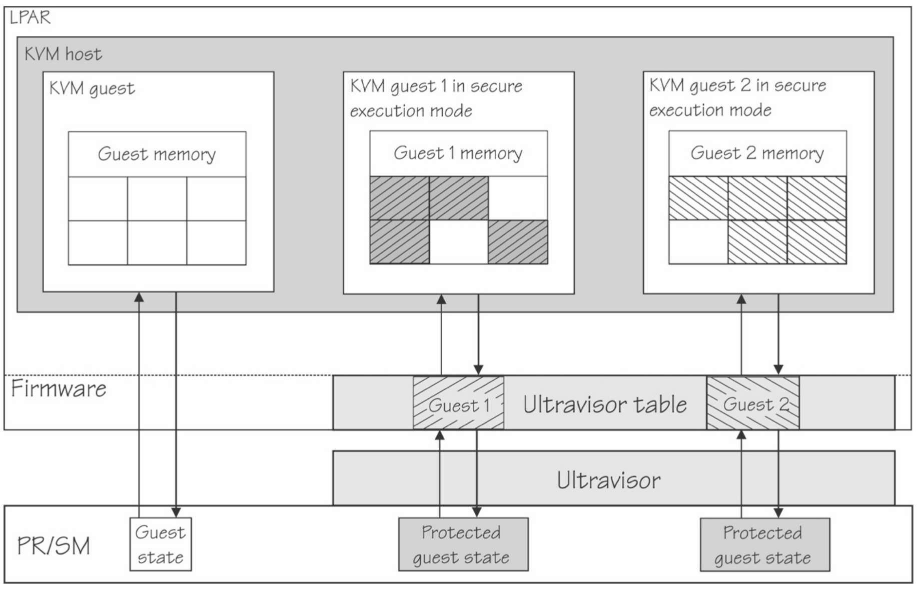


Figure 1 - taken from https://www.ibm.com/docs/en/linux-on-systems?topic=execution-components

To process the secret information, the Ultravisor uses the matching private host key. The private host key is specific to an IBM Z server and is hardware protected.

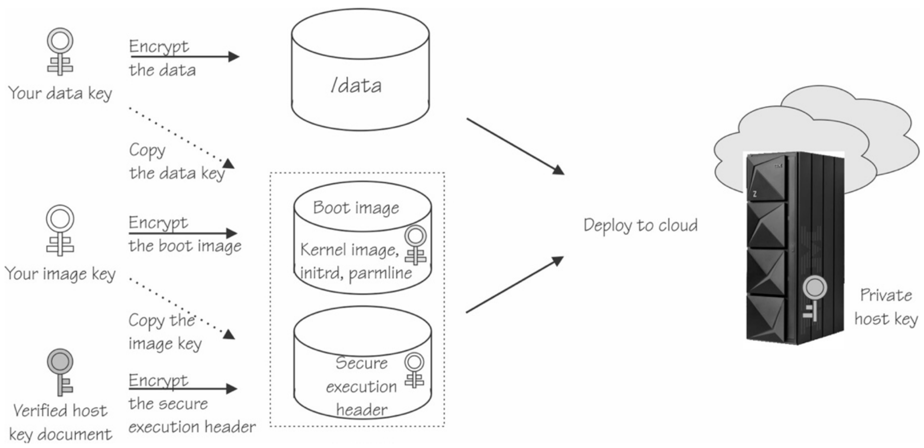


Figure 2 - taken from https://www.ibm.com/docs/en/linux-on-systems?topic=execution-secure-workload

Secure Execution for Linux is built to apply Zero Trust policies. Encryption combined with architectural, development and manufactural processes can establish technical assurance. Regulation and Audition require proof that the assumed environment and workload is present as well as appropriate personalisation of instances has happened. A common way to provide such proof are attestation methods or through an attestation record. In the latest generation of zSystems and IBM® LinuxONE the Ultravisor-powered attestation has been introduced, which can be leveraged independent of the concepts and platform described in this paper. Further information can be found in [5] and [6].

The protection of data security continues to gain market momentum and attention [7]. As countries and supranational unions tighten the requirements and regulation for data privacy and sovereignty, leveraging encryption for pseudonymization is accepted for security of processing – and privacy enhancing technologies [8] like confidential computing are acknowledged to address this need.

It is no surprise that in the context of Confidential Computing and Data Sovereignty various surveys and papers have been published recently (i.e. [9], [10], [11] and [12]). We also observe that such technologies and concepts are being used in production environments in datacenters and public cloud around the world. The next generation of Hyper Protect Platform has been architected to address this industry trend with regards to granularity, innovation velocity and protection boundaries and considerations.

Within the Hyper Protect platform, the Secure Execution technology is used to provide confidential computing capabilities by extending to a full end-to-end experience.

4. Hyper Protect Platform

The Hyper Protect Platform continues to evolve as it provides Confidential Computing capabilities for enterprises at-scale in Hybrid Cloud deployments. It remains the basis for IBM's Hyper Protect Services.

The already published whitepaper [13] about the IBM Hyper Protect Platform describes the first generation of the Hyper Protect Platform. The key principles of the platform are still valid in the second generation described in this paper. The Hyper Protect Platform provides a secure application workload runtime with technical isolation of the runtime from any person managing the environment. This platform includes not only the runtime itself. The end-to-end experience is extended by components to build trusted images within a customer owned environment, verification of the authenticity the to be started images and secure transportation of credentials within the trusted image itself.

The following figure shows the improved stack of the Hyper Protect platform. By reducing the size of the protected environment, it is possible to enhance data protection from potential attacks on the system from inside and outside. By the ability to protect data-in-use by individual KVMs it is possible to increase flexibility and scalability of the solution, which also becomes independent of underlaying software stack.

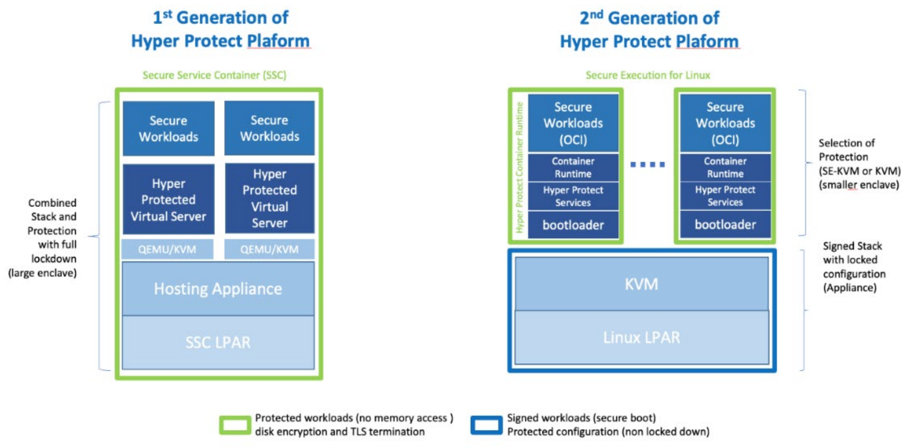

1st Generation of Hyper Protect Plaform
Secure Service Container (SSC)
Secure Workloads
Secure Workloads
Hyper Protected Virtual Server
Hyper Protected Virtual Server
QEMU/KVM
QEMU/KVM
Hosting Appliance
SSC LPAR
Combined Stack and Protection with full lockdown (large enclave)
2nd Generation of Hyper Protect Plaform
Secure Execution for Linux
Hyper Protect Container Runtime
Secure Workloads (OCI)
Container Runtime
Hyper Protect Services
bootloader
Secure Workloads (OCI)
Container Runtime
Hyper Protect Services
bootloader
KVM
Linux LPAR
Selection of Protection (SE-KVM or KVM) (smaller enclave)
Signed Stack with locked configuration (Appliance)
Protected workloads (no memory access) disk encryption and TLS termination
Signed workloads (secure boot) Protected configuration (non locked down)

Figure 3: The Hyper Protect Platform software stack

This chapter will show the different components which establish the Hyper Protect Platform. Components like the Hyper Protect Container Runtime (including the Bootloader and Hyper Protect Services used to validate the authenticity and trust of the workload). The Secure Build pipeline and Security considerations are discussed in detail to explain the principals of how trust is established within and towards the Hyper Protect Platform. We will outline the architectural concepts and introduce to you the different Personas who take part in the overall operations process of these workloads and continue with key enhancements provided as part of the platform described in this paper.

4.1. Architectural Concepts

4.1.1. Personas

When considering personas, we start with the view that the responsibility for defining the workload to be deployed is separate from control over how it gets deployed. And that there is no "assumed" trust that what is expected to be deployed is in fact what gets deployed.

Container Image Provider

A workload can consist of one or more container images which in combination deliver an expected solution. To ensure the integrity of the solution we must be able to verify the individual container images which deliver the solution, and ensure supply chain integrity through this.

All container images used are expected to be built in a secure manner that reassures the *Workload Provider* of their supply chain integrity and provides a way to verify the appropriate container images are being deployed. This means a signature is securely established over the generated container image and the signature provides a means for a *Workload Deployer Persona* to establish trust in the container image being deployed.

Workload Provider Persona

This may be the same persona as the *Container Image Provider*, but the workload may also combine container images from different sources. The workload provider persona defines the container(s) and environment requirements for the solution to be deployed. There is no trust given to allow any other persona to change the container images being used to provide the workload or to redefine the environment requirements specified by the workload provider.

The `workload` section of the *Contract* (explained later in the following chapter) is the key mechanism by which the *Workload Provider* defines the workload and environment expectations in a confidential and immutable manner. They do this ahead of deployment and provide this encrypted section to others for deployment to a specific environment.

Workload Deployer Persona

This persona is responsible for deploying the workload using the (cloud) infrastructure available.

The *Workload Provider* provides the encrypted workload contract section and identifies and communicates to the *Workload Deployer* the environment required to provide the workload to an end user. This allows the *Workload Deployer* to supplement the definition provided by the workload provider with instance/environment specific information (logging, storage etc)

The workload deployer is responsible for the service availability. They

- can control the networking, compute and storage resources made available to instance
- can influence network traffic in and out of a provided workload
- cannot change the workload to be deployed
- cannot change the *Workload Provider'* environment expectations.

If they do change the environment expectations, either the workload will not start, or the *Auditor* will not verify the environment when the workload is deployed.

Auditor

Since there is no "assumed" trust between the *Workload Provider* and other personas we must ensure at runtime that the workload is not deployed in a manner which breaks the *Workload Provider*'s or the *Workload Deployer*'s expectations.

The *Auditor* is the persona with responsibility for verifying that a deployed workload is both the workload expected and deployed to the expected environment. They do this by obtaining and verifying a trusted attestation record at runtime. The contents of this record can be verified against workload and environment expectations and details are covered in future section of this document.

Infrastructure/System Admin Persona

This role includes the system (cloud) administrator of a compute, storage, and network or support persona of the infrastructure like a Site Reliability Engineer (SRE). They will have responsibility for the underlying hardware but must not be able to use the capabilities this role offers to access confidential data or subvert the workload providers definition of the workload and expected environment. The use of Secure Execution achieves this through protecting the memory used by the workload from all external access at runtime, and the use of an encrypted Contract protects the intended workload and environment definitions at deployment time.

4.1.2. Contract

Least privilege and zero trust principles mean the *Workload Provider* and *Workload Deployer* personas identified in the previous chapter provides the workload and a corresponding deployment specific configuration and that this configuration can be independently provided and then combined without compromising integrity or confidentiality. We propose a so-called *Contract* is the mechanism to provide this configuration to be able to establish a confidential computing environment not only must we be able to control integrity and confidentiality of information between personas involved in creating and deploying the workload, but the *Contract* also ensures confidentiality from the infrastructure being used.

For the implementation in the Hyper Protect Platform we provide the *Hyper Protect Container Runtime (HPCR) image,* an encrypted Secure Execution image further introduced in a later chapter. The *Contract* is passed into a HPCR image as User Data via cloud-init process deployment. This means it is available to any Infrastructure persona before it is delivered to the HPCR image. To protect the *Contract* a public/private key pair is created as part of the *Hyper Protect Secure Build Pipeline* used to generate the *HPCR* image. This key pair is used to provide confidentiality for contract contents. The public X509 certificate of the *Contract Encryption* public key is published by IBM and can be validated by any persona out-of-band with the 3rd party certificate authority root key.

Each persona independently encrypts their contract section using this *Contract Encryption* public key. The *Contract Encryption* private key is randomized during image build and only existing inside the Secure Execution encrypted *HPCR* image. Such image can only be decrypted leveraging Secure Execution and keys rooted in hardware of a given zSystems or IBM® LinuxONE system. During deployment this key is used by the *Bootloader* to decrypt the *Contract* and the *Bootloader* ensures protection of this key from User space and the Workload.

The *Contract* enforcement within the *HPCR* image ensures

- W*orkload Deployer* cannot break the protection requirements of *Workload Provider* regarding use of the Secure Execution Enclave
- Only containers specified by the Workload Provider will run
- The *Workload Provider* and *Workload Deployer* can independently provide their section of the *Contract*, only sharing once encrypted
- Data volumes can be reattached to a new instance given the same relevant encryption config within the *Contract*.

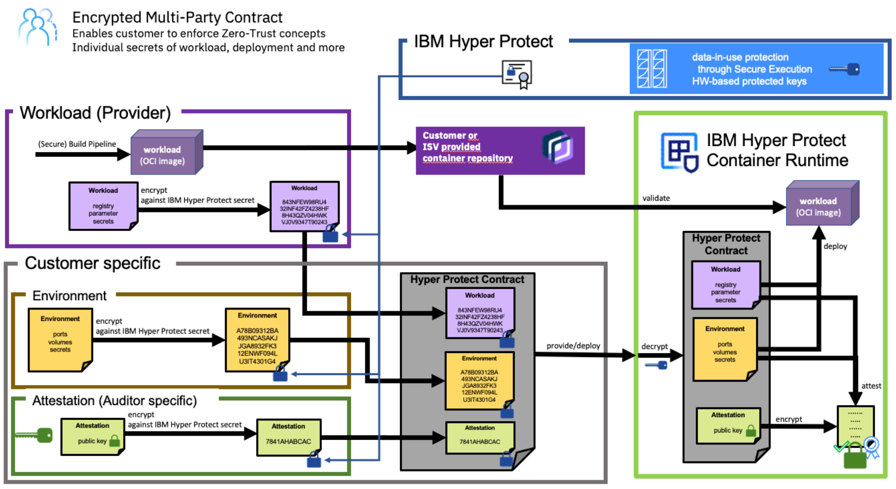
The *Contract* comprises four sections:

- `workload` is a mandatory section and contains information identifying and bringing up the workload
- `env` is a mandatory section and contains information defining the environment of the workload.
- `attestationPublicKey` is an optional section. The *Auditor* can provide a public key in this section, which is used to encrypt the attestation document.
- `envWorkloadSignature` is an optional section and contains the signature of the other sections of the contract.

This concept allows the *Workload Provider*, the *Workload Deployer* and the *Auditor* to cooperate while ensuring confidentiality and integrity of the respective information:

- The *Workload Provider* creates and encrypts the workload section and passes it securely to the *Workload Deployer*:

- The *Auditor* creates the `attestationPublicKey` and passes it securely to the *Workload Deployer*
- The *Workload Deployer* creates the `env` section for *Contract*
- The *Workload Deployer* combines the `env`, `workload` and `attestationPublicKey` sections, and calculates and adds the signature across the `env` and `workload` sections.
- The *Workload Deployer* provides the *Contract* within the cloud in its User Data section at provision time



Encrypted Multi-Party Contract
Enables customer to enforce Zero-Trust concepts
Individual secrets of workload, deployment and more
IBM Hyper Protect
data-in-use protection through Secure Execution HW-based protected keys
Workload (Provider)
(Secure) Build Pipeline
workload (OCI image)
Customer or ISV provided container repository
Workload registry parameter secrets
encrypt against IBM Hyper Protect secret
Workload 843NFEW98RU4 32INF42FZ4238HF 8H43QZV04HWK VJ0V9347T90243
IBM Hyper Protect Container Runtime
validate
workload (OCI image)
deploy
Customer specific
Environment
Environment ports volumes secrets
encrypt against IBM Hyper Protect secret
Environment A78B09312BA 493NCASAKJ JGA8932FK3 12ENWF094L U3IT4301G4
Attestation (Auditor specific)
Attestation public key
encrypt against IBM Hyper Protect secret
Attestation 7841AHABCAC
Hyper Protect Contract
Workload 843NFEW98RU4 32INF42FZ4238HF 8H43QZV04HWK VJ0V9347T90243
Environment A78B09312BA 493NCASAKJ JGA8932FK3 12ENWF094L U3IT4301G4
Attestation 7841AHABCAC
provide/deploy
decrypt
Hyper Protect Contract
Workload registry parameter secrets
Environment ports volumes secrets
Attestation public key
encrypt
attest

Figure 4: Flow of the Encrypted Multi-Party *Contract*

4.1.3. Data Volume Encryption

As the *Hyper Protect Container Runtime* image is successfully deployed, we refer to it as a *Hyper Protect Virtual Server* (HPVS) instance. Both the root disk and data disks in the HPVS instance are automatically encrypted with Linux Unified Key Setup (LUKS) Encryption. The root disk is re-created and encrypted on every boot with the original content provided with the HPCR image. A passphrase for the root disk is not stored.

The data disk encryption is configured by using the "seed" provided in the `workload` and `env` sections within the *Contract*. During the instance initiation, the disks are attached and encrypted by using the seed to create a LUKS passphrase. If the seed information or the data disk is not configured, the instance fails to initiate.

The disk encryption status check daemon checks the crypto headers of the root disk and data disks attached to the instance each hour, and then writes the information messages about the disk encryption status into the log.

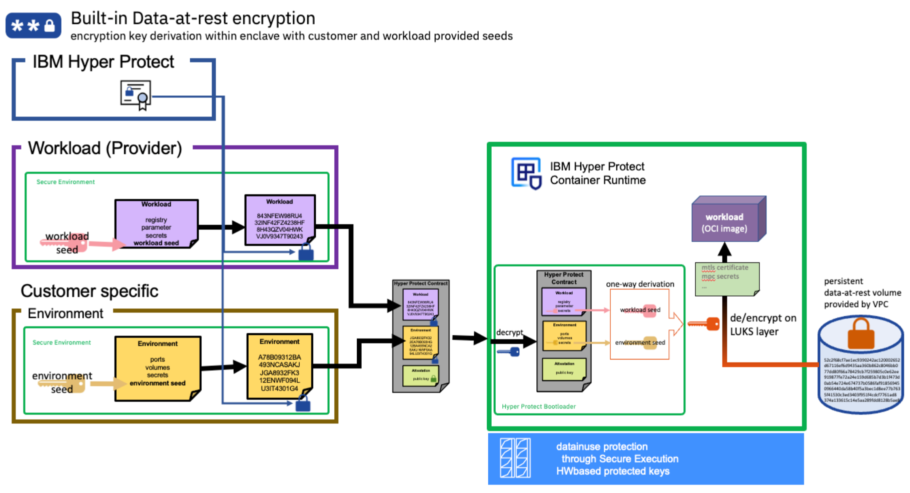

Built-in Data-at-rest encryption
encryption key derivation within enclave with customer and workload provided seeds
IBM Hyper Protect
Workload (Provider)
Secure Environment
workload seed
Workload
registry parameter secrets workload seed
Customer specific Environment
environment seed
Environment
ports volumes secrets environment seed
Hyper Protect Contract
IBM Hyper Protect Container Runtime
workload (OCI image)
mtls certificate
mpc secrets
decrypt
one-way derivation
workload seed
environment seed
de/encrypt on LUKS layer
Hyper Protect Bootloader
persistent data-at-rest volume provided by VPC
datainuse protection through Secure Execution HWbased protected keys

Figure 5: Data Volume Encryption by the *Hyper Protect Layer*

4.1.4. 3rd party Attestation of boot

The *Hyper Protect Virtual Server instance* provides an *Attestation Record* that is securely generated and signed by the *Bootloader* during the instance deployment. The signing key is published as a X509 certificate and can be validated out-of-band to a 3rd party certificate authority.

This *Attestation Record* is available to the workload only encrypted if the `attestationPublicKey` is provided in the *Contract*.

The *Workload Provider* can implement means for providing the attestation record to the *Auditor* persona. The *Auditor* then can verify out-of-band the environment in which the workload has been started.

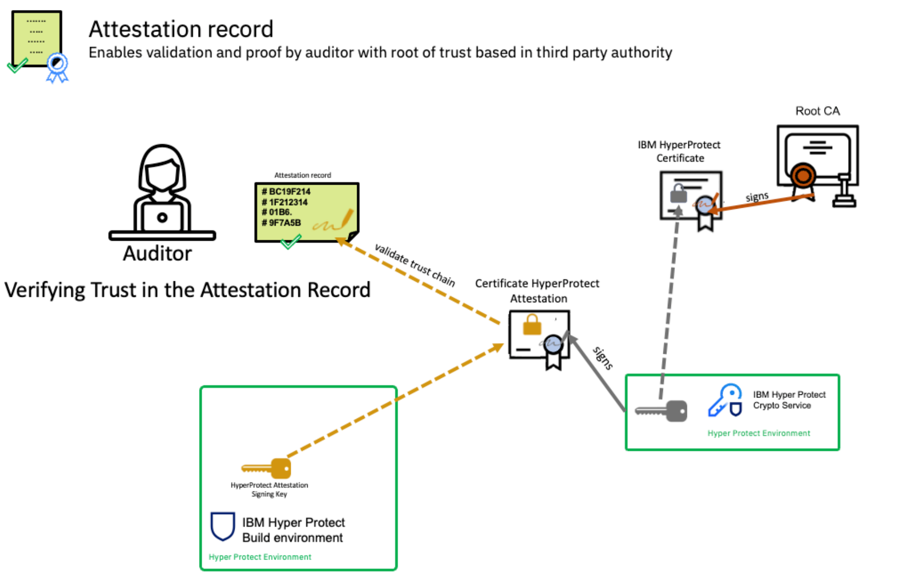

Attestation record
Enables validation and proof by auditor with root of trust based in third party authority
Auditor
Verifying Trust in the Attestation Record
Attestation record
# BC19F214
# 1F212314
# 01B6.
# 9F7A5B
validate trust chain
Certificate HyperProtect Attestation
signs
IBM HyperProtect Certificate
signs
Root CA
IBM Hyper Protect Crypto Service
Hyper Protect Environment
HyperProtect Attestation Signing Key
IBM Hyper Protect Build environment
Hyper Protect Environment

Figure 6: Verifying Trust in the Attestation Record

This diagram highlights the creation and management of the certificate hierarchy involved in signing the *Attestation Record*. The *Attestation Record* is signed by the *Hyper Protect Attestation Signing Key (HPASK)*. The HPASK can be confirmed by the published intermediate certificate. The intermediate certificate is signed by a 3rd party certificate authority, which is proven by the root certificate of that given certificate authority, thus completing the chain of trust.

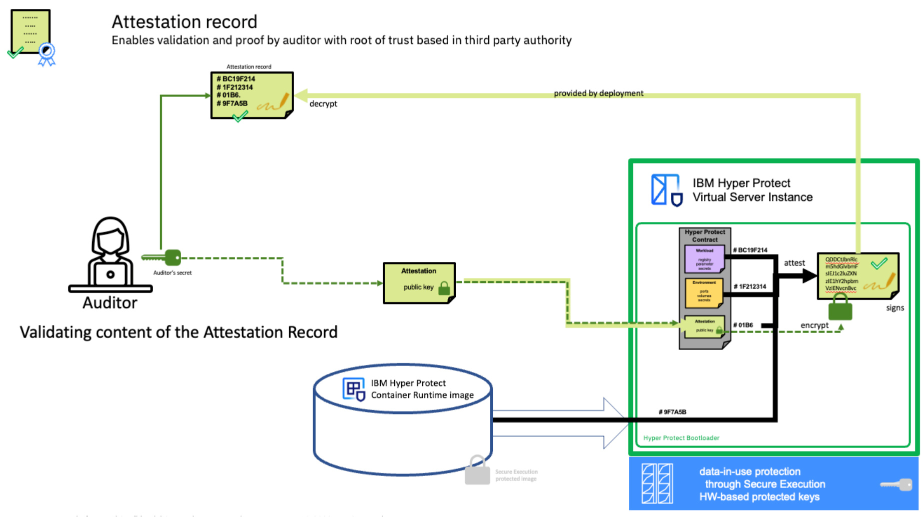


Figure 7: **Validating content of the Attestation Record**

An optional public key (*Contract* section: `attestationPublicKey`) for encrypting the *Attestation Record* at point of creation is provided by the *Auditor*, the private part of this is kept secret by the *Auditor*. The hash of this public key is added to the *Attestation Record*. By encrypting the *Attestation Record* and including the hash of the public key used for encryption within the *Attestation Record* only the *Auditor* is in the position to decrypt the attestation record and can validate that the workload that is deployed in the enclave is the expected and untampered version of the workload that is expected to be deployed

This *Attestation Record* contains measurements of what has been started and can be used to validate that the environment is the one deployed based on the following measurements:

- The original base image
- The compressed root filesystem to be stored into the LUKS encrypted partition
- The cloud initialization options incl. the *Contract*

The *Hyper Protect Attestation Signing Key* confirms with signature that the image got created in the Hyper Protect Secure Build pipeline.

The reference values for the measurements specific to the HPCR Image used are originated in the Hyper Protect Secure Build pipeline. Other measurements like the Cloud initialization options are dependent on the *Contract* or can be leverage by the workload to provide insight about individual instance identifier, which enables to *Auditor* to validate that the deployment is as expected.

4.2. Hyper Protect Container Runtime

In the following chapter we take a close look into the Hyper Protect Container Runtime image and its components and enhancements being provided.

As outlined previously the HPCR image is a Secure Execution image, which is by default encrypted and its decryption keys being protected by hardware.
The first code executed during deployment is the *Bootloader*, which will be cover next.
Next the *Hyper Protect Layer* is being introduced and its components.

The target usage in case of the HPCR are containerized workloads provided by a *Workload Provider* being deployed based on the *Contract* and the environment specifics being defined by the *Workload Deployer*. We will cover this in the *Workload and Environment deployment* section of this chapter.

4.2.1. Bootloader

The bootloader is the first part of the Linux start-up process that brings up a KVM virtual machine.

In Secure Execution the Linux Kernel and the temporary root filesystem (initramfs) are encrypted by a machine specific public RSA host key and stored on the QcCOW2 boot partition in a combined file. At boot the combined file is decrypted by the z15 Ultravisor into the virtual machine's memory which is inaccessible from the hypervisor. This ensures that secrets that are stored inside the initramfs such as private encryption keys are protected against the hypervisor environment and inaccessible from the virtual machine's operating system once the initial boot is done.

The bootloader provides its own implementation of a subset of cloud-init capability to reduce attack surface to a minimum.

The bootloader is responsible for

- Setting up the LUKS encrypted boot partition
- Writing the Operating System into the boot partition
- Decrypting the user provided contract and placing a decrypted copy in a known filesystem location, to be processed by Hyper Protect Layer components once boot completes. The bootloader destroys the private key after decryption. Writing a cryptographically signed attestation document of various checksums of important components

All steps are done at every boot. That means that the root disk filesystem is not persistent across reboots. All remote access such as SSH or login via serial console are disabled by default.

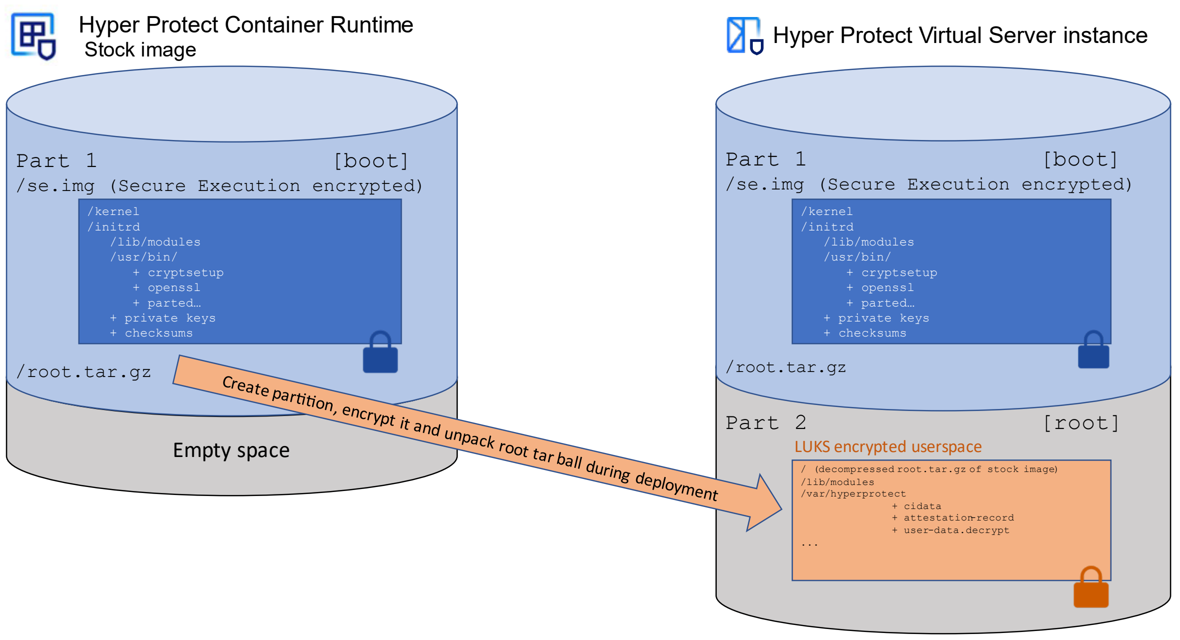


Figure 8: Bootloader of the Hyper Protect Container Runtime image

The *Bootloader* contains code which creates the attestation record, signs it using the attestation signing key. Only the Hyper Protect bootloader that is executed in the trusted execution environment provided through IBM Secure Execution for Linux on IBM® LinuxONE, can run the secure execution image, and therefore access the trusted attestation record signing key.

The `/var/hyperprotect` directory which contains the attestation record can be accessed by the workload container, the workload can then make this record available outside the Hyper Protect Virtual Server instance and ultimately to the *Auditor* to verify the record and therefore deployment has occurred into Hyper Protect Virtual Server before any data is introduced into the workload.

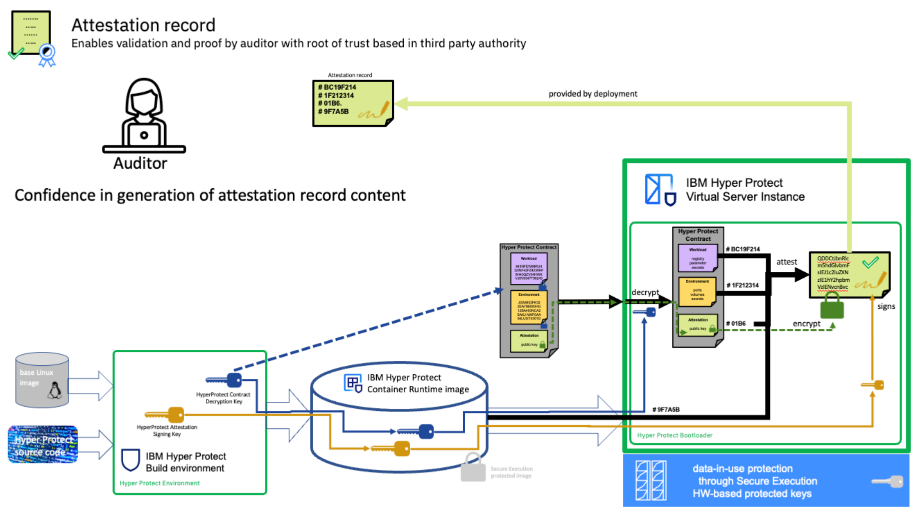


Figure 9: Confidence in the generation of the attestation record content

4.2.2. Hyper Protect Layer Services

In this section we will explore the code components provided by the Hyper Protect Layer.

All components share the following properties:

- a component is modelled as a systemd service. The dependency management of such services ensures that all components are executed in the right order and that the failure of one leads to the failure of its dependencies
- all components log to syslog including proper log levels that allow to identify a log message as informational, debug or error

All components are required to log to syslog as the central place for logs. They achieve this by configuring syslog as their logging target in the systemd configuration, the code of each component will send its log output to stderr.

Log output is customer facing and will be forwarded to the ingestion backend provided by the deployer. So all components make sure **not** to log any sensitive piece of information, such as PI data or credentials. The detail level of logging may differ from component to component, but the guiding principle is to provide enough information to identify problems without flooding the logs.

Errors are logged once, using the error log level. Each error features a unique identifier for the issuing component and the error situation. These identifiers are part of the API of HPCR, so they may be used in automation components to react on error situations. Identifiers will be kept stable across semantic releases.

These are component services of the Hyper Protect Layer:

Logging Service
The logging component is responsible for the setup of the logging configuration. This configuration is available in the contract and allows a deployer to configure a Mezmo logging backend or a custom backend compatible with the rsyslog protocol.
The logging component validates the configuration and transforms it into a configuration for the rsyslog component that is used as a log forwarder.

If the logging configuration is invalid this will be signalled on the serial console and the start of the virtual server instance will fail.

Contract Validation Service
The contract validation component validates the contract syntactically and semantically against a JSON schema. If validation fails so will the start up of the virtual server instance.

Registry Authentication Service
The registry authentication component assembles all credentials in the contract required to authenticate against a remote docker registry and transforms them into the configuration format required by the OCI runtime.

Image Service

The image service assembles all information in the contract related to the validation of OCI images, e.g. docker content trust or RedHat signatures and converts them into the format required by the OCI runtime.

Signature Service

The signature service verifies the optional signature that binds the workload section to an environment section.

Storage Service

The storage service initializes attached storage volumes according to their configuration in the contract. It creates necessary partitions, encrypts them via LUKS2 encryption or opens an already existing LUKS2 encrypted layer. After the successful execution of the storage service, storage is mounted to the filesystem ready to be consumed by OCI images.

Luks Passphrase Service

The LUKS passphrase service computes the passphrase used for LUKS2 encryption of the storage volumes.

Container Service

The container service is responsible for starting the OCI images using the OCI runtime of choice. After the successful execution of this service, the configured containers are running.

Catch Service

The catch service monitors the successful start of the other services. In case of failure, it will automatically shut down the VSI after a grace period.

4.2.3. Data Volume Encryption Services

Data-at-rest protection is key to ensure an end-to-end protection profile. Several services and tools provided as part of the HPCR image jointly ensure the protection of the attached data volume and enables a HPVS instance to have data volumes being protected by a unique LUKS passphrase which is a derivation per environment and workload. This derivation enables a reattachment of a given volume assuming the same seeds are provided within the contract. Neither the *Workload Provider Persona* nor *Workload Deployer Persona* can independently access the attached persistent storage to the deployed instance. In the initial revision of the HPCR image only one volume attachment is supported. Only the first volume provided in the Contract is used to complete attachment and must match the required storage by the workload. The mounted volume has a fixed mountpoint specified in the contract. The workload should create and use subdirectories under the mount point.

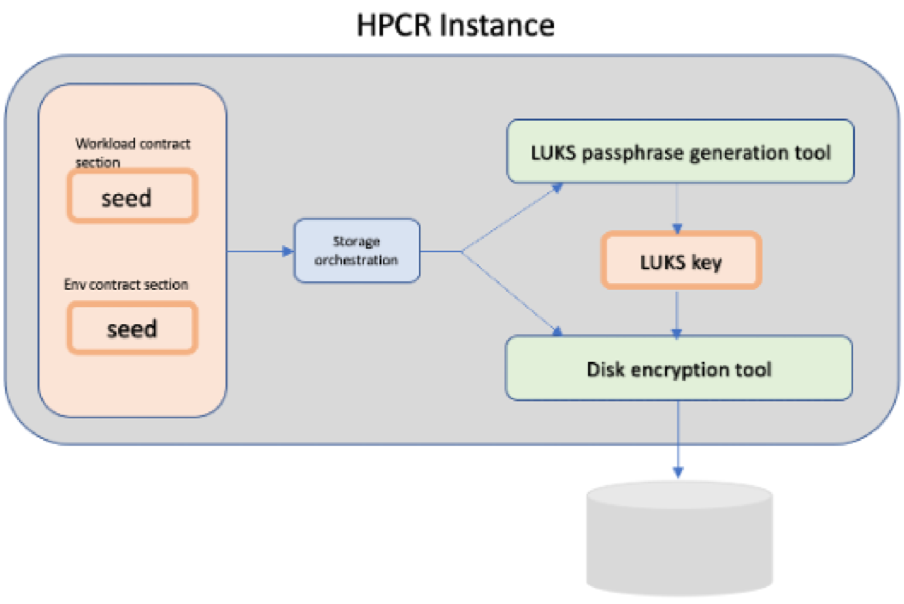


Figure 9: Flow to create passphrase for persistent data volume by the Hyper Protect Layer

The volume encryption is done automatically by storage service. Storage service internally uses two command line tools *se-lukspassphrase-gen* and *se-disk-encryption*. The volume encryption requires seed information in the *Contract* under both env and workload sections. The HPVS instantiation will fail if the seed information is missing in the *Contract*. These two seeds provided as input are internally converted to UTF-8 sequences and then concatenated. Later, the hash (SHA256) of the concatenated sequence is computed as a hex digest, which is used as the LUKS passphrase for the given data volume.

These tools are available for workloads to leverage further.

se-lukspassphrase-gen

Command line tool will accept seeds and return a LUKS passphrase for disk encryption.

- One or more seeds may be presented in the contract
- Given same seeds will generate the same LUKS passphrase

se-disk-encryption

Command line tool which will create a LUKS layer on the device using the provided passphrase and make the resulting filesystem available at a given mount point by LUKS open.

- The tool can be run simultaneously but only using different source devices
- The tool will detect a resized volume and expand the filesystem on VSI boot.

4.2.4. Workload Deployment and Considerations

The rise of containers and the ability to separate a workload from the underlaying software stack with this architectural concept helped to accelerate the development of software, enabled microservices and reuse of (software) components or services and reduced the deployments to deploy complex dependent software components and keeping them running and update to date. The workloads for the Hyper Protect Platform are being provided as a Bring-Your-Own-Image based on the Open-Container Initiative (OCI) image definition. This allows usage of any open-container initiative (OCI) image and gain the benefits of a confidential computing solution for additional levels of protection while reducing the trusted code base to the workload stack. The *Workload Deployer* can ensure through a signed *Contract,* that only a given combination of workload and environment are deployed. The container images defined in the workload section are pulled into the deployed instance through the mentioned *Container Services*. After additional checks are performed – like whether container image is trustworthy based on the *Contract* – the container image is being started in a container runtime provided within the *Hyper Protect Container Runtime* image.

Workload Expectations

Multiple independent workloads should not be started within the same instance of HPVS. Each instance should be a single unified workload or microservice which may be composed of multiple containers. The containers specified in a *Contract* are assumed to be mutually dependant if deployment of one container fails the workload stops.

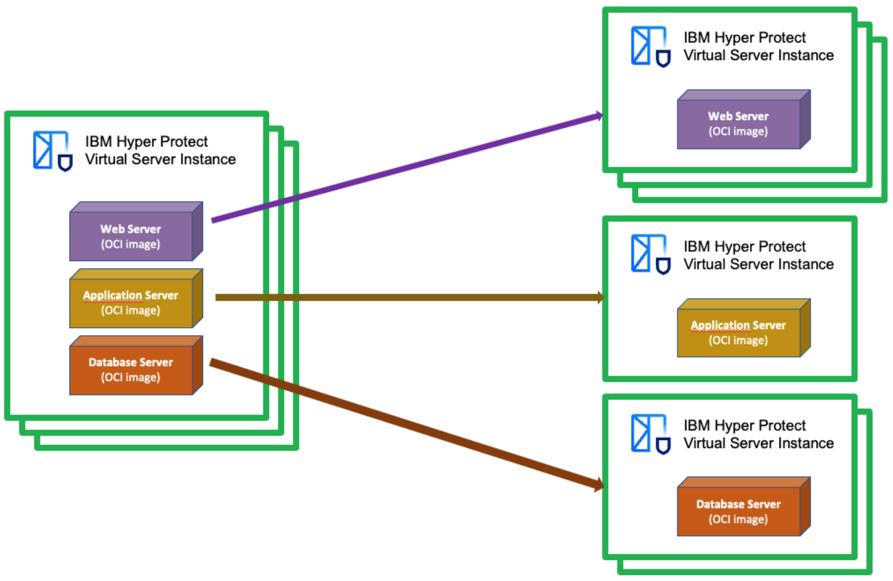

IBM Hyper Protect Virtual Server Instance
Web Server (OCI image)
Application Server (OCI image)
Database Server (OCI image)
IBM Hyper Protect Virtual Server Instance
Web Server (OCI image)
IBM Hyper Protect Virtual Server Instance
Application Server (OCI image)
IBM Hyper Protect Virtual Server Instance
Database Server (OCI image)

Figure 10: Different workloads to be deployed within HPVS

Decomposing a solution (e.g., microservices) to establish loose coupling is encouraged and in support of a cloud native direction. Multiple HPVS can be used for each microservice with the infrastructure (like the IBM Cloud® VPC offering) providing the necessary capabilities to secure networking communication between them. Even in the absence of cloud native orchestration such as Kubernetes, important benefits of loose coupling are

- Independent scaling of services
- Independent upgrade/update of services

as this help to mitigate the "Cattle" nature of HPVS.

Cattle not Pets

The HPVS instances are not intended to be cared for, destroying them, and recreating them is an expected practice during the lifetime of a workload. We go further to ensure that any changes to the root volume are not persisted after reboot.

This means:

- Contract is set at deployment and cannot be changed
- HPVS instances are not upgradeable
- snapshots of HPVS attached boot volumes make no sense
- Updated OCI container images require a new *Contract* and a new HPVS deployment
- Updated API keys/passwords etc. require a new *Contract* and new HPVS deployment
- Updated HPCR Image requires new HPVS instance, but previous *Contract* can be reused
- Reboot will repull container images (root volume changes are not persisted).
  This means that contract details relating to pulling images must remain valid through all HPVS reboots after creation.
- No workload data should be held on root/boot disk, this will get destroyed for any reboot and any update.
- Externally attached data disks will have an independent life cycle and can be reattached to replacement HPVS assuming contract sections relating to storage remain consistent.

Tooling such as Terraform can be used to make generating new contracts and replacing HPVS instances straightforward.

5. Consideration of trust

5.1. Leveraging Secure Execution for the Hyper Protect Platform

While host specific keys of Secure Execution for Linux offer great protection and the ability to bound the deployment to a given system, it comes with the burden to workload providers, that no generic workload can be built, which can be personalized to a given system or customer.

To address this, the zSystems and IBM® LinuxONE server have a second private keyset for the Hyper Protect Platform. This keyset is present to the Ultravisor in any enabled system and is global or common. This global key enables to leverage the Secure Execution technology without the limitation to a specific host or the need to predefine the set of hosts a workload can run on.

The second generation of the Hyper Protect Platform leverages the global (Hyper Protect) key to build its workloads against. By doing so the Hyper Protect Platform establishes a secure, encrypted and technical assured passage from our *Hyper Protect Build Pipeline* to the Ultravisors on any enabled system to provide the workload as well as embedded secrets which enables zero knowledge proofs: The *Hyper Protect Container Runtime* image.

As the Secure Execution Header and the Boot image are both encrypted it is ensured that these KVM QCOW2 image leaving our Hyper Protect Build pipeline can only be decrypted by the Ultravisor and the trusted firmware of the given zSystems or IBM® LinuxONE system.

As we have full control of the bootstrap of that deployment through the *Bootloader*, we can provide additional values and features that simplifies the usage of Confidential Computing even further. This has implication on the trust to be established by the build pipeline used, which we will outline now in more detail.

5.2. Hyper Protect Build Environment

The purpose of the secure build environment is to produce Secure Execution images such that the secrets required to execute this build are protected at and time and are not accessible to a single person. We achieve this by executing the build inside a Secure Execution protected build virtual machine to start with.

Hyper Protect Secure Build Virtual Machine

The Hyper Protect Secure Build Virtual Machine (*BuildVM*) is as mentioned a virtual machine running in secure execution mode. This implies that all data inside its memory or its disk image are protected at all times. Also, this virtual machine does not expose any services (such as SSH access or REST entry points) with the goal to reduce the potential attack vector to a minimum.

The *BuildVM* encapsulates all secrets required for its operation as part of its own image, i.e.

- the certificate used to sign the contract encryption certificate and the attestation key
- access credentials to Hyper Protect Cryto Service (HPCS) used to perform encryption and signing operations
- the key blobs wrapped by the HPCS protected master key

The purpose of the *BuildVM* is to build new secure execution enabled virtual machines and to inject credentials into these machines. Since these credentials are scoped to the *BuildVM* and not know to anyone outside of the *BuildVM*, they never leave the realm of Secure Execution. There exist the following modes of operation for the *BuildVM*:

- Build a new generation of itself. In this mode the *BuildVM* has the opportunity to update code components, its own secrets or both. While updates to code components are passed in as configuration of the build step, the update to secrets is implemented by the *BuildVM* itself, it is e.g. able to automatically update the API key used to access HPCS, so neither the new or the old API key leave the realm of secure execution
- Build a new version of an image that will be use by customers (aka. customer image). In this case the *BuildVM* will e.g. inject the private key used for contract encryption into the customer image.
- Provide signed Certifcate Revocation List of revoked certificates - if any - issued by the *BuildVM* e.g. Contract Encryption and Attestation Signing certificates

Bootstapping

Secrets inside the *BuildVM* are always protected because they are stored inside its image and all operations, including renewal or revocation are executed by the *BuildVM* itself. However there needs to be an initial step – the bootstrapping step – that produces the very version iteration of a *BuildVM*. This step involves the coordinated collaboration of different personas where each persona or role contributes a fraction of the total set of secrets. During an audited ceremony these personas come together and provide their pieces to the bootstrap to a precursor of the *BuildVM*. This *BootstrapVM* is a Secure Execution enabled virtual machine that exposes an SSH entry point for the limited period of the ceremony and restricted to a known set of personas, which identify themself to the *BuildVM* through a public key. Each member of the ceremony uses a dedicated entry point to provide their secrets, the setup ensures strict separation of duties and makes sure that secrets provided by other personas are not accessible.

After all secrets have been provided, the *BootstrapVM* rolls the provided API keys and builds the first iteration of the *BuildVM* with these rolled secrets. As a consequence, secrets stored inside the *BuildVM* have never been accessible to a persona.

Configuration

While all secrets are stored and managed by the *BuildVM*, an update to code components requires external configuration. Since such code will be executed inside the *BuildVM* with access to secrets, the content and quality of these components must be ensured. This is done by the following means:

- all code is packaged in installable code bundles (Debian packages at the time of this writing). These bundles are signed by their build process and the receiving *BuildVM* will validate the signature before accepting any bundle.
- the build process for code bundles is a managed build that is configured such that it will output signed bundles only for code on the main branch of managed github repositories
- these repositories in turn are configured such that all code that will be merged to the main branch has to be approved by at least two personas after a diligent code review

There exists a piece of configuration sent to the *BuildVM* at its start-up. Since the *BuildVM* does not expose any service entry points, this configuration is mounted into the *BuildVM* as a read-only block device. The content of this block device is a signed document containing references to updated code components and information about the kind of build to be executed (e.g. build a new *BuildVM* or a CustomerVM). The *BuildVM* will validate the signature before accepting any configuration.
The signer of the configuration is a managed Jenkins build. It pulls the configuration document from a github repository (setup with a strict approval process) and uses a Jenkins secret to execute the signature.

Summary

- secrets are stored and managed by a Secure Execution enabled *BuildVM*, never leaving the boundary of the Secure Execution protected KVM
- code and configuration are managed inside github, protected by a strict approval process for modifications
- the authenticity of code is proven by bundle signatures issued by the build pipeline

builds are triggered via Jenkins jobs and the authenticity of the build input is proven by a signature based on Jenkins credentials

5.3.Contract

There will be secrets in all sections of the *Contract,* however by choosing the workload to use the deployer trusts the workload provider with their secrets. They trust that the secure enclave and the workload OCI containers will not to leak the environment secrets. This trust could have been established by auditing the exact versions of workload to be used or by contract with the workload provider.

The workload provider may have secrets they do not wish to share but they trust that the Secure Execution protected KVM will not log secrets from the *Contract* (assuming the workload provider defined the values as secrets within the contract and not as environment variables). The workload provider also controls the vulnerabilities in their workload though they may wish to ensure the deployer keeps their workload up to date. Though the deployer if they are owner of data involved will have more to lose by not keeping the workload at the most recent revision.

The *Contract* needs to support the level of trust expected between the personas identified. To support these protection requirements and to protect the data used within the virtual server no remote access should be provided to a given instance by the workload. The HPVS instance does not provide any SSH or console login. We acknowledge that container breakout is possible and so no guarantee can be made on protecting this config once within the HPVS instance. However, any decision to provide accessibility should be a architected part of the workload and provided by privileges or logic with the workload OCI containers and workload definition.

Contract Usage Principles

Guiding principle for enclave contract contents

- Keep any secrets as specific to the workload as possible (Do not reuse principle)
- Any credentials provided should have least privileges in case of container breakout
- Use OCI container signatures to ensure only expected containers start

Contract Security Considerations

Keys personas involved in the Infrastructure can intercept the *Contract* as the HPVS instance or Secure Execution protected KVM in general is being deployed. They would be unable to alter or read the *Contract* contents within each section but they could substitute an entire section of cloud-init user data given the key to encrypt sections is public.

Replacement of workload section

This would allow a malicious actor to substitute the intended workload for one of their own, which would give access to any secrets held within the environment.

The use of a signature across the `workload` and `env` sections with the public key to verify the signature contained within the `env` section ensures the environment cannot be used with a different workload since the signature would then be invalid. The private key used to generate the signature is not within the contract and known only the true *Workload Deployer* persona, who created the `env` section.

Replacement of environment section

In general, the `workload` section is designed to work with any `env` section so the workload itself would not be concerned with `env` section substitution. The *Workload Deployer* must ensure they obtain the `workload` section through secure channel, this prevents the deployer combining their `env` section with an unintended `workload` section, which could expose env secrets to the unintended workload.

Data Volumes

Replacing the `env` section without already knowing the data volume seeds will not grant access to any pre-existing data.

Replace environment section to fool the end user.

It is assumed that any workload should allow for the provision of a private secret (such as SSL server certificate) within the `env` section. This means any end user will not be fooled by a workload started with a replaced environment section as it would not contain the expected private secret. Such a private secret would be considered environment specific.

Replace environment section to fool a connected service.

It is assumed that any connected services would require private credentials to access them (possibly login details or SSL client certificate). Should these credentials be provided within the environment section then any replacement of this section will not grant access to the connected service.

Logging Service

Given two points above impersonation through replacing the `env` section of the *Contract* will not result in any use of the workload. As such no logs will therefore be generated. However

- if the `workload` section contained secrets that were used at start up of the workload.
- And these were logged then access would be gained to these secrets

We assume that any workload secrets are explicitly defined as secrets for orchestration of the workload, which in turn means they will not ever be logged and so never revealed to any logging service.

Behaviour Considerations for contracts

The `env` section is defined with a specific workload in mind and so a signature across the combined encrypted `workload` and `env` section is provided with the *Contract*. The `env` section will contain the public signing key to be used to verify the signature ensuring the `env` section can only be used with the intended `workload` section.
HPVS instances are considered cattle not pets and *Contracts* for deployed instances are immutable - this means the instance will be redeployed with the same *Contracts* in response to updates to the HPCR image or with new *Contracts* in response to workload or environment updates/changes.

By using a signature (`envWorkloadSignature`) across the `workload` and `env` section rather than just hashes of the `workload` section it offers the opportunity to avoid the need to recreate the `env` section in response to an updated `workload` section of the *Contract*. The signing public key can remain the same within the `env` section and a new signature simply needs calculated and updated with the cloud-init user data provided. This does require careful management by the *Workload Deployer* of the signing key but the *Workload Deployer* has the choice and can still recreate the `env` section each time with a new fresh signing key if desired. In this scenario the private part of the signing key can be thrown away once used and the signing key pair regenerated each time, but it does mean all content including secrets of the `env` section would need to retrieved each time to generate a new `env` section to the contract.

Attestation of Deployment

Cloud-init must be secured to prevent injection into the deployed HPVS instance, only user data relevant to the contract will be processed and no scripts will be supported. The contract cannot be tampered with due to integrity (signatures) and confidentiality (encryption) protection in place and attestation will be the mechanism to prevent substitution of the contract with another. The cloud-init user data is part of the attestation record enabling verification that the contract used to create the Hyper Protect Virtual Server instance is as expected.

5.4. Secure Build for Hyper Protect Container Runtime

As outlined previously the integrity of the supply chain remains a key consideration to ensure that the container images are built securely. It is best practice to sign container images or corresponding meta data to ensure, that the deployed images are the one created in a trustworthy environment by the authorized Workload Provider. It is recommended to consider a protected Virtual Machine like a HPVS to perform such security relevant operation. The Hyper Protect Secure Build is capable to provide a secure container build environment, which protects the important signing key. For further details how to protect your image build see [14].

A securely build image leveraging this Hyper Protect Secure Build on IBM Cloud Hyper Protect Virtual Servers can be provisioned by the HPCR image as a HPVS instance. The *Contract* is created by the *Workload Provider* based on the Manifest file provided by the build pipeline or *Container Image Provider*. This way either be the public key used to sign the container image or the digest of the container image is extracted and provided into the `workload` section of the *Contract*. After the definition of the `env` section by the *Workload Deployer* you are set to leverage the Secure Platform of Hyper Protect.

6. Leverage the Secure Platform

As outlined in the previous chapters the Hyper Protect Virtual Servers based on the Hyper Protect Platform offers highest level of protection for the data and offers flexibility of deployments either on IBM Cloud or on-premises.

6.1. IBM Hyper Protect Virtual Servers for zSystems and IBM® LinuxONE

Hyper Protect Virtual Servers can be deployed On-premises – a private cloud deployment option that allows Hyper Protect Virtual Servers to be deployed on Linux LPAR (Logical Partition) running KVM enabled with Secure Execution. This option is suitable for customers having zSystems/IBM® LinuxONE and want to secure their workload deployments.

The *Infrastructure/System admin* has to ensure the system being ordered or upgraded to have Feature Code 115 present and the key bundles are imported based on the instructions available for the Secure Execution enablement as a Cloud provider [15].

Hyper Protect Virtual Server instances with the Hyper Protect Container Runtime image can be brought up with Linux on IBM Z in on-premise environments. Please find more details at https://www.ibm.com/docs/en/hpvs/2.1.x?topic=about-this-documentation

6.2. IBM Cloud Virtual Private Cloud (VPC)

IBM Cloud Virtual Private Cloud – a public cloud deployment option enabled by IBM Cloud. In this option, the workload can be securely deployed as containers running on Virtual Server Instance (VSI) protected by secure execution. This option is suitable for workloads that can access other Hyper Protect Services such as Hyper Protect Crypto Services or other IBM Cloud services such as IBM Cloud Databases. IBM Cloud IAM (Identity and Access Management) is used to control the access to the environment and billing can be enabled on hourly basis.

Hyper Protect Virtual Server for VPC (with Secure Execution) can be provisioned from the IBM Cloud portal (Virtual Private Infrastructure) by a registered IBM cloud subscriber. Review IBM Cloud VPC documentation [16] for general information and insight.

The IBM Cloud command line interface (ibmcloud) can be used to create an instance of Hyper Protect Virtual Server for VPC by selecting the “Hyper Protect Container Image Runtime” stock image and the appropriate Secure Execution Profile. More details on how to use IBM Cloud CLI can be found at [17].
The Hyper Protect Container Runtime stock image associated with the “Hyper Protect” operating system is periodically updated to provide security fixes and new functionality.
To be able to leverage the updated stock image (and any updates to the BYOI container images), bring up a new instance of HPVS for VPC with the same contract that was used previously (updating the latest version or digest of container images wherever applicable) with the same data volume that was used with the previous HPVS instance. Please note that the contract must have the same volume seeds that was passed in previously to be able to access the data volume.
Further documentation how to deploy a HPCR image with a *Contract* to create a HPVS instance can be found here: https://cloud.ibm.com/docs/vpc?topic=vpc-about-se

Given the workload expectations and the treatment of the HPVS instances as cattle and not pets, there are certain common capabilities of infrastructure middleware that serve very little purposes:

ssh keys

Providing an SSH key when creating the VSI makes no sense. The VSI is a locked one, there is no ssh capability and any ssh key provided by cloud init will be ignored.

Snapshot

On each reboot the VSI will revert to the original HPCR image content and reapply the contract. The cloud-init User Data (and hence the *Contract*) is not part of the snapshot, so any VSI created from the snapshot would receive new cloud-init User Data which may or may not be the same as the original *Contract*. This new VSI created from the snapshot simply applies the User Data it is given at creation time. So, behaviour is controlled by the *Contract* supplied to the HPCR image during the VSI creation and snapshot will not preserve anything about the previous application or VSI creation. A snapshot is essentially the "equivalent" of the original HPCR stock image.

It should be noted that while it is possible to mount a HPCR boot volume or snapshot as an additional disk to an existing VSI. This is useless because the LUKS passphrase is not known.

Image from a Volume

The VSI is intended to be an immutable thing once created; a different *Contract* cannot be provided to any HPCR image that maybe be created. Essentially it would be an image which would not accept any new cloud-init (*Contract*) and would only ever fulfil the *Contract* from the original HPVS.

7. Use cases of the Hyper Protect Platform

In general, any workload that deals with sensitive data or have the need to support compliance and regulatory requirements is very suitable to be deployed in Hyper Protect Virtual Servers environment. Particularly,

- Securing Digital assets – such as cryptocurrencies, non-fungible tokens (NFTs), blockchain distributed ledgers, smart contracts. For digital asset custodies, exchanges that must protect private keys, applications, and data, Hyper Protect provides a secure hosting environment that provides end-to-end security and has been chosen by key ISVs in this market to provide holistic solutions for Digital Assets.
- Secure Data Processing – Processing sensitive data such as financial data, healthcare data, data containing PII
- Secure Multi Party Compute (MPC) - Secure multi-party computation is to enable parties to jointly compute a function over their inputs, while at the same time keeping these inputs private. MPC provides solutions to various real-life problems such as distributed voting, private bidding and auctions, private information retrieval
- Secure Database Hosting – Hosting databases such as RDBMS (such as postgres SQL), noSQL databases (such as MongoDB)

7.1. Digital Assets

Digital Asset solution providers are at the forefront of adopting data protection and data privacy solutions. The digital asset domain is relatively new and just got started to be guided by regulations, the nature of the underlying technology defined by tokenization, cryptography and policy has enabled providers in this space to differentiate by data protection measures provided to their consumers.

Users who are not willing to undertake the effort of self-custody can select from set of custody providers (marketplaces) for hosting their wallets.

Custody solutions maintaining wallet application within a TEE can provide guarantee to their users that their data (temporary or not), cryptographic keys, monitoring and activity logging and its transaction execution policies cannot be circumvented by attacks taking advantage of operating system or hypervisor vulnerabilities or even stolen infrastructure administrative credentials. With the use of TEEs privileged users can be prevented from bypassing application security at the operating system or at the virtualization hypervisor and even at raw hardware level. This protection does not only need to consider runtime protection, but also secure the software supply chain. Bridging runtime and software supply is possible through TEE attestation mechanisms, which validates the application signature and verify it has not been tampered between the release time and the go live.

The IBM Hyper Protect Platform builds the all-in-one solution to deploy and manage such applications using TEE protection on zSystems and IBM® LinuxONE including connections to Hardware Security Modules (HSMs) providing physical protection for persisted cryptographic key material. Applications perform secure cryptographic operations using GREP11 services. (GREP11 API is similar to EP 11 API but stateless and executed over the gRPCs protocol. GREP11 API documentation can be found on [18])

Protecting the master seed with a HSM is a de facto standard to protect digital Hierarchical Deterministic wallets. Hierarchical deterministic key generation can derive an infinite number of cryptographic secrets from a single master seed using a derivation path as input. IBM Crypto Express HSM stores an unextractable wrapping key that is used to protect application cryptographic material. This key is loaded via a key ceremony and held by custodians. Wallet applications only store secure keys protected with the HSM's wrapping key. IBM Crypto Express adapters transparently process cryptographic operations using these secure keys as input, by unwrapping keys and performing the requested operations inside the HSM without running the risk of disclosing their value outside. This way, it is impossible to use application secret keys without and outside the HSM. (More detail provided in [19]).

7.2. Multi-Party Compute

The quality, accuracy and efficiency of AI systems and approaches greatly benefits from the amount and quality of the data that can be used to design, train, and evaluate these systems. Within today's landscape of data governing regulations, data protection rules and policies and the need for enhanced data protection and data privacy measures access and usage of data is often restricted.

There are scenarios where these restrictions are highly desirable and provide value in protecting individuals and intellectual property, in other cases this protection is misused as it allows malicious activates and actors to stay undetected.

Confidential Computing technology can be applied to enable learning and training of AI models on larger, combined sets of data while the data sets stay protected and undisclosed.

AI model training performed within Trusted Execution Environments allows the data to be protected from access outside of the actual training workload. Export and sharing of data from within the TEE is prevented by a combination of the workload and enclave isolation. This approach for model training can be used exclusively or in combination with federated learning models, depending on the specific needs of the scenario and data governance rules.

After the training has been performed the trained model can be used for inference actions outside of the trusted environment to be brough back into the systems benefiting from the insight.

There is a broad set of scenarios where this capability of protected, privacy preserving model training and be applied:

- Financial transaction systems evaluating the likelihood of a transaction to be fraudulent
- Healthcare provider systems identifying health risk present within a medical image
- Retailers identifying instances of crime to be committed by organized groups

Retail Industry Leaders Association (RILA) has labeled Organized Retail Crime (ORC) as a $40 billion problem in the United States. 97% of retailers have been affected by ORC. Every retailer has an Asset Loss / Prevention Team, with their own tracking applications / databases that track loss / pilferage in their retail stores around the United States, but there is no collaboration between retailers, because each retailer regards their own crime data as confidential.

The IBM Hyper Protect Accelerator team proposed and demonstrated a solution to ORC by using Confidential Computing platform like Hyper Protect Virtual Servers (HPVS) with Federated Learning (FL). The proposed “National ORC platform” will allow confidential collaboration between retailers without combining/exposing data, but still draw insights which will help law enforcement bring charges against the perpetrators.

Retailers who participated in the PoC brought live-data to their individual HPVS instances, protected by encryption keys from Hyper Protect Crypto Services (HPCS) with KYOK. The technical assurance of the Hyper Protect platform allowed the Federated Learning models to be created and iterated upon to generate insights.

The insights obtained exposed the following novel ORC patterns in the Home-improvement, Clothing and Fashion segments by using identifying characteristics like license plates, vehicle, physique, clothing, etc.

7.3. Data protection

Data is and was the heart of every company. Based on technology options of the last decade, the possibilities to improve business outcome based on data has grown exponential. With this potential the risk of losing or malicious access to business or private data has grown exponential.

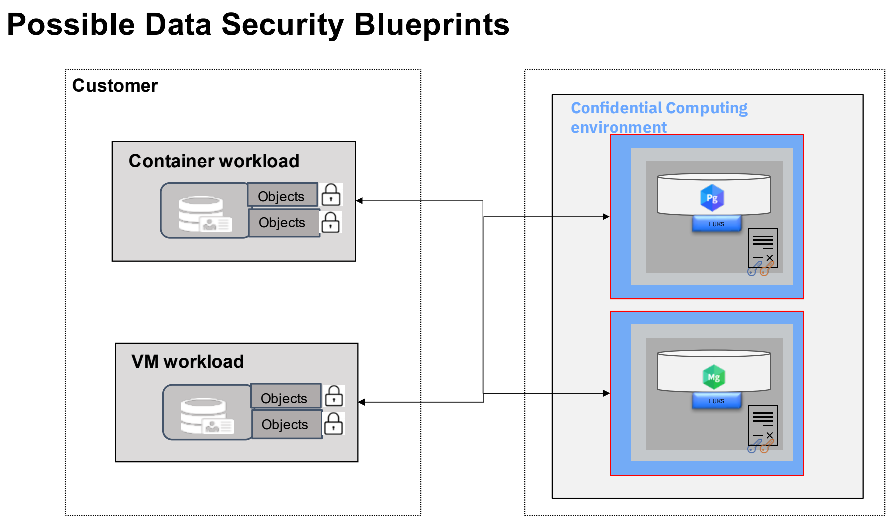


Figure 12: Data protection leveraging Confidential Computing environments

Protecting this data became more and more important to industries like banking or insurance but as well to not high regulated industries like car manufacturers. Establishing an environment with the least possible access to the data will protect personal information of customers, employees, or any other business critical data from external as well as internal miss use. Moving data into a cloud platform reduces the amount of control over the environment and increases the risk to have unwanted data access. Standard data protection patterns like disc encryption or access management always base on the trust of the people who setup the computation environment.

This is no longer given within a cloud environment where the system is setup outside of the data owners control. Using a confidential computing environment will protect the data as the only place to see that data in clear would be inside of the secured environment itself. The option described in this document enables a customer to run standard Databases inside of a high performance and high secure environment without adding risk to expose his data to the cloud provider. It provides control over the data and its access to the customer, again.

7.4. Kubernetes and Confidential Containers

Cloud Computing adoption continues to accelerate whether it be Public, Private, or increasingly common a Hybrid approach and with it the trust boundaries change. Consideration of insider threats needs to now include the cloud provider, infrastructure provider, managed service provider, this means that alongside protecting data-at-rest or data-in-transit within the Cloud we need to protect data in use.

The pursuit of Confidential Computing [20] has given rise to multiple different Trusted Execution Environments (TEE). Using a TEE will create an isolated execution boundary around the application and data, protecting them from actors outside the TEE. However, some actors still need to be able to cross this boundary otherwise the application is of limited value. There will also be practices required to use a specific TEE technology and attest that a workload is running inside a TEE, but are these enough to deliver confidential computing?

To examine what must cross this boundary we can consider two deployment models:

Virtual Machine

Here the application is thought of as a virtual machine with this virtual machine living inside the TEE. The application APIs define what crosses the TEE boundary and the application provider is in complete control of these APIs.

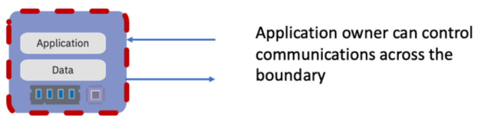

Application
Data
Application owner can control communications across the boundary

Figure 13: Confidential Computing and Virtual Machine

In this model the threats from external actors crossing the boundary can be known, controlled, and mitigated by the application provider alone.

A journey towards Cloud can lead to the containerization of this application, perhaps first a monolithic container deployed into a virtual machine, then a set of tightly coupled containers deployed into their own Virtual Machines. Each of these steps will alter the interactions required across the boundary but they remain in the domain of the application provider. However, the introduction of orchestration starts to move beyond the domain of the application provider.

Orchestrated set of containers

This model aligns with a cloud native approach, our application is a set of containers within a loosely coupled system with an orchestration layer enabling this system to be resilient, manageable, and observable.

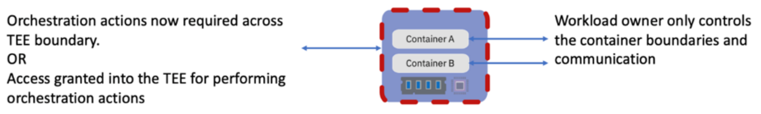

Orchestration actions now required across TEE boundary.
OR
Access granted into the TEE for performing orchestration actions
Container A
Container B
Workload owner only controls the container boundaries and communication

Figure 14: Confidential Computing and Orchestrated set of Containers

We still have APIs to allow interaction with the workload (containers in this case), but we also now require orchestration actions to cross this TEE boundary. We could just allow these orchestration interactions and smooth the path to our cloud native deployment, but can we satisfy ourselves that all interactions (orchestration and workload) across the TEE boundary are secure?

What is inside the TEE boundary needs to control permission for these orchestration actions to happen. More specifically the application provider should have control over both the workload and orchestration interactions across the boundary.

Outlook and Community Contribution

How does the application owner trust the orchestration actions required to deliver on the Cloud Native promise, take advantage of the TEE capabilities and deliver on their compliance/ security requirements?

We are working with others through the CNCF Confidential Containers Project (see [21]) to answer these questions. Our implementation will build on the Hyper Protect Virtual Servers outlined within this document.

8. Summary

Awareness of the risk posed to data privacy in today's digitally enabled connected world is broad within IT decision makes as well as within the technical community maintaining existing and creating new solutions. This risk awareness is fuelling a new curiosity of the technology community and the awareness of confidential computing technologies and concepts has substantially increased. At the same time confidential computing technologies and technology platforms leveraging confidential computing technology are evolving quickly. Both influencing factors are complementing each other and accelerate the rate of technology adoption. Confidential computing technology has gotten much stronger through awareness of threads and invention and implementation of new deflection concepts, at the same time platforms bringing this technology in reach to a much broader technology community of developers and software engineers.

The move of platforms towards container encapsulation and inclusion of confidential computing concepts within cloud native concepts (see [22]) is paving the way for accelerated adoption and new levels of data privacy enabled throughout even more solutions within the hybrid cloud ecosystem.

This paper has covered the state of technology of confidential computing enabled platforms striving to provide total data privacy and enabling developers in the hybrid cloud to easily leverage and integrate with these core concepts without the need to be a cryptographic or data protection expert. However, not all Confidential Computing solutions deliver the same levels of security, flexibility, and seamless developer support. Learning about the platforms and the breadth of services they offer allows for selection of a platform that supports data privacy and regulatory needs of solutions today and in the future. We have accomplished our objective of providing our contribution to this educated understanding of technological and platform capabilities.

I. Bios

Stefan Amann - *IBM Systems, Böblingen, Germany*

Stefan Amann is Senior Technical Staff Member at the IBM Lab in Boeblingen. He studied Computer Engineering at the Vocational Academy Stuttgart. He graduated in 1993 and then joined IBM at the Research and Development Laboratory in Boeblingen as an R&D engineer. He worked as architect on several projects for IBM Z. He is the lead architect for IBM Secure Service Container, and Hosting Appliance since 2018.
He is author or co-author of 32 patents and several technical papers.

Timo Kussmaul - *IBM Systems, Böblingen, Germany*

Timo Kussmaul is a Master Inventor and Solution Architect in the Hyper Protect Services Client Acceleration Team. He received his Dipl. Inform. degree in Computer Science from the University of Stuttgart in 1997 and then joined the IBM Research & Development Lab in Boeblingen. Timo is author or coauthor of more than 85 patents and multiple technical papers and books.

Carsten Leue – IBM Systems, Böblingen, Germany

Stefan Liesche – *IBM Systems, Böblingen, Germany*

Stefan Liesche is the Architect for IBM Hybrid Cloud and Hyper Protect Services on zSystems. Stefan is focused on security, transparency and protection of data and services in flexible cloud environments. Stefan worked in various areas as Technical leader within IBM, most recently as Chief Architect for IBM Cloud Hyper protect Services and IBMs Watson Talent Portfolio where Stefan was building AI driven solutions that transform recruiting and career decisions within global organizations, that not only enhances quality of decisions, but also allows HR functions to enhance fairness and tackle biases. Stefan also innovated within the Exceptional Web Experience products for several years with a focus on open solutions and integration. Stefan has more than 24 years of experience as technical leader, collaborating with partners and customers through joint projects, as well as within IBM's product development organization.

Anbazhagan Mani - *IBM Systems, Bangalore, India*

Anbazhagan Mani is a Master Inventor and Senior Technical Staff Member in the IBM Z and IBM® LinuxONE. Anbazhagan Mani leads the design and development of Hyper Protect Virtual Servers offering on IBM Cloud and on-premises. He has over 20 years of experience at IBM mainly in the systems management & cloud areas. He holds a masters degree from National Institute of Technology (NIT), Trichy, India.

Asha Shekharappa - IBM Systems, Bangalore, India

Asha Shekharappa is a Senior engineer at the IBM Lab in Bangalore. She holds a Bachelor Degree in Computer Science from VTU. She has joined IBM in 2013 and worked on multiple projects in IBM Z. She is the software Architect for IBM Secure Service Container.

Divya K Konoor - *IBM Systems, Bangalore, India*

Divya K Konoor is a Senior Technical Staff Member for the IBM Z Hyper Protect Services Infrastructure as a Service team. She leads the design and development for the infrastructure components that power the Hyper Protect Offerings in IBM Cloud VPC. She has over 16 years of experience in Cloud, Virtualization and Systems Management areas. In her previous role, she was the lead architect driving solutions with IBM POWER® on-premise (PowerVC) and Power Virtual Server cloud for the framework components.

Nicolas Mäding - *IBM Systems, Boeblingen, Germany*

Nicolas Mäding is a Senior Product Manager at the IBM Lab in Boeblingen. He received his Dipl. Ing. Degree in Electrical and Information Technology at the Technical University of Chemnitz. He joined IBM in 2001 and worked in various development and management positions in IBM Systems Hardware Development. After that he joined the Z-as-a-Service organization as Release Manager of Hyper Protect Hosting Appliance in 2018 and became the Product Manager for the Hyper Protect Platform and Confidential Computing with Linux on Z. He is author or co-author of 12 patents and several technical papers.

James Magowan - *IBM Systems, Hursley Park, United Kingdom*

James works within the IBM Hyper Protect family of offerings which deliver Confidential Computing to the Cloud using IBM® LinuxONE and IBM Z Systems technology. He has responsibility for the technical architecture to leverage the IBM Secure Execution for Linux capability (Trusted Execution Environment) in Cloud Native solutions.

James has an MA in Computer Science from the University of Cambridge and over 20 years experience solving customer problems with emerging technologies, contributing to and using open source projects as part of the solution. He is an active contributor to the open source Confidential Containers project which is looking to enable Cloud Native Confidential Computing by leveraging Trusted Execution Environments to protect containers and data.

Peter Morjan - *IBM Systems, Boeblingen, Germany*

Stefan Schmitt - *IBM Systems, Böblingen, Germany*

Stefan Schmitt is a Senior Technical Staff Member in the Hyper Protect Services and IBM Z organization. Stefan is focused on delivering confidential computing services within the IBM Cloud, like IBM Cloud Hyper Protect DBaaS and IBM Cloud Hyper Protect Crypto services. Stefan has more than 20 years of experience as technical leader, collaborating with partners and customers through joint projects. His experience spans from Security, Compliance, Web Development as well as enterprise grade cloud native applications.

II. References

[1- Wikipedia] https://de.wikipedia.org/wiki/Trusted_Execution_Environment

[2 - IBM] Integrating Solutions on IBM Z with Secure Service Container
https://www.ibm.com/downloads/cas/6JWREBWX

[3 - IEEE] Secure your cloud workloads with IBM Secure Execution for Linux on IBM z15 and IBM® LinuxONE III
https://ieeexplore.ieee.org/document/9138728

[4 - IBM] Secure Execution for Linux technical documentation
https://www.ibm.com/docs/en/linux-on-systems?topic=virtualization-introducing-secure-execution-linux

[5 – IBM] Secure Execution for Linux: Ultravisor-powered attestation
https://video.ibm.com/recorded/132127046

[6 - IBM] Attesting a KVM guest running in Secure Execution
https://www.ibm.com/docs/en/linux-on-systems?topic=execution-attesting

[7 - Everest Group] Confidential Computing – The Next Frontier in Data Security

https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/10/Everest_Group_-_Confidential_Computing_-_The_Next_Frontier_in_Data_Security_-_2021-10-19.pdf

[8 - Wikipedia] https://en.wikipedia.org/wiki/Privacy-enhancing_technologies

[9 – The register] Hardware assisted encryption of data in use gets confidential
https://www.theregister.com/2022/11/18/hardwareassisted_encryption_of_data_in/

[10 - Forbes] Why now is the time for Confidential Computing
https://www.forbes.com/sites/forbestechcouncil/2022/03/31/why-now-is-the-time-for-confidential-computing/

[11 – ACM Digital Library] Data Sovereignty in the Cloud - Wishful Thinking or Reality?

https://dl.acm.org/doi/abs/10.1145/3474123.3486792

[12 - CNCF] Annual Survey 2021
https://www.cncf.io/wp-content/uploads/2022/02/CNCF-AR_FINAL-edits-15.2.21.pdf

[13 – IBM] The IBM Hyper Protect Platform - Generation 1
https://www.ibm.com/downloads/cas/V8XERDQO

[14 – IBM] Protecting your image build
https://cloud.ibm.com/docs/hp-virtual-servers?topic=hp-virtual-servers-imagebuild

[15 – IBM] IBM Secure Execution – Cloud Provider tasks
https://www.ibm.com/docs/en/linux-on-systems?topic=execution-cloud-provider-tasks

[16 – IBM] IBM Cloud Virtual Private Cloud documentation
https://cloud.ibm.com/docs/vpc?topic=vpc-getting-started and
https://cloud.ibm.com/docs/vpc?topic=vpc-about-advanced-virtual-servers

[17 – IBM] IBM Cloud CLI documentation
https://cloud.ibm.com/docs/vpc?topic=vpc-infrastructure-cli-plugin-vpc-reference

[18 – IBM] IBM Cloud Hyper Protect Crypto Service documentation on Grep 11 API
https://cloud.ibm.com/docs/hs-crypto?topic=hs-crypto-grep11-api-ref

[19 – Jean-Yves Girard] Protecting Digital Asset HD Wallet using IBM Confidential Computing
https://www.linkedin.com/pulse/protecting-digital-asset-hd-wallet-using-ibm-computing-girard/

[20 – IBM] What is Confidential Computing
https://www.ibm.com/cloud/learn/confidential-computing

[21 – CNCF/GitHub] Confidential Container Project repository
https://github.com/confidential-containers

[22 – CNCF] Confidential Container Project repository
https://www.cncf.io/projects/confidential-containers/

(C)Copyright IBM Corporation 2022

IBM Corporation
New Orchard Road
Armonk, NY 10504

IBM, ibm.com, the IBM logo, IBM Cloud, IBM LinuxONE, IBM Z, POWER and z15 are trademarks or registered trademarks of the International Business Machines Corporation.

A current list of IBM trademarks is available on the Web at https://www.ibm.com/legal/us/en/copytrade.shtml, and select third party trademarks that might be referenced in this document is available at https://www.ibm.com/legal/us/en/copytrade.shtml#section_4.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

InfiniBand and InfiniBand Trade Association are registered trademarks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.

Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

RStudio®, the RStudio logo and Shiny® are registered trademarks of RStudio, Inc.

TEALEAF is a registered trademark of Tealeaf, an IBM Company.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Worklight is a trademark or registered trademark of Worklight, an IBM Company.

Zowe™, the Zowe™ logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided "as is" without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors and are not intended to be a commitment to future product or feature availability in any way.